Francisco Sebastião da Silva

Desenvolvimento de metodologias para classificação automática de imagens de satélite a partir de dados de treino gerados sem intervenção de operador

Dissertação de Mestrado em Ciências e Tecnologias de Informação Geográfica, orientada por José Paulo Elvas Duarte de Almeida e Joaquim António Saraiva Patriarca, e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

2023

· U C ·

Universidade de Coimbra

Francisco Sebastião da Silva

Desenvolvimento de metodologias para a classificação automática de imagens de satélite a partir de dados de treino gerados sem intervenção de operador

Dissertação de Mestreado em Ciências e Tecnologias de Informação Geográfica, orientada por José Paulo Elvas Duarte de Almeida e Joaquim António Saraiva Patriarca, e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

2023

UNIVERSIDADE DE COIMBRA

AGRADECIMENTOS

A toda a minha família, e em especial para os meus pais e irmão, fonte de educação, inspiração e de apoio ao longo de todo o meu percurso académico. Um grande obrigado pela vida que me proporcionaram e por terem feito de tudo o que estava ao seu alcance para a minha formação educacional. Este documento também é vosso.

À minha namorada Bruna, pelo apoio, transmissão de entusiasmo, de bom humor e paciência.

Ao Professor Doutor José Paulo Elvas, pela orientação de excelência, pelo intelecto rigor e disponibilidade infatigável, sem os quais o desenvolvimento desta Dissertação não teria sido possível.

Ao Professor Mestre Joaquim Patriarca, pilar dos conhecimentos desenvolvidos, imprescindíveis para a realização deste estudo. Muito obrigado pela qualidade de orientação, pela dedicação, paciência, disponibilidade e amizade. Os seus ensinamentos foram muito além do espectado, soube despertar a admiração de um modo único, tornando-se numa inspiração.

Com certeza que há pessoas que marcam a nossa vida, que despertam algo de especial em nós, que abrem os nossos olhos de modo irreversível, os Professores foram sem dúvida algumas dessas pessoas.

A todos os Professores do Departamento de Matemática da Universidade de Coimbra, pela transmissão de conhecimentos e aprendizagens ao longo do percurso académico deste Mestrado.

Resumo

Ao longo das últimas décadas, devido ao crescente desenvolvimento das comunidades e à recorrente incidência de catástrofes naturais, aliada das alterações climáticas, têm-se observado transformações nas características da superfície terrestre a olhos vistos. Ao tomar consciência desta realidade, os investigadores incentivam a procura de novas metodologias para a produção de mapas de uso e ocupação do solo, com o objetivo de entrar na corrida através da frequente atualização dos produtos, emancipando métodos de criação de mapas mais tradicional, dependendo de longos períodos de produção e despesas avultadas. De certo modo, os mapas de uso e ocupação do solo constituem informação vital em diferentes domínios, e metodologias automáticas para a sua criação, como é caso de exemplo o estudo introduzido por Fonte, *et al.* (2020), que fundamenta métodos essenciais como pontos de partida no domínio da produção de mapas de ocupação do solo de forma rápida e de exatidão temática.

O quadro acima descrito levou ao desenvolvimento da presente investigação, de modo a estabelecer metodologias programadas para gerar de dados de treino extraídos da informação disponibilizada por comunidades de informação geográfica voluntária, tendo em vista os métodos apresentados pelo trabalho mencionado, com o intuito de aperfeiçoar componentes práticos para a filtragem de dados de treino e posterior classificação automática de imagens de satélite. Neste contexto, são propostas estratégias automáticas para criar dados de treino a partir da informação OpenStreetMap (i.e. sem a intervenção direta do utilizar), através da identificação de novas regras de filtragem, auxiliadas pelo cálculo de índices radiométricos, para a classificação combinada de várias fontes de imagens de satélite e dados de elevação.

Para as ideias propostas, que dizem respeito à otimização da metodologia, considerou-se a redefinição de regras para filtrar os dados de treino, por meio da inserção de novos índices radiométricos. Esta metodologia envolve, também, a identificação de objetos com características particulares de uma única classe do solo a partir de técnicas de segmentação comparadas com um mapa de referência para seleção de amostras de informações únicas para as determinadas classes, considerando a variação dos valores dos índices radiométricos nos objetos. Entre os diferentes conjuntos de dados de treino filtrados, e a combinação de imagens de satélite (Sentinel-2 e Landsat-8) e dados altimétricos (MDE), fez-se uma comparação e análise de resultados das classificações obtidas, validados por intermédio de

duas estratégias, auxiliadas pelo mapa de referência CLC + *Backbone* e uma amostra de pontos aleatórios.

Das várias metodologias abordadas, estas foram testadas para a área de estudo definida como a ilha Terceira do arquipélago dos Açores. A classificação e geração de mapas de ocupação do solo, produzidos através do algoritmo *Random Forest*, revelaram resultados promissores. Numa primeira fase de análise de resultados, percebeu-se que a combinação de imagens de satélite e informação altimétrica aumentava a exatidão global dos mapas, em comparação com a utilização singular de dados de um satélite. Numa fase de análise seguinte, os resultados mostraram que os processos de filtragem dos dados melhoravam as exatidões das classificações com exatidões globais superiores a 80% para os dados de treino produzidos através das regras de filtragem introduzidas no estudo de Fonte, *et al.* (2020) (F1). Ademais, a metodologia introduzida e o desenvolvimento de novas regras de filtragem revelaram uma melhoria de sequência gradual dos dados de treino com a introdução de novas regras de filtragem para os índices radiométricos utilizados no artigo de referência (F2) para as novas regras de filtragem com a introdução de outros índices radiométricos (F3) na etapa de treino do classificador, com percentagens de exatidão superiores a 90%.

Palavras-Chave: Dados de treino; Mapas de ocupação do solo; CLC+ *Backbone*; Imagens de satélite; Regras de filtragem.

ABSTRACT

Over the last few decades due to the growing development of communities and the recurring incidence of natural disasters combined with climate change, the characteristics of the earth's surface has transformed at a rapid pace. By realizing this reality, researchers are encouraging the search for new methodologies for the production of land use and occupation maps, with the aim of entering the race by frequently updating products and emancipating more traditional methods of creating maps which depend on long production times and high costs. In a way, land use and occupation maps constitute vital information in different domains, and automatic methodologies for their creation, such as the study introduced by Fonte, *et al.* (2020), which bases essential methods as starting points in the field of producing land use maps quickly and with thematic accuracy.

The framework described above led to the development of this research, in order to establish programmed methodologies for generating training data extracted from the information made available by voluntary geographic information communities, with a view to the methods presented by the mentioned work, with the aim of perfecting practical components for filtering training data and the subsequent automatic classification of satellite images. In this context, automatic strategies are proposed for creating training data from OpenStreetMap information (i.e. without direct user intervention), by identifying new filtering rules aided by the calculation of radiometric indices, for the combined classification of various sources of satellite images and elevation data.

The proposed ideas for optimizing the methodology include redefining the rules for filtering the training data by inserting new radiometric indices. This methodology also involves identifying objects with particular characteristics of a single soil class using segmentation techniques compared with reference maps to select samples of unique information for the given classes, taking into account the variation in the values of the radiometric indices in the objects. The different sets of filtered training data and the combination of satellite images (Sentinel-2 and Landsat-8) and altimetric data (MDE) were compared and the results of the classifications obtained were analysed and validated using two strategies, aided by CLC + Backbone reference maps and a sample of random points.

Of the various methodologies discussed, these were tested for the study area defined as Terceira Island in the Azores. The classification and generation of land cover maps produced using the Random Forest algorithm, showed promising results. In a first phase of analysing

the results, it was realized that the combination of satellite images and altimetric information increased the overall accuracy of the maps, compared to using satellite data alone. In a subsequent analysis phase, the results showed that data filtering processes improved classification accuracy, with overall accuracies of over 80 per cent for the training data produced using the filtering rules introduced in the study by Fonte, *et al.* (2020) (F1). In addition the methodology introduced and the development of new filtering rules revealed a gradual sequence improvement from the training data with the introduction of new filtering rules for the radiometric indices utilised in the reference article (F2) to the new filtering rules with the introduction of other radiometric indices (F3) in the training stage of the classifier with accuracy percentages of over 90%.

Key-words: Training data; Land cover mapping; CLC+ Backbone; Satellite images; Filtering rules.

ABREVIATURAS, ACRÓNIMOS E SIGLAS

AI – *Artificial intelligence*

ALOS – *Advanced Land Observing Satellite*

API – *Application Programming Interface*

ASF – *Alaska Satellite Facility*

AIST – *Advanced Industrial Science and Technology*

ASTER – *Advanced Spaceborne Thermal Emission and Reflectation Radiometer*

C-SAR – C-*band* SAR

CCI – *Coastal Changes and Impacts*

CDED – *Canadian Digital Elevation Dataset*

CDED1 – *Canadian Digital Elevation Data Level 1*

CDED3 – *Canadian Digital Elevation Data Level* 3

CH – *Landscape CHARACTERISTICS*

CLC – *CORINE Land Cover*

CLC+ BB – *CORINE Land Cover plus Backbone*

CNN – *Convolutional Neural Network*

COS – Carta de Ocupação do Solo

COTS – *Comercial Off-The-Self*

CPU – *Central Processing Unit*

DEM – *Digital Elevation Model*

DGT – Direção Geral do Território

DN – *Digital Numbers*

DT – *Decesion Tree*

DTED1 – *Digital Terrain Elevation Data 1*

DTED2 – *Digital Terrain Elevation Data 2*

EAGLE – *EIONET Action Group on Land Monitoring in Europe*

EEA – *European Environment Agency*

EGM96 – *Earth Gravitational Model 1996*

EIONET – *Environment Information and Observation Network*

EO – *Earth Observation*

EORC – *Earth Observation Research Center*

ESA – *European Space Agency*

Esri – *Environmental System Research Institute*

ETM+ – *Enhanced Thematic Mapper Plus*

ETRS89 – *European Terrestrial Reference System 1989*

EU – *European Union*

EU-DEM – *European Digital Elevation Model*

EU-Hydro – *European Hydrography Database*

Fmask – *Function of mask*

FOSS4G – *Free and Open Source Software for Geospatial*

GDEM – *Global Digital Elevation Model*

GEOS – *Geometry Engine, Open Source*

GLC2000 – *Global Land Cover 2000*

GLASS – *The Geoscientific Library for Analysis of Spatial Systems*

GML – *Gaussian Maximum likelihood*

GMTED2010 – *Global Multi-resolution Terrain Elevation Data*

GTOPO30 – *Global 30 Arc-Second Elevation*

HDF5 – *Hierarchical Data Format version 5*

HiRI – *High Resolution optical Imagers*

ICESat – *Ice, Cloud, and Elevation Satellite*

IGV – Informação Geográfica Voluntária

InSar – *Interferometric Synthetic Aperture Radar*

JAXA – *Japan Aerospace Exploration Agency*

kNN – *K-Nearest Neighbor*

L1TP – *Level 1 Precision and Terrain*

LAEA – *Lambert Azimuthal Equal Area*

LC – *Land Cover*

LCC – *Land Cover Component*

LR – *Logistic Regression*

LUA – *LAND USE attributes*

LULC – *Land Use/ Land Cover*

MBRF – *Multiclass Border Reduction Filter*

MDE – Modelo Digital de Elevação

MMU – *Minimum Mapping Unit*

MSAVI – *Modified Soil Adjusted Vegetation Index*

MSI – *Multispectral Instrument*

NAOMI – *New AstroSat Optical Modular Instrument*

NASA – *National Aeronautics and Space Administration*

NBR – *Normalized Burn Ratio.*

NDBI – *Normalized Difference Built-up Index*

NDVI – *Normalized Difference Vegetation Index*

NDWI – *Normalized Difference Water Index*

NED – *National Elevation Dataset* – Alaska

NED1 – *National Elevation Dataset 1 arc-second*

NED2 – *National Elevation Dataset 2 arc-second*

NED13 – *National Elevation Dataset 1/3 arc-second*

NGA – *National Geospatial-Intelligence Agency*

NICT – *National Institute of Information and Communications Technology*

NIR – *Near Infrared*

NN – *Nearest Neighbor*

NNs – *Neural Networks*

NB – *Naive Bayes*

METI – *Japan's Ministry of Economy, Trade, and Industry*

OLC – *Open Land Cover*

OLI – *Operational Land Imager*

OSGeo – *Open Source Geospatial Foundation*

OSM – OpenStreetMap

PALSAR – *Phase Array type L-band Synthetic Aperture Radar*

PAN/HYP – *Panchromatic and Hyperspectral*

PCA – *Principal Component Analysis*

RF – *Random Forest*

RMSE – *Root Mean Square Error*

RTC – *Radiometric Terrain Correction*

SAR – *Synthetic Aperture Radar*

SCL – *Scene Classification Maps*

SIG – Sistemas de Informação Geográfica

SQL – *Structured Query Language*

SRTM – *Shuttle Radar Topography Mission*

SRTM GL1 – *Shuttle Radar Topography Mission Global 1 arc-second*

SRTM US1 – *Shuttle Radar Topography Mission United States 1 arc-second*

SVM – *Support Vector Machine*

SWIR – *Short Wavelength InfraRed*

TIG – Tecnologias de Informações Geográficas

TIRS – *Thermal Infrared Sensor*

TM – *Thematic Mapper*

TOA – *Top of Atmosphere*

UA – *Urban Atlas*

UML – *Unified Modeling Language*

USA – *Unites States of America*

USGS – *United States Geological Surve*

UTM – *Universal Transverse Mercator*

VHP – *Volcano Hazards Program*

WGS84 – *World Geodetic System 84*

ÍNDICE GERAL

Índice De Figuras

Índice De Tabelas

1 Introdução

A procura de metodologias inovadoras, relacionadas com a filtragem de dados de treino para produção de mapas de ocupação do solo a partir da classificação automática de imagens de satélite, através de tecnologias que envolvem a automatização de processos, é uma área de investigação importante e em constante evolução. Com o crescimento exponencial da disponibilização de dados de imagens de satélite e dados produzidos por projetos de Informação Geográfica Voluntária (IGV), são também impulsionadas as necessidades de desenvolver métodos eficazes, para compreender até que ponto o processo se pode realizar de forma totalmente automática, com qualidade aceitável ou ao nível dos resultados obtidos com um treino gerado de forma manual. Os métodos referidos devem ser, também, capazes de extrair informação relevante e com qualidade suficiente para gerar este mesmo treino, podendo posteriormente, ser utilizado para a classificação de imagens de satélite, automatizando todo o processo, ao contrário do que acontece na maioria dos casos.

É no seio do que foi mencionado anteriormente, que se desenvolve a presente dissertação. Assim, para dar início ao documento, é necessário estabelecer marcos que justificam a sua razoabilidade, bem como, metas que delineiam os percursos mais adequados para alcançar a contextualização indicada. O primeiro capítulo, tem o propósito de permitir uma familiarização intuitiva, do enquadramento da problemática e motivação do trabalho, seguida da definição dos objetivos do trabalho.

1.1 Enquadramento da problemática e motivação do trabalho

A motivação e problemática do trabalho são o motor de arranque, após a tomada de decisão que havia conduzido à seleção da temática. O referido capítulo, foca-se assim, na problemática referente à investigação neste documento, com o intuito de permitir ao leitor um conhecimento prévio, optando por apresentar uma síntese dos pilares mais relevantes que envolve:

i. As rápidas alterações da superfície terrestre, e a crescente necessidade de monitorização da informação relativa à ocupação e uso do solo, por meio de métodos de produção de mapas de forma automatizada, rápida e eficaz;

ii. A produção de mapas de qualidade segue com o intuito de analisar como é que as estratégias desenvolvidas no estudo podem ser integradas nos processos de produção de cartografia de uso e ocupação;
iii. Na perspetiva dos métodos automatizados para produção de mapas e tendo por base inúmeras motivações, entre elas a utilização de uma metodologia de referência introduzida no artigo Fonte, *et al.* (2020). A referida investigação patenteia resultados interessantes, nomeadamente no que diz respeito à classificação de imagens de satélite. Porém, como em qualquer outra investigação, existem algumas lacunas, pontos que não terão sido abordados da melhor forma e, portanto, equaciona-se uma intervenção nesta metodologia, tendo por base, os métodos de filtragem aplicados aos dados, com vista a obter novos resultados;
iv. Uma vez que as classificações e produção de mapas de uso e ocupação do solo requerem, indiscutivelmente, a utilização de dados, pretende realizar-se a análise das capacidades de classificação de imagens de satélite através da aplicação única e exclusiva de fontes de dados disponíveis de forma gratuita;
v. Para as fontes de dados de imagens de satélite, acrescenta-se a procura por uma classificação de imagens com recurso a outras fontes;
vi. Na medida em que o território português carece de informação para algumas regiões do país (como é o caso do arquipélago dos Açores), considera-se como uma agravante que lamenta dados, extremamente importantes atualmente.

Tudo aquilo que foi mencionado, faz alusão ao enquadramento da problemática e motivação do trabalho, servindo como ponto de partida para o início do desenvolvimento desta dissertação, dada à ambição e curiosidade pela procura de respostas para estas questões.

1.1.1 Produção de mapas de uso e ocupação do solo: Abordagens tradicionais vs. Abordagens semi-automáticas

A crescente necessidade pelo conhecimento sobre a informação presente à superfície terrestre, como forma de explicar a variação da ocupação e uso do solo, são considerados como temas imprescindíveis para a compreensão de territórios de discussão recorrentes nas organizações governamentais por todo o mundo. A representação destas transformações através da conceção de mapas de uso e ocupação do solo, são considerados como meios que possibilitam tomadas de decisão para causas como, segurança alimentar, ordenamento do

território, geolocalização de águas superficiais, gestão de recursos e muitos outros pontos de interesse.

Mapas de uso e ocupação do solo são ferramentas constantemente produzidas para responder a questões desta natureza, no entanto, dados os processos naturais e a evolução da atividade humana, estes podem desatualizar-se num curto período de tempo, verificando-se incompatibilidades com a realidade. Considerando, a produção de mapas, como uma abordagem de longa durabilidade, desde a recolha de dados geográficos, processos de tratamento e seleção, até alcançar os produtos finais, torna-se difícil acompanhar as rápidas transformações da superfície, com origem antrópica ou natural. Destes podem destacar-se respetivamente incêndios florestais, desflorestação, incidentes provocados por catástrofes naturais e até mesmo o avanço das alterações climáticas.

A produção de mapas de qualidade assenta, na grande maioria, em técnicas tradicionais, com auxílio de pesquisas de campo e fotointerpretação, e métodos que dependem da intervenção e deslocação do especialista. Estes métodos desenvolvem mapas de legendas com um grau de detalhe elevado, porém limitados em termos de atualização, devido a altos custos, tornando-se difícil aceder a metodologias que requerem estas práticas, constantemente.

Contudo, devido ao desenvolvimento das novas tecnologias, as metodologias têm vindo a ser substituídas por métodos automáticos, que passam pela classificação de imagens de satélite, permitindo visões consistentes e amplas de variáveis de interesse. Em virtude do aumento da disponibilidade de conjuntos de dados de acesso livre, estes têm vindo a assumir o papel responsável do desenvolvimento de metodologias, para classificação de imagens de satélite mediante da utilização de tecnologias automáticas.

A alta resolução, em comunhão com a vasta cobertura de satélites e dados produzidos por grupos IGV, possibilitam o acesso a imagens e informações em tempo real, e a consequente criação de mapas atualizados. Estas técnicas requerem a utilização de *softwares* SIG, relacionadas com a implementação de tecnologias aliadas à programação de uma linguagem informática, exploração de bibliotecas programadas e algoritmos de classificação.

1.1.2 Princípios e normas legislativas para a produção cartográfica em Portugal

Em Portugal, a produção de cartografia é considerada uma ferramenta central para a gestão do território, e de todas as atividades que englobam uma componente espacial. A informação cartográfica nacional é essencialmente da responsabilidade da DGT, principal agente criadora de cartografia oficial do país, todavia a conceção de cartografia pode também passar por câmaras municipais, pequenas instituições e empresas. Esta organização, é a única, em território nacional, que detém cartografia considerada oficial ou certificada, utilizada em operações oficiais na administração pública, já a informação geoespacial produzida por outros agentes dá designa-se "mapeamento" ou cartografia privada, produzida para fins particulares em ambientes SIG.

Esta merece destaque, pois para além da produção, manutenção e relacionamento das informações geográficas, tem o dever de divulgar todo o conjunto de dados às comunidades. Promove cartografia de média escala, assim como oferece cobertura da superfície de Portugal Continental através de ortofotos, e ainda a frequente produção de cartografia temática vetorial da COS.

No âmbito da cartografia temática de ocupação e uso do solo COS 2018, poder-se-á criticar a instituição pela cobertura incompleta do território nacional, dada à carência do produto cartográfico para os territórios insulares, nomeadamente para o arquipélago dos Açores. Contudo, é pertinente ter em consideração que, devido a normas legislativas, a organização está isenta de produzir informação cartográfica para as regiões autónomas.

Na medida das circunstâncias de divisões administrativas, os territórios insulares gozam de autonomia própria, em virtude das características geográficas, económicas, sociais e culturais, concedendo-lhes administração política própria. Esta autonomia é fortalecida, pela alteração dos princípios e normas que obedecem à produção cartográfica no território nacional, que segundo o Decreto-Lei n.º 130/2019, de 30 de agosto, vem avigorar competências às regiões autónomas para a produção e homologação de cartografia topográfica vetorial, cartografia topográfica de imagem e cartografia temática oficial concebida nos respetivos territórios, com atribuição conjunta de poder para fiscalização. Não se deve deixar de destacar, que quando se trata de cartografia temática, as entidades responsáveis pela produção devem assegurar a cartografia topográfica vetorial, cartografia topográfica de imagem ou hidrográfica utilizada na produção como oficial.

Na sequência do decreto, as normas estabelecem o poder das regiões autónomas, para a produção da sua própria cartografia, considerando a utilização de informações provenientes de cartografia oficial, de forma que esta se considere igualmente como oficial. Neste sentido, a DGT está fora deste paradigma, sem qualquer responsabilidade que visa a conceção de informação cartográfica para estes territórios. Em relação ao produto de informação vetorial da COS 2018, reconhece-se a falta de cobertura de dados para as ilhas, mediante dos princípios e normas legislativas para a produção cartográfica em Portugal.

1.2 Objetivos do trabalho

Perante a temática selecionada para o trabalho, reconhecendo o valor potencial dos resultados ambicionados, é necessário admitir que nem todas as metas são alcançáveis, devido à complexidade das abordagens que podem ser exploradas, combinado o tempo disponível. É preciso notar, que em qualquer trabalho investigativo, a aprendizagem consistente e a busca pelo novo conhecimento são aliadas, e muitas vezes, devido a uma gestão de tarefas imprópria, pode conduzir a longos períodos de execução limitando determinados objetivos desejados, levando a uma redefinição de novas metas.

Com base nos aspetos motivacionais referidos, procuraram-se definir metas e objetivos mais modestos, de cumprimento exequível, embora não se devam considerar de valor inferior. Posto isto, como o presente trabalho investigativo procura responder às questões e motivações colocadas, tendo por base as competências técnicas no âmbito dos SIG, o principal objetivo enfoca-se no desenvolvimento de metodologias para criação de mapas temáticos a partir da classificação automática de imagens de satélite auxiliada da aplicação de dados de treino gerados sem a intervenção direta do utilizador. Esta classificação envolve a utilização assente de imagens Sentinel-2, e o treino de um algoritmo de classificação, através de dados extraídos do OSM, para a classificação das imagens de satélite.

Na sequência da definição do principal objetivo, o surgimento de evidências para a materialização de metas mais específicas, é imprescindível. Estes pilares relacionam-se inteiramente com a problemática e motivação, assim como são oportunos, no sentido que, integram numa estrutura lógica e coerente do trabalho, proporcionando o seu enriquecimento.

Pela sua abrangência, os objetivos específicos, de suporte à concretização do objetivo primordial, adiantam-se nos seguintes pontos:

i. Produzir mapas temáticos de ocupação do solo para uma das áreas dos territórios insulares do país;
ii. Executar a metodologia descrita e implementada por Fonte *et al.* (2020) e produzir resultados, a partir de processos programados com recurso a várias tecnologias;
iii. Avaliar o impacto de utilização de mais imagens na classificação, com o aumento do número de bandas usadas por imagem;
iv. Adicionar outras variáveis ao processo de classificação, nomeadamente Modelos Digitais de Elevação (i.e. EU-DEM) e outro tipo de imagens de satélite (ex. Landsat-8);
v. Refinar os limiares de referência e as regras que definem se uma dada célula da matriz *raster* deve ser incluída ou excluída dos dados de treino. Esta meta específica, subdivide-se em:
    a. Identificar vários índices radiométricos e saber que valores costumam estar associados a cada classe de ocupação do solo;
    b. Este último aspeto pode ser apenas conseguido da seguinte forma:

        1) escolher uma ou mais regiões com a COS 2018 disponível;

        2) identificar manualmente uma amostra estratificada de polígonos que contêm apenas elementos pertencentes à classe (ex. floresta sem qualquer outro tipo de ocupação; territórios artificializados sem áreas verdes entre edifícios);

        3) considerando os polígonos da amostra, avaliar, para cada classe, a distribuição dos valores dos vários indicies radiométricos;

        4) retirar conclusões sobre os limiares de referência que devem ser usados para indicar se uma célula deve ser incluída ou não no treino.
vi. Produzir várias amostras de treino e treinar o algoritmo de classificação, consoante o número do conjunto de dados de treino;
vii. Comparar os resultados produzidos no ponto anterior, através de medidas de exatidão da classificação.

2 Enquadramento conceptual e revisão da literatura

A pesquisa realizada na presente secção do documento foi desenvolvida através da análise de diferentes artigos e estudos científicos, servindo como base teórica para o aprofundamento de conhecimentos específicos, no contexto da problemática exposta.

Serão abordados os mapas de ocupação e uso do solo, com distinção dos dois conceitos, principais algoritmos de classificação de imagens de satélite, índices radiométricos e o seu desempenho no auxílio da aplicação de filtros para definição de classes de ocupação do solo. Além destes tópicos, serão também evidenciados os diferentes produtos de mapas existentes e respetivas nomenclaturas associadas, assim como a descrição de artigos, no capítulo 2.2, cujos temas focam no desenvolvimento de técnicas e metodologias de classificação de imagens em combinação com as práticas de filtragem de dados de treino, através de processos totalmente automatizados e alguns procedimentos manuais. Estes artigos servirão de base, na divulgação de ferramentas aplicadas para implementação das várias metodologias, completadas pelas respetivas descrições e resultados adquiridos. Será feita uma abordagem a conhecimentos acerca de:

2.1.1 Mapas de ocupação e uso do Solo (*Land Use/Land Cover* – LULC) - Distinção entre uso do solo e ocupação do solo;

2.1.2 Algoritmos de classificação, apresentação dos classificadores e descrição das respetivas características e particularidades;

2.1.3 Índices radiométricos, divulgação da quantificação individual da aplicação dos mesmos e das respetivas fórmulas de cálculo;

2.1.4 Produtos de mapas existentes, destaque dos principais produtos de cartografia de uso e ocupação do solo, e nomenclaturas vinculadas.

Em adição à descrição dos conceitos referidos, serão incluídas revisões literárias na secção 2.2 destinada ao Estado de Arte, vinculando documentos centrados na extração e filtragem automática de dados para a criação de conjuntos de dados de treino, para a posterior classificação de imagens de satélite, assim como na análise do documento referente à implementação do *software* OSM2LULC. Do mesmo modo, artigos que remontem para informação da mesma amplitude serão mencionados pelo elevado contributo para o desenvolvimento metodológico da dissertação.

2.1 Enquadramento conceptual

Com o intuito de apresentar noções explicativas relativas ao tema deste projeto, o seguinte capítulo, pretende expor abordagens para criação de mapas de ocupação do solo, através da aplicação de metodologias de classificação automática de imagens de satélite. Atualmente, a distribuição geospacial do uso e ocupação da superfície terrestre é de especial interesse na investigação, e tomada de medidas essenciais para a organização de comunidades, dado que vivemos num ambiente onde a informação deste caráter é de extrema importância para a resolução de problemas em diferentes domínios do conhecimento, bem como ao nível divergente entre áreas rurais e urbanas.

A criação de mapas LULC e a adaptação às novas tecnologias, para o desenvolvimento de metodologias que proporcionam informação tão valiosa, são pontos de interesse na comunidade científica, procurando constantemente maior detalhe nos mapas produzidos, a baixo custo. Tradicionalmente, a criação de dados e mapas LULC são baseados na classificação de imagens obtidas por sensores ou imagens aéreas, todavia devido à evolução tecnológica, foram surgindo novos métodos para aquisição de dados. A elevada capacidade de aquisição de imagens e a ampla cobertura de satélites remotos, como por exemplo os satélites Landsat-8 e Sentinel-2, possibilitam o desenvolvimento de metodologias para a produção automática de mapas LULC, devido à sua alta resolução temporal, espacial e radiométrica. Para além da alta cobertura da superfície por imagens de satélite, projetos direcionados à IGV, como o OSM, um dos programas pioneiros na disponibilização de dados geoespaciais de livre acesso, consideram-se uma forte contribuição, para os mapas de uso e ocupação do solo. A sua ampla cobertura e atualização por parte da comunidade de utilizadores, permite dados diversificados e completos, podendo ser aproveitados para o desenvolvimento de novas metodologias na criação de dados de treino.

Os dados do OSM podem ser utilizados para a produção de dados de treino, e posterior definição de classes LULC das nomenclaturas. Como exemplo de nomenclatura, a CLC, uma das mais bem-conceituadas, suma conhecimentos sobre a ocupação e o uso do solo (LC/LU) com vários conjuntos de classes, subdivididas por diferentes níveis de pormenorização. Graças ao esforço e dedicação colocados por parte dos desenvolvedores da Universidade de Coimbra, o *software* OSM2LULC do estudo feito por Patriarca, *et al.* (2019) proporciona a conversão automática de dados recolhidos do OSM para a nomenclatura referida. A partir dos dados de saída, produzidos pela execução do programa, estes são submetidos a diferentes

processos de filtragem e harmonização, caso seja necessário realizar a transição de uma nomenclatura para outra pretendida, antes de alcançarem a dados de treino, preparados para a classificação de imagens de satélite.

Os índices radiométricos são a base da aplicação de processos de filtragem dos dados retirados do OSM em bruto, impactantes na discriminação de diferentes classes da nomenclatura, que posteriormente origina uma maior exatidão temática no resultado dos mapas LULC (Fonte, *et al.* 2020). Por outro lado, a escolha do algoritmo de classificação de imagens de satélite torna-se, também, relevante para a produção do resultado da classificação dos mapas LULC (Noi, *et al.* 2017). No presente subcapítulo é realizada uma abordagem dos conceitos e noções relevantes para a compreensão do trabalho investigativo, através da sua explicação detalhada.

2.1.1 Mapas de Uso e Ocupação do Solo (*Land Use/Land Cover* - LULC)

A constante recolha de informação relativa ao uso e ocupação da superfície terrestre, a partir de imagens de satélite, é fundamental para desenvolver projetos e relatórios de monitorização, servindo também de apoio para atividades científicas. O livre acesso às imagens de satélite, combinado com o aumento do poder computacional e o desenvolvimento de metodologias automáticas para processamento de dados, proporcionaram um avanço na capacidade de criação de mapas LULC. A exatidão dos mapas está diretamente influenciada pelos dados de treino usados, bem como processos de filtragem submetidos e algoritmos de classificação incluídos no modelo, mas deve ter-se também em conta, a própria qualidade das imagens utilizadas.

Os termos de uso e ocupação do solo são e devem ser utilizados de forma discriminativa, uma vez que determinam conceitos com significados e sentidos distintos. Enquanto a ocupação da superfície terrestre se refere, por exemplo à vegetação, água, solo descoberto, espaços urbanos, entre outras, com informações básicas acerca do que se encontra realmente à superfície, o uso do solo enquadra-se no propósito das atividades e utilização do solo, por exemplo agricultura, espaço cultural, área desportiva, etc.

Um mapa de uso e ocupação do solo permite realizar a representação da classificação de uma dada área, contendo a sua ocupação e o uso que lhe é atribuído, consoante as atividades e o seu propósito. Define-se como um mapa LULC, a simples correspondência de cada ponto

da região em questão a uma ou mais classes de uso e ocupação de solo, ainda assim os mais comuns são caracterizados por polígonos ou áreas que assumem uma relação única a apenas uma classe de uso e/ou ocupação do solo (Bédard, *et al.* 2015). Neste sentido, tendo em vista que cada ponto situado no interior de uma área ou polígono pertence à respetiva classe temática, permite afirmar que existe uma relação direta.

O processo de criação de mapas, pode ser feito de forma manual, ou seja, com o auxílio de imagens de satélite ou ortofotos fazendo a foto-interpretação daquilo que é visível ao olho do cartografo, completado com trabalho de campo. É interessante reconhecer que grande parte dos produtos oficiais, ou seja, mapas de uso e ocupação gerados com informações oficiais, dependem bastante destas metodologias mais tradicionais. A cartografia desenvolvida por estes especialistas, requer um contexto técnico e artístico aliado a princípios geográficos, conferindo um papel científico devido ao estudo, análise e representação de informação para o território (Buchroithner & Fernández, 2011).

Todavia, as metodologias manuais encontram-se num paradigma de uma substituição gradual por métodos automatizados, uma alternativa para a produção de mapas de forma rápida e de fácil atualização, limitando custos monetários. Estes métodos requerem da utilização de imagens de satélite com características que permitem realizar a correspondência de cada pixel da imagem a uma determinada característica do solo, para a posterior definição de uma respetiva classe temática. (Hu, *et al.* 2013). Aliado a questões de representação dos objetos da superfície terrestre, relativamente às metodologias mais tradicionais, estes métodos automáticos permitem do mesmo modo, a produção de mapas de informações sobre o uso e ocupação do solo.

Para melhor entendimento acerca dos conceitos mencionados, nos subcapítulos seguintes serão apresentadas as suas definições e informação, de forma a individualizar e apresentar diversidades com a intenção de esclarecer o seu significado, uma vez que, nesta dissertação será desenvolvido e selecionado, apenas um dos termos na produção da mapas e definição da nomenclatura.

2.1.1.1 *Uso do Solo* (Land Use)

O termo de uso do solo engloba um conjunto de características relacionadas com as atividades das comunidades e o seu desenvolvimento espacial na representação do espaço territorial. Quando se realiza uma classificação de imagem de satélite, para além de se monitorizar tudo o que se encontra à superfície terrestre, está-se sobretudo a reconhecer e a identificar todas as atividades humanas no espaço e no tempo, com base nas informações espectrais e em conjunto com trabalho auxiliar de fotointerpretação e de campo (Nandaia, 2020). De uma forma genérica, o uso do solo documenta uma porção da superfície que diz respeito aos fins socioeconómicos para satisfação das sociedades, tendo em consideração, por exemplo, as áreas comerciais, as áreas de residência, fins recreativos e culturais (Sabins, 2007).

Este conceito considera algo mais conceptual, e é importante realçar que o uso do solo alude à "pegada" introduzida pelo Homem na paisagem natural, com a intenção de tirar proveito daquilo que é oferecido, garantido comodidade e bem-estar (Donnay, *et al.* 2000).

Em nomenclaturas de uso e ocupação do solo existe uma certa relação entre os dois conceitos, ou seja, uma classe de uso do solo pode estar diretamente relacionada com a ocupação. Num simples exemplo em que se considera uma classe de uso do solo como "agrícola", entende-se que a ocupação será, quase obrigatoriamente, designada por uma classe de "culturas permanentes" ou "culturas temporárias". Neste pequeno caso, reconhece-se que a ocupação da classe será um tipo de cultura vegetal ou arbórea com uso do solo direcionado à atividade agrícola.

Contudo, a definição de ocupação do solo diz respeito a outra terminologia que abrange características especificas e discriminativas do conceito de uso do solo, como serão descritas no subcapítulo seguinte.

2.1.1.2 *Ocupação do Solo* (Land Cover)

O conceito de ocupação do solo, ao contrário do anterior, representa a ocupação física de uma determinada área, ou seja, não está dependente das atividades humanas, ainda que, anteriormente tenha sido referido que os dois conceitos possam estar associados. Embora se assuma uma relação entre ambos os conceitos, estes acabam por apresentar singularidades que permitem realizar a sua distinção. Enquanto o uso do solo envolve a intervenção por parte das atividades ligadas ao Homem para fins económicos, incluindo a usufruto e consequente alteração do território, a ocupação do solo não envolve qualquer tipo de atividade ligada ao ser humano (Nandaia, 2020).

A ocupação do solo está inteiramente ligada aos elementos presentes na superfície terrestre, com o propósito de os descrever. São considerados como elementos de ocupação a vegetação, as rochas, os edifícios, solo descoberto, corpos de água, e outros ligados a algo concreto e material (Donnay, *et al.* 2000). O conceito está também direcionado aos contextos biofísicos da superfície, que se altera consoante as modificações introduzidas pelas atividades humanas, embora, a sua definição ultrapasse as decisões implementadas pelo Homem (Nandaia, 2020).

Existe uma certa condicionante introduzida pelo desenvolvimento das comunidades, sendo ainda assim, possível afirmar que a ocupação do solo, identifica todas as características físicas da terra, desde rios, florestas, montanhas, oceanos, que sofrem efeitos das ações humanas, através das explorações efetuadas por via do uso do solo. Todavia, a ocupação do solo ultrapassa todas as consequências das atividades humanas e faz alusão apenas às particularidades dos elementos do meio natural (Nandaia, 2020).

2.1.2 Algoritmos de classificação supervisionada

No domínio da classificação de imagens de satélite, são numerosas as investigações científicas com o objetivo de evidenciar as mais variadas aplicações de técnicas de aprendizagem automática. Estes estudos exploram e implementam ferramentas direcionadas para a classificação de imagens, acompanhadas das avaliações de eficácia e desempenho, utilizando os mais variados algoritmos. São identificadas as vantagens e desvantagens aquando da realização da análise dos resultados, promovendo a identificação das abordagens e metodologias mais eficazes, assim como na compreensão do papel desempenhado pelo algoritmo selecionado.

Na área da classificação de imagens, o estudo da aprendizagem automática ou *machine learning*, que auxilia na deteção de padrões, tem vindo a emergir para aplicação em diferentes metodologias. Esta técnica de classificação é uma abordagem, que permite a aprendizagem do funcionamento interno de um sistema, a partir de um elevado número de dados introduzidos sem qualquer conhecimento antecipado sobre o mesmo. Graças à alta capacidade de modelar a mais variada informação de entrada, nos sistemas complexos, estes acabaram por assumir uma grande popularidade.

Na fase de classificação torna-se imperativo avaliar, de maneira cuidadosa, a seleção do algoritmo mais adequado ao exercício, acabando por se considerar como uma tarefa de elevado grau de complexidade, mediante do vasto número de algoritmos de classificação disponíveis, assim como dos resultados pretendidos. Para os diferentes resultados, as abordagens metodológicas e as características especificas dos dados, são também extremamente relevantes para a seleção do algoritmo e desta forma é recomendado realizar uma vasta análise dos múltiplos algoritmos de classificação.

Os classificadores abordados podem tratar-se de classificadores supervisionados e não supervisionados. Para os classificadores supervisionados são realizadas etapas sequenciais, enquanto a classificação não supervisionada é exercida de uma forma inversa. Considerando as tarefas sequenciais da classificação supervisionada, são utilizados grupos de dados de treino, com informações espectrais referentes aos objetos ou características das imagens, que posteriormente, o algoritmo irá, em função das particularidades, agregar pixéis com informações semelhantes de maneira a fazer a classificação de imagens de satélite. No caso da classificação não supervisionada, não são necessários dados de treino como entrada, mas requer parâmetros pré-definidos, como por exemplo, o número de classes pretendidas e o

algoritmo em função das características da imagem, agrupa os pixéis com valores espectrais mais próximos nessas classes.

Neste trabalho será utilizado um algoritmo de classificação supervisionado, portanto, alguns aspetos devem ser considerados. Durante o treino do modelo de classificação, é importante ter em atenção o conjunto de dados de treino, sendo necessário considerar a sua qualidade e a quantidade. De modo geral, quanto maior a quantidade e a qualidade dos dados de treino, melhor será o resultado da classificação, da mesma forma que o número de classes da nomenclatura que se pretendem considerar, tendo também influencia no final da classificação.

Nos subcapítulos seguintes serão descritos de forma breve, alguns dos algoritmos de classificação de imagens de satélite mais conhecidos aplicados em diversas investigações, descrevendo-os e completando com uma análise das suas potencialidades e desvantagens.

2.1.2.1 *K-Nearest Neighbor* (kNN)

O algoritmo kNN refere-se a um método de classificação de objetos, fundamentado em exemplos de dados de treino com características mais próximas, é considerado como um tipo de aprendizagem de baixo esforço, sendo que a função de classificação é caracterizada por apenas um cálculo de proximidade de particularidades, ou seja, utiliza os valores próximos como referência de calculo (Mohamed, 2017).

A técnica de classificação kNN é considerada de simples utilização quando o conhecimento da distribuição dos dados é muito baixo, permitindo à regra a atribuição da classe em função dos valores dos vizinhos mais próximos do valor do dado de treino. Quanto ao valor de k = 1, o algoritmo tem maior facilidade em conceder a classe, todavia, se o valor do dado for desconhecido, é considerado o valor a partir dos dados dos vizinhos mais próximos (Imandoust, *et al* 2013). Esta atribuição requer o cálculo da distância do dado desconhecido para cada um dos valores dos dados de treino mais próximos. Para melhor ajuste do classificador, muitas vezes é aconselhável a ponderação dos valores dos vizinhos mais próximos, como decisores para jurisdição do valor da classe.

A seleção do valor k, assim como a métrica de distância calculada entre valores são os principais determinantes no desempenho do algoritmo kNN. A determinação do raio da região local é calculada pela distância do elemento de ordem k, até ao elemento da classe

desconhecida, deste modo diferentes valores de k levam a várias distribuições de probabilidade condicional de classe. O tamanho do valor k é uma questão-chave que se deve ter em consideração para o desempenho geral da classificação kNN (Imandoust, *et al.* 2013). Quando o valor de k é baixo, a avaliação do local tende a ser mais pobre uma vez que existe escassez de dados e pontos ruidosos, indefinidos ou classificados de forma incorreta. Em contrapartida, quando o valor de k é elevado, a estimativa e atribuição da classe é facilitada com a supressão dos efeitos de ruído, mas o desempenho de classificação entre os elementos restantes torna-se menos distintiva.

Aquando da atribuição do valor da classe, é importante avaliar as distâncias espectrais, assim como o número de pixéis dos dados de treino utilizados, exigindo uma elevada capacidade de processamento por parte do algoritmo. Quando conjuntos de imagens de satélite incorporam grande número de bandas espectrais, tornam a tarefa do algoritmo mais dificultosa, portanto, não se considera adequado utilizar imagens de satélites adquiridas por sensores hiperespectrais (Imandoust, *et al.* 2013).

2.1.2.2 *Random Forest* (RF)

Considerado como um dos classificadores com maior exatidão na produção de resultados, após a sua execução em diversas áreas de estudo, o algoritmo RF combina várias árvores de decisão pela seleção aleatória de subconjuntos de dados de treino para a construção de diversas árvores singulares (Verdiguier, *et al.* 2020).

Este classificador torna-se não só adequado como atrativo, por não ser influenciável pelo ruído, mas também por se basear na classificação de múltiplas classes e em regressões. Desenvolve-se de forma relativamente rápida, evitando problemas de execução, e acaba por ser aplicado a problemas que tratam um grande número de dados. No seu funcionamento normal, auxilia de uma técnica de agregação de amostras (*bagging* - *bootstrap aggregating*), permitindo que diferentes árvores analisem a mesma amostra, enquanto outros dados podem nem sequer ser selecionados (Breiman, 2001).

Em cada subconjunto de dados são selecionados os melhores valores de *split* (medida do grau de impureza) e desta forma vão sendo construídas múltiplas árvores. A seleção do seu *output* final resulta do cálculo médio dos diferentes *outputs* individuais ou do voto maioritário resultante dos diferentes nós das ramificações das árvores produzidas (Mohamed, 2017).

A implementação do algoritmo RF, em classificações de imagens de satélite, tornou-se popular, pela sua particularidade de auxiliar na escolha dos dados das bandas de satélite, pela importância dos mesmos. Para além disso, o seu rápido processamento, a capacidade em lidar com dados pesados, funcionamento estável e tolerância ao ruído permitem que obtenha melhores valores de assertividade (Belgiu & Drăguţ, 2016).

Na área da deteção remota, a aplicação do algoritmo é favorável, pela sua versatilidade na classificação das características da superfície terrestre, na utilização de dados de diferentes periodicidades e características hiperespectrais. Caracteriza-se como um classificador tolerante aos fenómenos de sobreposições, mas acaba por ser sensível aos dados de treino e métodos de amostragem. Mede a importância das variáveis incorporadas na classificação para a compreensão dos dados na discriminação das classes, oferecendo a possibilidade de redução do esforço computacional, pela seleção das bandas de maior interesse, mantendo sempre valores de exatidão elevados (Belgiu & Drăguţ, 2016).

2.1.2.3 *Support Vector Machine* (SVM)

O algoritmo de classificação supervisionada SVM, requer de dados de treino como entrada para o funcionamento do seu exercício. Este método necessita destes conjuntos de dados de treino para que o algoritmo desenvolva um hiperplano ótimo (separação decisiva que minimize classificações incorretas) que realize a separação dos conjuntos das amostras num número de classes desconhecido de uma forma consistente. O processo iterativo do algoritmo pretende alcançar o limite de decisão para separar os dados de treino num conjunto de grandes dimensões, e de seguida desagregar os dados de simulação nas mesmas dimensões. De forma genérica, este classificador é caracterizado por ser um algoritmo binário linear, que atribui os dados de treino a uma classe de duas hipóteses possíveis, no entanto, no caso de uma possível classificação de imagem de satélite multiespectral, os dados das suas variáveis, sobrepõem-se no espaço acabando por dificultar a sua separabilidade linear com exatidão (Mountrakis, *et al.* 2011).

Conhecido pelas técnicas de formação de aprendizagem supervisionada, este algoritmo engloba elevadas capacidades de aplicação de classificações, regressão e seleção de características. Na separação linear, o conceito de margem, ou seja, a distância entre o hiperplano e os pontos mais próximos desse limite de ambos os lados, são importantes maximizar para atingir uma melhor generalização (Mohamed, 2017).

Há alguns limites que regem a relação entre o desempenho do modelo e a sua capacidade, o que pode ser utilizado para um equilíbrio entre dois ou mais elementos que são, mutuamente, exclusivos ou difíceis de otimizar em simultâneo (*trade-off*), entre o desvio do modelo e a variância do modelo (Mohamed, 2017).

Na área da deteção remota, o SVM tem vindo a ganhar alguma popularidade devido à capacidade de lidar com pequenos conjuntos de dados de treino, com produção de classificações de elevada exatidão ao contrário da utilização de outros métodos tradicionais. Para além disso, o algoritmo é conhecido por seguir pequenos riscos estruturais, minimizando o erro de classificação de dados (Belgiu & Drăguţ, 2016).

O SVM dispõe de vantagens ligadas à boa operabilidade com dados de alta dimensão, tendo capacidade de lidar com dados contínuos como dados categóricos, onde o compromisso entre a complexidade do modelo e o erro acabam por ser facilmente controlados. A captação das relações não lineares dos dados, é outro aspeto positivo do algoritmo, bem como a exata previsão que se manifesta extremamente elevada, desencadeando um ótimo desempenho de generalização (Mohamed, 2017). Oferece ainda uma solução única relacionada com o problema de otimização, que é convexo (tem um valor mínimo único), neste caso a sua robustez ao ruído torna-o num algoritmo capaz de lidar com dados que contenham erros. A dificuldade de interpretação de alguns conjuntos de dados e a falta de transparência nos resultados, por ser um método não paramétrico, são as maiores desvantagens do algoritmo SVM (Mohamed, 2017).

2.1.2.4 *Neural Networks* (NNs)

Caracterizado como um modelo matemático, o algoritmo NNs funciona como um sistema nervoso biológico, que tem por base três regras, nomeadamente a multiplicação, soma e ativação. Inicialmente, o algoritmo pondera todos os valores dos dados de treino colocados no *input*, realizando a multiplicação de cada valor da amostra por um peso específico. De seguida, para todas as entradas ponderadas, é adicionado um termo de enviesamento. Por fim, a soma de todas as entradas ponderadas e o termo de enviesamento são transformados, por uma função de ativação, para chegar ao cálculo final de saída (Mohamed, 2017).

Estas redes consistem num conjunto de nós (ou neurónios) interligados, que realizam o processamento da informação em paralelo com a capacidade de serem implementadas em

modelações de relações complexas entre dados de treino e resultados, encontrando padrões nos dados. A interligação entre os nós da rede, pode ser feita de duas formas distintas, podendo ocorrer apenas num sentido (*feed-forward*) ou em forma de ciclos por retropropagação, ou seja, o algoritmo é eficiente para ajustar os pesos da rede, permitindo um ajuste dos dados de entrada para as saídas desejadas. A rede neuronal é um sistema de fácil ajuste, no que diz respeito à modificação da sua estrutura, com base na informação presente durante a fase de treino, onde aprendizagem do algoritmo se faz pelo acerto dos pesos com objetivo de minimizar a disparidade entre a ativação do nó de saída e o resultado (Kavzoglu, *et al.* 2003; Silva, 2020).

No campo da deteção remota, as NNs alcançaram alguma popularidade devido à sua adaptabilidade e capacidade de produzir classificações de alta exatidão em comparação com outros algoritmos paramétricos. Nas tarefas de reconhecimento de imagem, os valores de entrada são os pixéis, sendo que se observa uma vantagem na capacidade de lidar com uma grande quantidade de dados com a finalidade de reduzir a sua dimensão, aliviando o processamento computacional. As NNs provaram ser de excelente execução na extração de características das imagens de satélite, produzindo classificação, segmentação semântica e deteção de diferentes características (Kavzoglu, *et al.* 2003; Silva, 2020).

Este algoritmo pode ser utilizado na resolução de problemas direcionados à programação linear, e também, não linear, assim como para a geração dos dados, onde não é necessário qualquer conhecimento antecipado. O algoritmo é, também, inteligente na aprendizagem dos dados de treino tornando-os poderosos e flexíveis, isto é, a rede neuronal não necessita de ser reprogramada. Graças às suas características, tem vindo a tornar-se num algoritmo de sucesso na resolução de muitos problemas de classificação, agrupamento e regressão (Mohamed, 2017).

Em contrapartida, o NNs não deve ser visto como um algoritmo geral de resolução de problemas, uma vez que outras técnicas podem manter-se em pé de igualdade. Existe uma falta de diretrizes claras para a arquitetura da rede artificial ótima, pois o processo envolve tentativas e erros, assim como o sucesso deste algoritmo depende do peso e qualidade dos dados (Mohamed, 2017).

2.1.2.5 *Naive Bayes* (NB)

O algoritmo NB consiste num modelo estrutural de um grafo (objetos relacionados num determinado conjunto) no qual são apresentados os nós como atributos e arcos que representam as dependências dos atributos. As dependências dos atributos são calculadas através de probabilidades condicionais de cada nó do dado de treino. Por sua vez, a suposição da probabilidade condicional neste método, nem sempre é verdadeira, o que prejudica o seu desempenho em aplicações com dependências de dados muito complexos. O classificador deduz, que a probabilidade de um nó pertence a uma determinada classe, porque tem por base a compreensão de que todos os atributos são independentes uns dos outros. Na prática, a maioria das combinações dos valores dos atributos não se encontram nos dados de treino, nem são apresentados num número consideravelmente suficiente, ou seja, a variedade de probabilidades não é confiável. O NB contorna esta situação pela sua suposta independência condicional e apesar disso, é considerado como um classificador realmente competente nas mais variadas aplicações (Rish, 2001).

Este classificador tem-se mostrado como um dos algoritmos de aprendizagem automática mais eficientes e eficazes em trabalhos de classificação, apesar do problema das suposições incorretas. Em alguns episódios, o algoritmo mostra-se incapaz de prever ou realizar a classificação correta em exercícios de grande simplicidade e por isso têm-se feito ajustes e alterações no modelo. De facto, a tentativa de melhorar a rede do NB é uma tarefa difícil, devido ao complicado entendimento da sua estrutura, bem como restrições que exigem uma alta complexidade computacional (Rish, 2001).

2.1.2.6 *Logistic Regression* (LR)

O modelo LR apresenta grande utilidade na modelação e previsão de resultados de dois níveis (evento de interesse e evento), através da introdução das mais distintas variáveis. É extremamente útil perante a necessidade de prever a presença ou ausência de uma determinada característica ou resultado, com base nos valores do conjunto de variáveis preditivas (Maroco, 2007; Bergonse, *et al.* 2010). É apropriado para modelos que envolvam tomadas de decisão, bem como no auxílio para a realização de previsões num alargado conjunto de estudos (LaValley, 2008).

O cálculo dos coeficientes a partir dos dados originais é mais complexo, tornando o cálculo manual impossível de ser executado. No modelo LR estão envolvidas equações que não podem ser calculadas de forma explicita, no entanto é possível resolver através de processos interativos, expressos em forma computacional e pela disponibilização de pacotes de *software* (LaValley, 2008). É de especial importância, no processo de modelação de um conjunto de dados para determinar se existe interação ou problemas de confusão. O termo de confusão de variáveis descreve que esta está associada à variável dependente de interesse, bem como a uma variável independente primária. De certa forma, situações desta natureza devem ser incluídas no modelo, independentemente do peso que possa trazer para o resultado (Cheng, *et al.* 2006).

Em deteção remota a LR é utilizada para flexibilizar os resultados da classificação, sendo particularmente adequada para fenómenos de presença e ausência, com o objetivo de prever distribuições espaciais de organismos, ou de características do solo. O algoritmo em questão necessita de dados de treino, que são importantes para previsões, características de classificações supervisionadas. Todavia, existem soluções para automatizar processos, sem a utilização de dados de treino, o que só acontece ao aceder a imagens binárias, obtidas a partir de uma classificação com o grupo de informações para duas classes. A LR permite a previsão da probabilidade de presença da classe de um fenómeno e em contexto de classificação, podendo ser estabelecido um limiar crítico que define a separação entre presença e ausência de uma classe com o grau e rigor pretendido (Cheng, *et al.* 2006; Lee, 2005).

2.1.2.7 *Decision Tree* (DT)

O algoritmo DT tem a finalidade de organizar grupos homogéneos, tendo por base um conjunto de dados de treino que obedecem a regras de decisão, e advêm de testes aplicados a uma ou mais variáveis dependentes, atendendo à sua hierarquização. Considerado como um procedimento analítico, que atenta a classes conhecidas, retiradas dos dados de treino, este algoritmo organiza a sua estrutura com base nos valores de refletância de cada banda constituinte da imagem de satélite (Pal & Mather, 2003).

Constitui-se, do topo para a base, por um nó inicial que engloba todos os dados de treino, e em cada ramificação o nó escolhido é definido pelo atributo que melhor se enquadra, ou seja, o valor que pode trazer maiores ganhos de informação. O conjunto de dados de treino inicial vai sofrendo divisões por subconjuntos, correspondendo cada um aos diferentes valores

desse atributo num processo repetitivo, até que os nós dos resultados de cada subconjunto pertençam a uma classe definitiva (Xu, *et al.* 2005; Friedl, *et al.* 1997).

Numa classificação de DT, os dados de treino percorrem o caminho desde a raiz até às folhas, de maneira a que cada percurso, desde o dado de treino (*input*) até à classe definitiva, são aplicadas regras de classificação. Os nós internos das árvores fazem referência aos diferentes atributos, os ramos aos possíveis valores e as folhas (nós finais) às classes possíveis de atribuição (Xu, *et al.* 2005; Friedl, *et al.* 1997).

A caracterização das medidas de repartição dos dados de entrada do nó inicial até ao resultado das classes categóricas, é considerada pela medida da diferença dada por uma função baseada nas proporções das classes, entre o nó corrente e os nós descendentes. São valorizadas a pureza e a entropia (medida da aleatoriedade de uma variável) do dado de treino, bem como a medida da diferença dada por uma função, baseada nas proporções das classes entre os nós descendentes, onde se valoriza a disparidade entre as repartições. Por fim, é também valorizada a medida de independência, ou seja, do grau de associação entre os atributos e a classe (Han, *et al.* 2022).

Este é considerado como vantajoso, uma vez que permite fases de treino mais rápidas e com um menor esforço de amostras, sem causar a redução de exatidão temática. Este método não paramétrico traz benefícios na construção do seu modelo para qualquer função, desde que no *input* sejam colocados dados suficientes sem que a eles seja assumida alguma distribuição particular (Pal & Mather, 2003).

As transformações monótonas das variáveis não alteram de qualquer forma a estrutura da árvore, ou seja, a sua composição é independente da escala das variáveis. O elevado grau de interoperabilidade é importante em decisões complexas, sendo considerado como um algoritmo robusto à presença de pontos e atributos redundantes ou irrelevantes. Por outro lado, o algoritmo acarreta uma certa instabilidade causada por pequenas perturbações que provocam alterações nos modelos. A presença de valores desconhecidos e a fragmentação da informação (replicação de subárvores) são outras desvantagens que apresentam problemas no classificador. É importante acrescentar que assume um esforço computacional elevado, assim como pode sobrepor dados facilmente, o que também se torna extremamente negativo, mas na prática existem algumas ferramentas para evitar tal limitação. Por fim, este algoritmo acaba por ser mais apropriado para classificações do que para estimativas de regressão (Mohamed, 2017; Han, *et al.* 2022).

2.1.3 Índices Radiométricos

Um índice radiométrico, é conhecido como a composição de uma banda complementar, executado a partir de bandas espectrais que compõem imagens de satélite, de maneira a evidenciar informação adicional (Prasad, *et al.* 2022). Os índices derivados dos dados de imagens de satélite, tornaram-se numa das principais fontes de informação para a monitorização das características da imagem, assim como das suas alterações ao longo do tempo (Teillet, *et al.* 1997).

Este é baseado em algoritmos simples e eficazes para diferentes avaliações de análise, e desenvolvimento de determinadas características da superfície. Até ao momento, não existe uma expressão matemática única que detenha a capacidade de definir todos os índices radiométricos, devido à complexa combinação de diferentes bandas de imagens multiespectrais, ao vasto conjunto de diferentes satélites, e à influência das resoluções espectrais (Prasad, *et al.* 2022).

A aplicação dos índices radiométricos é geralmente adaptada consoante os requisitos de utilização, em conjunto com as ferramentas e metodologias de validação apropriadas no solo. Segundo os índices selecionados, é possível evidenciar as características da superfície terrestre, como por exemplo na identificação de água, vegetação e edifícios. Estas composições podem ser utilizadas como bandas adicionais na classificação de imagens, assim como informação adicional para filtrar os diferentes dados de treino. Todavia, antes de realizar qualquer cálculo de um índice radiométrico, os valores brutos do pixel, ou seja, os DN devem ser calibrados, a partir de correções radiométricas. Qualquer cálculo de um índice representa dados, tecnicamente corretos, embora dependa das correções das imagens, sendo produzidos resultados diferentes para as mesmas condições da superfície terrestre (Gu, *et al.* 2011).

Geralmente, as abordagens de aplicação de índices radiométricos em diferentes imagens são importantes para otimização de erros e instabilidades, como forma de ultrapassar problemas de valores espectrais e geometrias de polígonos de informações disponibilizados pela IGV. Os cálculos de índices acabam por impactar na análise da variação de classes, pois a partir das observações dos valores espectrais de índices é possível excluir regiões indesejadas dos dados de treino, auxiliando na discriminação das diferentes características da superfície e solidificação de classes. Portanto, o processo de filtragem com aplicação de diferentes índices

radiométricos coopera de maneira positiva para a melhorar as métricas de exatidão temática dos mapas de uso e ocupação do solo desenvolvidos.

2.1.3.1 NDVI (*Normalized Difference Vegetation Index*)

A utilização de instrumentos espectrais de alta resolução tem vindo a possibilitar o aumento do número de bandas recolhidas, pelas imagens de satélite, já a largura dessas mesmas bandas tem vindo a ficar cada vez mais pequena (Xue, & Su, 2017; Honkavaara, *et al.* 2013).

O Índice de Diferença Normalizada da Vegetação (NDVI – *Normalized Difference Vegetation Index*) quantifica valores da diferença entre a radiação do infravermelho próximo e a radiação de luz vermelha do espectro eletromagnético. Este é um dos índices mais usados e implementados a partir de informação multiespectral, caracterizando de forma clara o crescimento e o vigor das plantas (Xue & Su, 2017; Karnieli, *et al.* 2010). A partir das imagens de satélite é possível interpretar as diferenças e alterações das folhas e o estado de saúde da planta. O cálculo do NDVI, é expresso a partir da fórmula (1), onde o RED corresponde à banda do vermelho e o NIR à banda do infravermelho próximo.

O valor do índice varia entre -1 e 1, tendo uma sensibilidade à vegetação, independentemente do seu tamanho. Quando se está sob uma vegetação forte e em bom estado de saúde, esta absorve grandes quantidades de comprimentos de onda do visível (vermelho), refletindo, em compensação, grandes porções de radiação infravermelha próxima (Xue & Su, 2017).

Por outro lado, quando a vegetação se encontra em baixas condições de estado de saúde, a luz da banda do vermelho é mais refletida e a radiação do infravermelho o oposto. O NDVI tem valores mais altos quando se está sob a presença de uma vegetação mais saudável e densa, e em situações contrárias, com valores mais baixos e, portanto, este índice é usado muitas vezes, na monitorização e deteção de vegetação (Luo, *et al.* 2019; Albarakat & Lakshmi, 2019). Para o cálculo do mesmo são usadas as bandas das imagens Sentinel-2, sendo elas a banda 8 (NIR) e a banda 4 (RED) representadas na fórmula (2).

Na deteção remota, a aplicação deste método desencadeou grande sucesso na monitorização de culturas, podendo ser utilizado ao longo de toda a estação da produção agrícola e para avaliação do seu rendimento anual, desenvolvimento de pastos, indicação de estados de seca, atividade de fotossíntese das plantas, águas superficiais, quantidade de biomassa e outros determinantes. Todavia, pode encontrar problemas relacionados com a sensibilidade aos

efeitos do brilho e da cor do solo, dos efeitos da atmosfera, nuvens e até da sombra, como as sombras das copas das árvores, que acaba por necessitar de uma calibração do sensor do satélite (Xue & Su, 2017).

Tabela I – Fórmulas de cálculo para o índice NDVI e utilização das bandas.

Índice radiométrico	Fórmula	Fórmula (Bandas Sentinel-2)
NDVI (*Normalized Difference Vegetation Index*)	$\text{NDVI} = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}}$ (1)	$\text{NDVI} = \frac{\text{b8} - \text{b4}}{\text{b8} + \text{b4}}$ (2)

2.1.3.2 NDWI (*Normalized Difference Water Index*)

As imagens de satélite têm sido um fator de grande importância na avaliação de recursos hídricos e monitorização das áreas costeiras, onde são submetidas a aplicações que envolvem a delimitação de águas superficiais, através da execução de técnicas para a extração de informações temáticas. Existem várias formas de extrair informação sobre a água utilizando o cálculo de bandas, e o Índice de Diferença Normalizada da Água (NDWI - *Normalized Difference Water Index*) obtêm resultados eficientes (Xu, 2005).

Este índice foi projetado com função de maximizar a refletância da água, através de comprimentos de onda verde, bem como minimizar a baixa refletância da banda do infravermelho pelos recursos hídricos e aproveitar a alta refletância derivada da vegetação e características do solo no comprimento de onda do infravermelho. Como resultados do índice, as características dos corpos de água apresentam valores positivos, sendo aprimoradas, e em contrapartida a vegetação e o solo são assinalados, regularmente, por valores zero ou negativos e, portanto, acabam por ser suprimidos (McFeeters, 1996; Xu, 2005).

O NDWI tem a capacidade de delinear os corpos de água abertos nas imagens de satélite, avaliar a sua turvação e medir de forma eficaz o teor de humidade. É calculado através da combinação das bandas do verde e do infravermelho próximo (respetivamente GREEN e NIR), permitindo realçar e destacar as pequenas alterações no teor da água de todos os corpos superficiais. O cálculo do índice é realizado a partir da fórmula (3).

A sua utilização é adequada à deteção e monitorização de ligeiras alterações no teor das águas, contudo é muito sensível a infraestruturas, podendo originar problemas de confusão. (Xu, 2005; Gao, 1996). O NDWI é um índice mais robusto e, portanto, é menos sensível aos efeitos atmosféricos que o NDVI (Gao, 1996). Os valores deste índice, tal como o NDVI, variam entre -1 e 1, os valores positivos são respetivos aos corpos de água, enquanto os valores negativos ou zero representam outras caraterísticas da imagem. Situações de seca rígida, seca moderada, inundações e estado de saúde da vegetação são outras monitorizações para as quais o NDWI é considerado útil e importante (McFeeters, 1996)

No processamento de imagens Sentinel-2, o NDWI é calculado através da aplicação das bandas 3 (GREEN) e banda 8 (NIR), como se encontra constatado na fórmula (4).

Tabela II – Fórmulas de cálculo para o índice NDWI e utilização das bandas.

Índice radiométrico	Fórmula	Fórmula (Bandas Sentinel-2)
NDWI (*Normalized Difference Water Index*)	$\text{NDWI} = \frac{\text{GREEN} - \text{NIR}}{\text{GREEN} + \text{NIR}}$ (3)	$\text{NDWI} = \frac{\text{b3} - \text{b8}}{\text{b3} + \text{b8}}$ (4)

2.1.3.3 NDBI *(Normalized Difference Built-up Index)*

O Índice de Diferença Normalizada de Construções (NDBI – *Normalized Difference Built-up Index*) é o mais apropriado para realizar a quantificação das construções de áreas urbanas. Tem a capacidade de evidenciar de forma positiva este tipo de características da superfície terrestre, sendo importante para controlar e monitorizar a expansão dos vários ambientes citadinos. A expansão da área construída e edificado é uma realidade muito comum em qualquer cidade, revelando que a importância do NDBI tem vindo a aumentar progressivamente (He, *et al.* 2010; Jacquin, *et al.* 2008)

O índice quantifica as bandas espectrais com a radiação infravermelha de onda curta (SWIR) e a radiação infravermelha próxima (NIR), calculada pela fórmula (5). Os valores do NDBI variam entre -1 e 1, onde a área edificada aumenta com os valores positivos do índice, enquanto os valores negativos ou equivalentes a 0 representam outras características da imagem (Karanam, 2018). Este é, também, relevante na extração de outras caraterísticas da

superfície terrestre, como por exemplo a vegetação quando o valor de NDBI é inferior a 0, e redes hidrográficas com valores de NDBI entre 0 e 0.1 (He, *et al.* 2010). Em relação ao espectro eletromagnético, as áreas edificadas influenciam numa maior quantidade de absorção de radiação de infravermelho próximo (NIR), em comparação com a banda do infravermelho de onda curta (SWIR) em que a radiação é menos absorvida. Posto isto, pode garantir-se que a refletância de NIR é, comparativamente, inferior à refletância da banda SWIR, sendo estes níveis de refletância muito superiores (Jesus, 2020).

Existem, de facto, alguns fatores adjacentes, como a monitorização de áreas de solo descoberto identificadas como se fossem construções ou edificado, o que pode acarretar inconvenientes quando se realiza a análise do crescimento de áreas urbanas. O principal ponto negativo na aplicação do índice assenta, deste modo, na grande taxa de confusão espectral entre as caraterísticas de solo descoberto e a construção artificial (Karanam, 2018).

Para a execução do cálculo do índice NDBI através de imagens Sentinel-2, é necessário utilizar a banda 11 e a banda 8, como consta na fórmula (6).

Tabela III – Fórmulas de cálculo para o índice NDBI e utilização das bandas.

Índice radiométrico	Fórmula	Fórmula (Bandas Sentinel-2)
NDBI (*Normalized Difference Built-up Index*)	$\text{NDBI} = \frac{\text{SWIR} - \text{NIR}}{\text{SWIR} + \text{NIR}}$ (5)	$\text{NDBI} = \frac{\text{b11} - \text{b8}}{\text{b11} + \text{b8}}$ (6)

2.1.3.4 MSAVI *(Modified Soil Adjusted Vegetation Index)*

Uma outra tipologia que tem vindo a gerar interesse por parte da comunidade científica refere-se à caracterização de padrões temporais e espaciais da vegetação, a partir de imagens de satélite, de modo a analisar o uso e ocupação do solo da superfície da terra. Para tal, incentivou-se o desenvolvimento aprofundado de modelos e índices radiométricos, para ampliar a sensibilidade das características vegetativas, relativamente a fatores de efeitos da atmosfera e do solo, sendo exemplo disso, a introdução do índice SAVI.

Na investigação de Qi, *et al.* (1994), os autores exploraram o ajuste do fator L do SAVI para atenuação dos efeitos do solo, e apresentaram propostas de melhorias, para aumentar as dinâmicas da refletância emitida pela vegetação e minimizar o ruído causado pelo solo. O seu foco seria o maior realce dado à vegetação, consoante os efeitos adversos das características

do solo, sem que esta informação fosse colocada em causa. Com base no SAVI, nasceu o Índice de Vegetação Ajustado ao solo Modificado (MSAVI – *Modified Soil Adjusted Vegetation Index*), um indicador que detém maior sensibilidade à vegetação e efeitos adversos (Qi, *et al.* 1994).

O MSAVI, versão modificada do anterior SAVI, não se distingue apenas pela maior sensibilidade das características, mas também pela substituição do fator L, fazendo uma melhoria no SAVI, passando a ter um processo de ajuste automático para valores ótimos. (Qi, *et al.* 1994). Tem capacidade de eliminar a influência de determinadas particularidades da superfície para determinação da vegetação. Os valores do MSAVI variam entre -1 e 1 com a fórmula (7), com a utilização das bandas do infravermelho próximo (NIR) e do vermelho (RED), respetivamente banda 8 e banda 4 das imagens Sentinel-2 (8). Este índice é utilizado em situações onde outros índices de vegetação não funcionam, e quando se está perante valores de MSAVI baixos, estes dizem respeito a áreas de solo descoberto, enquanto valores mais altos retratam vegetação densa, capaz de cobrir o solo.

Tabela IV – Fórmulas de cálculo para o índice MSAVI e utilização das bandas.

Índice radiométrico	Fórmula
MSAVI (Modified *Soil Adjusted Vegetation Index*)	$\text{MSAVI} = \frac{(2 * \text{NIR} + 1 - \sqrt{(2 * \text{NIR} + 1)^2 - 8 * (\text{NIR} - \text{RED})})}{2}$ (7)
	Fórmula (Bandas Sentinel-2)
	$\text{MSAVI} = \frac{(2 * \text{b8} + 1 - \sqrt{(2 * \text{b8} + 1)^2 - 8 * (\text{b8} - \text{b4})})}{2}$ (8)

2.1.3.5 NBR *(Normalized Burn Ratio)*

Existem determinados pontos que se têm tornado uma prioridade na investigação, bem como a identificação do estado de saúde da vegetação e de áreas recentemente queimadas, mediante dos incêndios florestais. Estas envolvem uma análise de escalas espaciais e temporais de forma a detetar perturbação nos ecossistemas, na degradação de solos e na perda da biomassa. Para explicar estas consequências introduziu-se um índice radiométrico,

capaz de estabelecer a relação entre a vegetação saudável e as regiões que sofreram causas da deflagrarão do fogo, a Taxa de Queima Normalizada (NBR – *Normalized Burn Ratio*).

Idealmente, é utilizado para destacar regiões, nas imagens de satélite, com ocorrência de incêndios, quantificando a relação das respostas espetrais da radiação do infravermelho próximo (NIR) e a radiação do infravermelho de onda curta (SWIR), expressa pela fórmula (9). Os resultados do NBR apontam para áreas ardidas recentemente, com baixa refletância no comprimento de onda do NIR e grandes quantidades de refletância por parte do SWIR, permitindo verificar as diferenças entre as respostas espetrais da vegetação saudável e as regiões queimadas (Keeley, 2009). Com base no índice correspondente, é possível obter informação adicional, que tem por base a avaliação da gravidade do incêndio, conseguida a partir do cálculo do NBR da imagem antes do incêndio e subtraindo-a pelo NBR calculado após o incêndio. Este é dado pela diferença temporal dNBR (Roy, *et al.* 2006).

Para as bandas das imagens Sentinel-2, o índice é calculado através da utilização da banda 8 (NIR) e banda12 (SWIR), como se apresenta na fórmula (10). Os valores de NBR mais altos indicam que se está sob a presença de vegetação saudável, enquanto valores mais baixos aludem para áreas ardidas. Considerado para detetar áreas após incêndios, mesmo que estes não sejam frequentes numa determinada região, o índice pode tornar-se importante na distinção entre vegetação e áreas de solo descoberto.

Tabela V – Fórmulas de cálculo para o índice NBR e utilização das bandas.

Índice radiométrico	Fórmula	Fórmula (Bandas Sentinel-2)
NBR (*Normalized Burn Ratio*)	$\mathrm{NBR} = \frac{\mathrm{NIR} - \mathrm{SWIR}}{\mathrm{NIR} + \mathrm{SWIR}}$ (9)	$\mathrm{NBR} = \frac{\mathrm{b8} - \mathrm{b12}}{\mathrm{b8} + \mathrm{b12}}$ (10)

2.1.4 Produtos de Mapas

A criação de mapas de uso e ocupação do solo tem vindo a ganhar extrema relevância no ordenamento do território, possibilitando a representação da superfície terrestre correspondente à realidade, bem como monitorização das atividades implementadas pelo Homem e ainda deteção da distribuição espacial de diferentes fenómenos ocorrentes no

espaço territorial. Os mapas caracterizam-se como uma ferramenta básica nos estudos ambientais, auxiliando nas tomadas de decisão e implementação de medidas para o desenvolvimento de planos de ordenamento e planeamento do território e ainda na definição de políticas de administração de recursos naturais (Carrão, 2002; Mota, *et al.* 2012).

No contexto da sua utilização ou finalidade, devem ser tomadas metodologias distintas de elaboração. Com o crescimento de produção de mapas de uso e ocupação do solo por diferentes entidades, cada agente particular realiza o seu produto em função daquilo que pretende ser considerado e representado pelos mapas temáticos, com definição do número e representatividade das classes para fazer corresponder o que se encontra à superfície na sua representação em mapa (Mota, *et al.* 2012).

Nos processos de produção de mapas temáticos, é necessária uma fase inicial para definição de nomenclatura e do tópico de representação sobre a informação da superfície. Esta definição deve ter em consideração a função e os objetivos da informação temática produzida, as características técnicas dos objetos, ponderando por exemplo a *minimum mapping unit* (MMU), os métodos e técnicas de produção e, deve ainda, respeitar critérios de coerência e clareza. Uma nomenclatura deve ter uma definição taxonomicamente correta das classes informativas, assim como o entendimento das regras de generalização para que a informação representada corresponda com a realidade pretendida.

Existem várias nomenclaturas de uso e ocupação do solo a nível nacional e internacional, sendo as mais conhecias, os dados *CORINE Land Cover* (CLC), Carta de Ocupação do Solo (COS), *Urban Atlas* (UA), *United States Geological Survey* (USGS), *GlobeLand30*, *Dynamic World* e *Environmental System Research Institute* (Esri). De facto, todas estas merecem destaque, pois na verdade, estes projetos revelam características próprias e representações temáticas do solo distintas tendo, contudo, como fundamento único o estabelecimento e promoção de informação sobre classes geográficas com possível aplicação nos diferentes territórios.

A caracterização das classes de uso e ocupação do solo das nomenclaturas recorrem a técnicas de segmentação, que são apenas nomeadas após a definição do nível de segmentação que melhor identifica os objetos nas imagens. O nível de desagregação de cada nomenclatura é diretamente proporcional ao valor de rigidez e rigor das diferentes classes e, deste modo, quantos mais níveis de desagregação implementados, maior será o número de classes da nomenclatura. Todavia, a tabela seguinte resume as principais características das nomenclaturas das cartografias temáticas globais mais icónicas.

Tabela VI – Exemplos de mapas temáticos com respetivas nomenclaturas e características.

Nome	Sensor	Formato	Resolução	Nomenclatura	Extensão	Produtor	Metodologia	Atualização	Referência bibliográfica
CORINE Land Cover (CLC)	Landsat-5 - MSS/TM, Landsat-7 – ETM+, SPOT 4 e 5, IRS P6 LISS III, RapidEye, Sentinel-2 e Landsat-8	Produtos vetoriais e produtos *raster*	10 m	LC com Nível 1: 5 classes; Nível 2: 15 classes; Nível 3: 44 classes	União Europeia (EU)	EEA	Interpretação visual para a identificação de objetos, envolvendo processos de classificação automática e uma revisão manual efetuada por equipas nacionais de cada país envolvido.	A cada 5 anos	(Büttner, *et al.* 2021)
Carta de Ocupação do Solo (COS)	DMC129 e UltraCam Falcon	Produtos vetoriais	0.25 m	LULC com Nível 1: 9 classes; Nível 2: 20 classes; Nível 3: 46 Classes; Nível 4: 81 classes	Portugal Continental	DGT	Fotointerpretação de imagens aéreas ortorretificadas de grande resolução com quatro bandas espetrais (azul, verde, vermelho e infravermelho próximo).	Indefinida	(Caetano & Marcelino, 2019)
Urban Atlas (UA)	Pleiades, KOMPSAT, Planet, SPOT 6, SuperView, Sentinel-2 etc.	Produtos vetoriais	10 m	LULC com Nível 1: 5 classes gerais; Nível 2, Nível 3 e Nível 4: contam ao todo 27 classes mais detalhadas	União Europeia (EU), Países Balcãs Ocidentais, Reino Unido e Turquia	EU *represented by the European Commission - Directorate-General for Defence Industry and Space*	Aplicação de classificações automáticas para distinção de classes genéricas para maior detalhe com uso de dados de referência como o OSM, COTS e mapas topográficos. Classificação manual para maiores níveis pormenorização, com interpretação visual de dados EO.	A cada 6 anos	(European Commission, 2020)
United States Geological Survey (USGS)	Satélites Landsat	*Raster*	80 m	LULC com Nível 1: 9 classes generalistas; Nível 2: 24 classes representativas; Nível 3 e 4: classes muito detalhadas	Estados Unidos da América (USA)	*US Geological Survey Circular*	Classificação realizada a partir de técnicas de fotointerpretação convencional a partir de fotografias infravermelhas de grandes altitudes.	Anual	(Anderson, *et al.* 1976)
GlobeLand30	Landsat 7 - ETM+, Landsat 8 - OLI, Gaefon 1, Huan Jing 1	Produtos vetoriais e produtos *raster*	30 m	LC com 10 classes	Global	Centro Nacional de Geomática da China	Classificação supervisionada através da aplicação do algoritmo RF, com diferentes passos desde a criação dos dados de treino obtidos a partir de grupos de pessoas espalhadas pelo mundo inteiro, classificação de imagens, verificação dos dados, validação e controlo da qualidade dos resultados.	Decenal	(Chen, *et al.* 2015)

Environmental System Research Institute (*Esri*)	Sentinel-2 L2A	*Raster*	10 m	LULC com 9 classes	Global	*Esri*, *Microsoft*, *Impact Observatory*	Classificação assistida, realizada a partir de modelo de AI do *Impact Observatory*, com utilização de conjuntos de dados de treino com bilhões de pixéis classificados pela comunidade da *National Geographic Society*.	Anual	(Esri, 2022)
Dynamic World	Sentinel-2 L1C	*Raster*	10 m	LULC com 9 classes	Global	*Google*, *World Resources Institute*	Classificação supervisionada com aplicação do algoritmo CNN, através de dados de treino da *Dynamic World* da *National Geographic Society*.	2-5 Dias	(Brown, *et al.* 2022)

2.2 Estado de Arte

Atualmente, a atividade humana tem vindo a desenvolver-se de forma evidente, impactando na alteração da ocupação do solo e nas modificações da superfície, com consequências nos diferentes ecossistemas e na modelação da paisagem. A velocidade das alterações causadas, em parte, pelo crescimento populacional e consequente aumento das atividades económicas, originam transformações em curtos espaços de tempo e, portanto, cresce a necessidade de desenvolver mapas de uso e ocupação do solo capazes de acompanhar este avanço repentino. Para entrar nesta corrida sem fim, a produção de mapas atualizados tem vindo a ganhar cada vez mais importância.

No seio da deteção remota, os dados de imagens satélites, demonstram forte relevância na produção de mapas LULC, graças à alta resolução temporal, espetral e espacial, permitindo a produção gradual e automática de mapas. O acesso a informação, por exemplo IGV, capaz de caracterizar e individualizar em classes a informação da superfície, facilitam o acesso a dados de treino para a conceção de mapas, tornando-se importante saber aproveitá-los.

No presente capítulo serão fundamentadas, as principais metodologias e técnicas que envolvem a filtragem de dados de treino, utilizados para a posterior classificação de imagens de satélite, com recurso à identificação e descrição de investigações científicas que suportam, de maneira conceptual e técnica, o tema do trabalho. Como em qualquer trabalho, as referências bibliográficas revelam elevado grau de importância na recolha de informações, conceitos, metodologias e aprendizagens desenvolvidos pelos mais variados investigadores. Na presente dissertação não será exceção e, portanto, na aquisição de competências ao longo deste percurso, destacam-se os estudos aprofundados de Patriarca, *et al.* (2019), Fonte, *et al.* (2020), Radoux, *et al.* (2014) Schultz, *et al.* (2017).

Fonte, *et al.* (2020) desenvolveu uma metodologia automática, para a criação de dados de treino a partir de informação OSM, e classificação de imagens de Sentinel-2 através de uma nomenclatura de classes LULC. No artigo, os dados em bruto, retirados do OSM, passaram por processos de conversão para classes de uso e ocupação do solo, através da aplicação do programa OSM2LULC (Patriarca, *et al.* 2019). Seguidamente à conversão, e aplicação de processos de filtragem dos dados OSM, foi realizada a classificação da série temporal de

imagens de satélite, em 8 classes LULC, pelo algoritmo RF, a partir da definição de regras para os índices radiométricos calculados.

O exercício foi aplicado para duas áreas de estudo selecionadas de forma rigorosa. A área de estudo A, situada na região oeste de Portugal Continental junto ao estuário do rio Tejo, correspondendo para a Área Metropolitana de Lisboa (NUTS III), enquadrada num ambiente principalmente urbano. Com uma área de 1560 km² incluindo a cidade de Lisboa e aglomerados urbanos periféricos, sendo grande parte caracterizada como área urbana, todavia, vegetação natural, florestas, etc. também se incluíam (Fonte, *et al.* 2020). E a área de estudo B, correspondente à região da Serra da Estrela, localizada no interior de Portugal Continental (NUTS III). Esta ocupava uma área de 2000 km², incluindo o parque natural da Serra da Estrela e pequenos aglomerados urbanos distanciados e de pequenas dimensões. (Fonte, *et al.* 2020).

A seleção das áreas de estudo realizou-se em função da sua representatividade, isto é, retratam-se duas tipologias de áreas com desigualdades associadas, uma predominantemente urbana e outra rural, sendo que também restringem diferentes cobertura dos dados OSM. Lisboa apresentava um problema de falta de informação nas periferias da cidade, enquanto na segunda área, a cobertura era quase inexistente e os poucos dados disponíveis, faziam apenas referência ao Parque Natural da Serra da Estrela. É ainda, importante referir que para cada área de estudo foi utilizada uma sequência temporal composta por três imagens Sentinel-2 com objetivo de captar a variação sazonal da vegetação ao longo dos meses de março, junho e outubro (Fonte, *et al.* 2020).

A metodologia aplicada no artigo mencionado identifica uma sucessão passos de:

i. Conversão dos dados brutos retirados do OSM para classes CLC através do *software* OSM2LULC_4T (versão adaptada);
ii. Harmonização das classes temáticas da nomenclatura CLC no nível de desagregação número 2 e nível 3;
iii. Produção de três conjuntos de dados de treino resultados do segundo passo através da aplicação sucessiva de processos de filtragem, auxiliada de índices radiométricos (TD0, TD1 e TD2);
iv. Seleção de amostras de treino de cada conjunto dos dados de treino (TS0, TS1 e TS2);
v. Avaliação da separabilidade entre as classes e avaliação da exatidão dos dados de treino e das amostras retiradas dos mesmos;

vi. Classificação das imagens Sentinel-2 com as amostras dos dados de treino criadas no passo 4;
vii. Classificação geral das imagens Sentinel-2 utilizando todos os conjuntos de dados de treino, e respetivas amostras através do classificador RF;
viii. Criação de diferentes classificações através dos dados brutos do OSM e os conjuntos de amostras de treino (TS0, TS1 e TS2);
ix. Avaliação da exatidão dos resultados da classificação para os mapas produzidos.

A partir do processo de conversão e de filtragem dos dados recolhidos do OSM foram produzidos três conjuntos de dados de treino diferentes, denominados por TD0, TD1 e TD2. Para o TD0, foi realizada a simples conversão dos dados OSM para classes LULC com grupos de células mistas. Para o TD1, foram criadas grelhas de células vetoriais com as mesmas dimensões dos pixéis que compunham as imagens Sentinel-2, com as seguidas interseções entre as grelhas e as classes produzidas pelo *software* OSM2LULC_4T, bem como o cálculo da percentagem de ocupação da classe no interior de cada célula, com o objetivo de identificar os pixéis mistos (células com mais do que uma classe no seu interior). Todas as células mistas dos dados de treino TD0 foram removidas de forma a produzir os dados de treino TD1. O TD2 resultou da aplicação de três índices radiométricos (NDVI, NDWI e NDBI) para filtrar os dados de treino e estabelecer valores limites para excluir ou considerar de forma rigorosa informações das classes (Fonte, *et al.* 2020).

As amostras TS1, TS2 e TS3 retiradas do conjunto dos dados de treino (TD0, TD1 e TD2 respetivamente) foram criadas devido a restrições computacionais e de forma a atenuar o desequilíbrio quantitativo de classes nos diferentes conjuntos de dados. O tamanho de dados das amostras era proporcional à área de cada classe da COS de 2015. Caracterizada como uma classificação pixel a pixel para as diferentes imagens, teve-se em consideração a MMU de 1 hectare, portanto, foi ainda aplicado um filtro universal, com o intuito de remover a maioria das células isoladas e de pequena dimensão. (Fonte, *et al.* 2020).

Para avaliação da exatidão temática dos resultados dos mapas produzidos, utilizou-se a COS de 2018 como referência, assim como as classificações realizadas com os três grupos de dados de treino (TD0, TD1 e TD2). As duas áreas de estudo apresentaram erros de classificação, a partir da aplicação do TD0 e TD1, quanto ao TD2, tal não se sucedeu, assumindo-se que à medida que eram aplicados os dados de treino de TD0 para TD1 e de TD1 para TD2 a exatidão da classificação ia aumentando, concluindo que, de facto, os processos de filtragem

a que os dados foram submetidos conduziriam a resultados de exatidão global superior (Fonte, *et al.* 2020).

Radoux, *et al.* (2014), também ressaltou um interesse nos mesmos campos, conhecimentos abordados, metodologia adotada e resultados obtidos. A sua investigação consistiu num processo de criação sucessiva de mapas de exatidão temática para vários períodos anuais, com o objetivo de investigar técnicas para a classificação automática de imagens óticas. Foi realizada uma classificação através da criação de dados de treino locais a partir da seleção automática de dados sobre a ocupação do solo, mas desatualizados. Por sua vez, a extração da informação dos dados foi baseada nas características espaciais e auxiliada na definição de dados para treino, através da aplicação de critérios espetrais. O seu objetivo visava aplicar classificadores supervisionados (SVM e GML) a partir de informações de mapas já existente. O estudo desenvolveu diferentes métodos para a minimização de problemas que dificultam a utilização de mapas de ocupação do solo como referência. Os principais pontos do trabalho, subdividiam-se na:

i. Seleção adaptada dos dados de treino, através de um processo automático de extração das assinaturas espetrais para cada classe de ocupação do solo presente na informação temática.
ii. Aplicação de filtros e seleção dos dados de treino com o intuito de reduzir o impacto dos conjuntos de pixéis mal classificados, no sentido de realizar classificações exatas. Seriam avaliados métodos para a localização das células de referência e para a distribuição das assinaturas espetrais.

Estes métodos foram aplicados para duas vastas áreas, sendo que a primeira, englobava grande parte da Europa Central e Ocidental e a segunda incluía uma grande parte da bacia do rio Amazonas. As imagens utilizadas, foram descarregadas do satélite MERIS com uma resolução de 300 m num período anual entre 2008 e de 2012. As imagens foram submetidas a pré-processamentos de correção radiométrica e atmosférica, eliminação da cobertura de nuvens, e ajustes na precisão da navegação. Para os mapas, importantes para recolha dos dados de treino, definiram o mapa do GLC2000 para a área de estudo da bacia do Amazonas e para a europa a informação temática CLC (Radoux, *et al.* 2014).

A metodologia utilizada ao longo do artigo, focou-se na extração de dados de treino a partir da informação dos mapas temáticos, para a posterior classificação global da ocupação do solo. De forma sintética, as etapas subdividiram-se, primeiramente na otimização local, através da estratificação das duas áreas de estudo, como forma de dividir os dados e as

características da superfície. Em seguida, realizaram-se duas fases de filtragem dos dados que se dividiram em aplicação de filtros espaciais e espetrais. Os filtros espaciais procederam à exclusão de células dos mapas temáticos com classificações incorretas de acordo com a localização e através do *software* MBRF como forma de analisar e manter os pixéis com um número considerável de vizinhos mais próximos da mesma classe, podendo este ser considerado como *No Data* caso não atingisse o número necessário. Para os filtros espetrais, foi realizada uma análise da distribuição das informações espetrais numa determinada classe de ocupação do solo e em função desses valores era feita a eliminação das células com valores anormais. Os *outliers* foram eliminados por meio de testes qui-quadrados na distância Mahalanobis[1] entre a instância e a distribuição do modelo (Radoux, *et al.* 2014).

Para validação dos resultados foi utilizada uma amostra aleatória de pontos independentes dos mapas temáticos. Na área de estudo do rio Amazonas, o conjunto de dados de validação surgiu no âmbito da iniciativa *GlobCover*, concebida a partir de fotointerpretação com base em imagens de alta resolução, cálculo anual de NDVI e conhecimento da área. Na europa, o conjunto de dados surgiu de uma amostra aleatória de pontos interpretados visualmente pelos próprios autores utilizando o serviço *web "Bing Map"*. Para cada área, realizou-se o cálculo do valor de exatidão global, com o objetivo de identificar o grupo de pixéis corretamente classificados, bem como medidas de exatidão ajustadas, que toleram erros com pouco impacto na representação da superfície (Radoux, *et al.* 2014).

As classificações obtidas registaram-se, comparativamente, superiores aos outros produtos globais. Os resultados revelaram-se promissores e para a área da Europa, a exatidão global do algoritmo GML, acabou por ser significativamente melhor do que a do classificador SVM. A estratégia de filtragem dos dados a partir do MBRF teve um excelente desempenho, assim como a sua combinação com filtros espectrais, resultando numa melhoria da classificação e consequente exatidão temática superior. Para a segunda área de estudo, a exatidão global do classificador SVM foi inferior à do GML, mas a diferença acabou por ser insignificante. Contudo, os métodos implementados, que abarcaram os processos de filtragem espacial e espetral para definição dos dados de treino, atingiram valores de exatidões globais elevadas (Radoux, *et al.* 2014).

[1] Distância Mahalanobis: Recebeu este nome em homenagem ao matemático indiano Prasanta Chandra Mahalanobis e define-se como uma medida amplamente utilizada no reconhecimento de padrões. É uma medida entre dois pontos de dados no espaço definido por recursos relevantes. Considera variações desiguais, bem como correlações entre recursos e, posteriormente, avalia de forma adequada a distância atribuindo pesos ou fatores de importância aos recursos dos pontos de dados (Xiang, *et al.* 2008).

Em Schultz, *et al.* (2017), a ideia principal do estudo dos autores nasceu para a produção de mapas de uso e ocupação do solo (OLC) para uma área restrita no sul da Alemanha, cidade Heidelberg, com base nas informações e dados recolhidos do OSM. O tema desta investigação ressaltou também um certo interesse dado aos conhecimentos desenvolvidos. Estes promoveram a exploração da monitorização da superfície terrestre, com o recurso a imagens Landsat e a sua classificação auxiliada da nomenclatura CLC no seu nível 2, após o processo de harmonização de *tags* OSM. Os desenvolvedores do artigo centraram os seus principais objetivos para encontrar respostas às questões (Schultz, *et al.* 2017):

i. Que *tags* e relações em OSM podem ser utilizados para criar classes LULC a partir de CLC?
ii. Pode um mapa de ocupação *open source* ter uma cobertura completa mesmo que se tenha informações incompletas nos dados IGV?
iii. Quão exato é este produto de ocupação do solo, e como é que se compara com produtos existente?

Para alcançar respostas às perguntas colocadas, os investigadores adotaram uma abordagem metodológica, que passou por três principais fluxos de trabalho. Num primeiro momento, determinaram os elementos das informações OSM para a definida área de estudo, a partir da extração das *tags* do respetivo projeto IGV e harmonização para classes CLC. Este processo serviu para que, em etapas da metodologia mais avançadas, o processo de validação fosse possível com a comparação dos mapas produzidos e informações de outras cartografias. Uma vez que os dados OSM sofrem de problemas ligados à sobreposição de elementos, este *handicap* foi resolvido com a atribuição de uma regra de prioridade aos polígonos mais pequenos relativamente aos de dimensões superiores (Schultz, *et al.* 2017).

Num segundo momento da abordagem utilizada, como toda a área de estudo não foi completamente caracterizada utilizando apenas as informações OSM, as imagens de satélite Landsat foram introduzidas para a classificação das áreas em falta. Contudo antes de avançarem para a fase de classificação, as imagens Landsat no seu nível L1TP submeteram-se a um processo de correção dos efeitos atmosféricos, auxiliada do *software* de criação de máscaras de nuvens Fmask. Ainda num momento de preparação de dados, os polígonos harmonizados da informação OSM para a nomenclatura CLC, passaram por uma conversão para *raster*, com uma resolução de 30 metros, a mesma das bandas lansat de forma a obter a uniformidade dos dados. Ainda nesta segunda fase, os investigadores definiram o algoritmo RF, utilizando 500 árvores, para preencher a carência de informação da área de estudo. Este

classificador foi selecionado, devido à sua alta capacidade para etapas de treino e classificação, receber uma grande quantidade de dados, tendo em vista que todos os dados OSM convertidos foram introduzidos (Schultz, *et al.* 2017).

O terceiro e último ponto da metodologia direcionou-se para a avaliação e comparação da exatidão temática do mapa OLC produzido com outra cartografia. A estratégia utilizada pelos autores descreveu uma abordagem de validação através da criação de pontos amostra, distribuídos utilizando o método aleatório estratificado por classe, para o mapa OLC e o mapa CLC 2012. As informações atribuídas a cada ponto de referência foram extraídas de dados de imagens de alta resolução, de forma a definir o valor real da ocupação do solo. A validação foi completada com o cálculo de medidas de validação complementares. Estas resumem-se à criação de tabelas de contingência e o cálculo de exatidão global, exatidão do produtor e utilizador, erro de omissão e de comissão (Schultz, *et al.* 2017).

No que concerne aos resultados, os desenvolvedores alcançaram metas de um mapa OLC com uma superioridade de 6% de exatidão temática à cartografia CLC 2012. Relativamente às classes, o mapa OLC deteve de uma melhor individualização para a maioria das informações temáticas, no entanto a classe de corpos de água foi a única inconsistência encontrada comparativamente com o mapa CLC (Schultz, *et al.* 2017). A concordância temática entre os dois mapas foi de 84%, contudo perceberam que o CLC carateriza grandes áreas homogéneas para determinadas classes, enquanto as classes do mapa OLC foram espacialmente heterogéneas e segmentadas. A equipa concluiu que para a área de estudo selecionada, foi permitida a conversão de *tags* OSM para um nível detalhado de classes CLC com uma área de cobertura de 91,8%, no entanto para regiões de carência de informação OSM, isto pode vir a ser uma lacuna (Schultz, *et al.* 2017).

O artigo mostrou um eficiente proveito dos dados OSM para o treino de um algoritmo (RF), e classificação de imagens Landsat. Entenderam que onde a ocupação do solo era perfeitamente identificada pelos dados, estas informações podiam ser utilizadas para o treino do algorimo, para classificar e compensar as regiões sem dados. O contributo teórico deste artigo interliga-se diretamente com o proveito de informação OSM para classificação de imagens de satélite. Ainda assim, a possibilidade de realizar a combinação dos dados IGV para uma nomenclatura de mapas globais, e posterior geração de mapas de ocupação do solo de detalhe, é também reconhecido como um contributo material.

3 Dados

Os dados que suportam uma investigação permitem determinar a maturidade da mesma, sendo que estes representam a estrutura central das quais são retirados resultados e conclusões, dando credibilidade e consistência a um trabalho. A integridade e a seleção cuidadosa dos respetivos dados, são cruciais na construção de uma narrativa fiável, e é evidente que para qualquer trabalho investigativo, se visa o cumprimento de objetivos, através da definição dos melhores dados disponíveis. Todavia, em domínios como a deteção remota, muitos dados com qualidade superior implicam investimento para a sua aquisição. Neste sentido, para a presente dissertação, o exercício foca-se essencialmente em dados provenientes de fontes gratuitas. Embora a partir de um conjunto de dados mais restrito, é crucial determinar os que melhor se adequam às necessidades, com um olhar voltado para o alcance de resultados pretendidos. Na generalidade dos projetos, cuja abordagem central remete para a classificação de imagens e criação de mapas de ocupação ou uso do solo, como no presente estudo, a perspetiva principal está na definição eficaz dos dados de imagens de satélite, na sua utilização conjunta com outras fontes de dados da mesma natureza e no conjunto de dados de treino utilizados para classificar as respetivas imagens.

Neste capítulo, explora-se um vasto conjunto de dados provenientes de imagens de satélite, bem como repositórios de dados ligados à produção de IGV, imprescindíveis para a produção dos dados de treino, e ainda a definição da área de estudo para o desenvolvimento investigativo. Numa primeira fase, a presente secção 3, está subdividida pelo subcapítulo 3.1, no qual é feita uma análise minuciosa das fontes de imagens de satélite e dos programas mais conhecidos e de maior contributo para a área de exploração de conhecimento direcionado à classificação de imagens. São facultados os objetivos, as vantagens e desvantagens dos satélites e respetivos produtos, de forma a apresentar informação sucinta para a posterior seleção. É neste subcapítulo 3.1, que se consolidam ainda, duas secções para a descrição detalhada das características das imagens de satélite selecionadas.

Em seguida, o subcapítulo 3.2 apresenta um esclarecimento, de modo sucinto e breve, do conceito de modelos digitais de elevação e posteriormente, uma discriminação dos diferentes produtos disponíveis. Espera-se com a definição do MDE que este possa contribuir de forma decisora na atribuição de informação adicional, em auxílio da identificação de diferentes áreas de elevações.

Adicionalmente ao respetivo capítulo, importa também ter presente uma secção que se destine ao conjunto dos dados de treino. No subcapítulo 3.3, é apresentado o conceito de IGV, a qual tem vindo a adquirir uma relevância significativa na classificação de imagens e nas suas aplicações. A secção é essencialmente dedicada ao OSM, uma comunidade de pessoas que se dedica à produção e partilha de informação geográfica de acesso gratuito e ao alcance de qualquer um, e explora como estes dados geográficos podem ser aproveitados para produção de dados de treino.

Finalmente, para a última secção do respetivo capítulo, esta destina-se à definição da área de estudo, utilizada para todo o desenvolvimento experimental e investigativo. Esta secção 3.4 garante a descrição e o enquadramento da respetiva região para classificação, tendo por base os objetivos definidos inicialmente, direcionados tanto ao nível do desenvolvimento metodológico, bem como no principal propósito de produção de informação de ocupação do solo para a área em questão.

3.1 Imagens óticas multiespectrais

Os sistemas de imagens de satélite têm a capacidade de obter e medir a energia eletromagnética refletida ou emitida pelos objetos que se encontram à superfície da terra, em diferentes comprimentos de onda. Estes comprimentos de onda incluem a luz visível, infravermelho, micro-ondas e ultravioleta, sendo a principal fonte de energia o sol. Cada uma das características que se encontra à superfície acaba por refletir a radiação emitida pelo sol em diferentes comprimentos de onda, auxiliando na identificação dos objetos, como árvores, edifícios, corpos de água, entre outros (Ouchra & Belangour, 2021; Borra, *et al.* 2019).

Atualmente, existe uma grande quantidade de imagens de satélite disponíveis em diferentes plataformas de observação da superfície terrestre. Graças ao crescente número de sensores abarcados nos vários satélites, a frequência de aquisição de imagens está a aumentar, gradualmente. As imagens capturadas pelos sensores são armazenadas em formato digital e, de seguida, submetidas a processos computacionais para eliminação de interferências (Amro, *et al.* 2011). Estas imagens de satélite podem ser categorizadas por três tipos diferentes: pancromáticas, multiespectrais e hiperespectrais. Concentramos o nosso interesse, exclusivamente, nas imagens multiespectrais, uma vez que serão as únicas utilizadas para fins de análise e estudo. As imagens multiespectrais, correspondem a um conjunto de imagens adquiridas por vários sensores operando as faixas dos comprimentos de onda mais pequenos

ou contínuos. São imagens compostas por um número de 3 a 8 bandas espectrais, que possibilitam a combinação entre elas para a produção de imagens com diferentes cores (Ouchra & Belangour, 2021; Borra, *et al.* 2019).

O aproveito de imagens multiespectrais tem vindo a ganhar importância nas mais variadas aplicações. O processo de classificação de imagem tem vindo a desempenhar um papel fundamental na análise das informações à superfície, assim como na produção de mapas de uso e ocupação do solo (Amro, *et al.* 2011). Missões espaciais de grande relevância, como os programas Landsat e Copernicus, que incorporam os notáveis satélites Landsat-8 e Sentinel-2, consolidaram-se como os produtores de imagens de satélite preferenciais tanto na Europa como no resto do mundo. Todavia, no âmbito dos principais contribuidores de dados de imagens de satélite, o conjunto de satélites Sentinel merece destaque, abrangendo o Sentinel-1 até ao Sentinel-6, assim como os satélites do programa Landsat, com ênfase para o Landsat-7, Landsat-8 e, mais recentemente, o Landsat-9. Ademais, os satélites Pleiades, PRISMA e SPOT 6-7 também desempenham um papel significativo na aquisição de imagens de satélite.

É relevante compreender que cada satélite foi lançado com propósitos específicos, incorporando na sua plataforma um sensor com características únicas, o qual proporciona vantagens, mas também inconvenientes, que devem ser tidos em consideração. Com o intuito de fornecer ao leitor um melhor entendimento, a Tabela VII apresenta os diferentes satélites, as respetivas missões e descrição de características, visando auxiliar à seleção do satélite e respetivas imagens mais adequadas para o desenvolvimento da presente dissertação.

Tabela VII – Caracterização de alguns exemplos de satélites em missões, respetivas vantagens e desvantagens.

Missão	Satélite	Ano de Lançamento	Sensor	Objetivo	Vantagens	Desvantagens	Referência bibliográfica
Copernicus	Sentinel-1	2014	C-SAR	Dar continuidade aos dados ERS/ENVISAT para monitorização da superfície terrestre e oceano. Permite obter imagens em todas as condições atmosféricas e a qualquer hora, seja dia ou noite. Oferece observações elementares para aplicação de operações nos oceanos, mares e zonas polares. Auxiliar em aplicações na superfície terrestre, com resposta a emergências de condições atmosféricas.	- A interação da radiação com a cobertura de nuvens é reduzida ou mesmo nula, devido aos elevados comprimentos de onda; - As imagens têm uma resolução espacial adequada e são obtidas sob quaisquer condições meteorológicas, dia ou de noite e independentemente das condições de iluminação; - As imagens adquiridas são de grande utilidade para a observação da superfície em áreas de grande cobertura de nuvens.	- O fluxo de trabalho para extrair informação das imagens é mais complexo do que no caso das imagens óticas multiespectrais; - A disponibilidade de imagens é limitada; - A resolução espacial, apesar de elevada, não consegue ainda atingir o pormenor das imagens óticas; - Recentemente Sentinel-1B, deixou de enviar dados a 23 de dezembro de 2021 após sofrer uma anomalia e as tentativas de correção do problema não tiveram sucesso.	(Potin, *et al.* 2019; ESA, s.d.; ESA, 2023a)
	Sentinel-2	2014	MSI	Oferece sucessão pormenorizada de serviços que dependem de observações óticas multiespectrais de resolução espacial elevada na superfície terrestre e zonas costeiras. Disponibiliza imagens para aplicação de monitorização das características da superfície, como por exemplo de áreas agrícolas, florestais ou para ecossistemas. Fornece dados para administrar emergências, como incêndios florestais, movimentos em massa e aplicações relacionadas com a segurança da população. Faculta informações óticas de alta resolução de para toda a superfície terrestre de maneira a dar continuidade aos dados disponibilizados pelas séries Landsat e SPOT.	- Resolução espacial e espetral com bandas no visível e infravermelho próximo de 10 metros, bandas do infravermelho próximo de 20 metros; - Imagens adquiridas sob condições restritas fixas, facilitando para o estudo da superfície terrestre; - Acesso a metadados garantindo transformações de valores e correções em casos necessários; - Cobertura da superfície a um nível global, disponibilizando dados para cada região do planeta; - Acesso gratuito às imagens e de grande disponibilidade; - As diferentes bandas espectrais do sensor MSI revelam elementos não visíveis ao olho humano.	- Limitações relativamente às condições atmosféricas; - O acesso aos dados deixa muito a desejar, uma vez que as imagens são imprecisas e de baixa qualidade de visualização; - O acesso aos ficheiros não é claro e gera dúvidas; - As imagens de alta resolução originam uma ocupação significativa na memória do computador, assim como o processamento dos dados provocam um elevado esforço computacional.	(Potin, *et al.* 2019; Deliry, *et al.* 2020; ESA, s.d.; ESA, 2012a)
Landsat	Landsat-7	1999	ETM+	Conceder continuidade aos dados das missões Landsat anteriores. Fornecer imagens com informações sobre águas, oceanos, neve, vegetação, balanço da radiação, albedo e refletância, tipos de nuvens, temperaturas e porção de nuvens, temperatura da superfície terrestre, imagens da espessura do gelo continental e marítimo, cor/biologia dos oceanos. Apoiar as comunidades governamentais, internacionais e comerciais. Disponibilização de dados de imagens totalmente gratuitos e de livre acesso.	- Banda pancromática com resolução espacial de 15 m; - Calibração radiométrica absoluta de 5%; - Infravermelho térmico com uma melhoria de resolução, quatro vezes superior em relação ao TM do Landsat-5; - Disponibilidade de imagens gratuitas desde o ano de 2008; - Armazena dados de períodos temporais muito anteriores que podem ser comparados e combinados com dados de outras missões, mais recentes.	- A precisão da correção atmosférica depende da qualidade dos dados de aerossol disponíveis para determinar o perfil atmosférico no momento de aquisição de imagens; - Necessidade de aplicação de algoritmos de correção para aumentar a resolução e qualidade de imagens recolhidas pelo satélite; - Falha que desencadeou imagens com faixas diagonais, afetando a qualidade e a utilização de alguns dados; - Resolução espectral limitada, com restrição na capacidade de análise das características da superfície.	(Ihlen, 2019a; Wijedasa, *et al.* 2012; ESA, 2012b)

	Landsat-8	2013	*Imaging Multispectral Radiometers* (OLI/TIRS)	Garantir dados suficientemente consistentes comparativamente aos dados das missões Landsat anteriores, em termos de geometria de aquisição, calibração, características da superfície, características espectrais e espaciais, qualidade do produto final e em relação à disponibilização das imagens para estudos ao nível da superfície terrestre e da sua alteração ao longo do tempo. Atualizar periodicamente um repositório global de imagens da superfície livres de nuvens.	- Permite captar comprimentos de onda de luz invisíveis ao olho humano; - Possui bandas espectrais de maior resolução radiométrica, melhorando a calibração e as características de sinal de ruído em comparação com os Landsat anteriores; - Reduziu a saturação das superfícies com altos valores de refletividade permitindo uma melhor avaliação das condições à superfície; - Prevê a cobertura de nuvens e evita a aquisição de dados irrelevantes.	- Limitações relativamente aos efeitos atmosféricos, posição do sol, ângulo de visão do sensor, inclinação da superfície e características presentes à superfície; - Objetos brilhantes, como por exemplo praias, edifícios e lagos de água salgada, podem ser confundidos como coberto de nuvens quando são aplicados algoritmos de eliminação de nuvens; - Bordas e franjas de nuvens tendem a ser opacas e podem ser perdidas por algoritmos de deteção de nuvens.	(Ihlen, 2019b; Li, *et al.* 2015; ESA, 2022a)
	Landsat-9	2021	*Imaging Multispectral Radiometers* (OLI-2/TIRS-2)	Garantir e dar continuidade ao papel do programa Landsat com dados suficientemente consistentes. Monitorizar e gerir os recursos naturais necessários à vida das populações, através do acompanhamento das alterações e crescentes alterações da ocupação global do solo, com consequências profundas nas alterações climáticas, ecossistemas, ciclos de carbono, gestão de recursos, economias nacionais e globais e saúde humana. Distribuição de produtos Landsat-9 aos utilizadores de forma imparcial e não discriminatória. Adquirir dados de imagens multiespectrais de resolução média de aproximadamente 30 metros.	- Permite uma melhor distinção de assinaturas espectrais em áreas sombrias, ambientes mais escuros como águas ou florestas densas; - Com uma resolução radiométrica aperfeiçoada, permite distinguir várias tonalidades em determinados comprimentos de onda; - Reduz significativamente a luz difusa, permitindo uma melhor correção atmosférica para avaliações de superfícies mais precisas.	- Várias anomalias e problemas ao nível do funcionamento interno podem ocorrer a qualquer momento; - Imperfeições ao nível do processo de varrimento e deteção da informação à superfície terrestre são algumas das anomalias que podem influenciar os dados; - Quando um objeto é mais brilhante do que o que sensor consegue suportar, pode desencadear-se o efeito de supersaturação fazendo com que o detetor pare de funcionar corretamente por alguns momentos; - Blocos de dados defeituosos e sem informações em algumas imagens (evento raro).	(Sayler, 2022; ESA, 2022b)
COSMO-SkyMed e Pleiades	Pleiades	2011	HiRI	Fornecer imagens multiespectrais de alta resolução auxiliada de uma geolocalização extremamente precisa. Oferece vários modos de aquisição de informação para atender a vários campos de aplicação da cartografia, como a agricultura, silvicultura, hidrologia, levantamento geológico, geologia dinâmica, gestão de risco e proteção para catástrofes naturais.	- Modelação com cobertura mínima de nuvens, oferecendo produtos com baixa interferência deste efeito atmosférico; - Composto por dois satélites gémeos, fornece capacidade de estereoscopia, essencial à criação de mapas 3D; - Capacidade de adquirir até três imagens para uma área assinalada na mesma órbita; - Possui uma das mais altas resoluções espaciais, com capacidade até 0.5 metros.	- Otimização obrigatória aquando se utilizam modelos para a criação de mapas 3D de elevada exatidão; - Sensibilidade à iluminação, com anomalias de reflexão no campo do espectro eletromagnético das micro-ondas; - Limitações aliadas às condições atmosféricas; - O acesso a dados pode ser dispendioso.	(Coeurdeley, *et al.* 2012; Collin, *et al.* 2018; ESA, 2012d)
SPOT	SPOT 6-7	2012	NAOMI	Operar numa órbita segmentada com os satélites Pleiades, causando uma combinação ideal de detalhes e cobertura, com visitas diárias em qualquer ponto do planeta. O desempenho foi desenvolvido com sucesso após a experiência realizada com o SPOT-5 fornecendo ortofotos com resolução espacial de 1.5 m de imagens de cores naturais. Dados de alta resolução aplicados para a criação e atualização de mapas de escalas 1: 25 000, modelos digitais do terreno com precisões de 5 a 10 metros, análise física e monitorização de construções de grandes dimensões, como aeródromos, barragens, etc., controlo do estado ecológico, análise agrícola e desenvolvimento de infraestruturas urbanas.	- Capacidade de revisita de elevada frequência; - Capacidade de estereoscopia, representando a superfície em 2D e 3D com precisão; - Ampla capacidade de cobertura global, importante para a gestão das previsões meteorológicas; - Resposta rápida às necessidades de informações pretendida; - Permite ajustar a sua órbita e os seus ângulos, possibilitando uma flexibilidade operacional e aquisição de imagens em regiões de interesse; - Possui uma alta resolução espacial até 1.5 metros; - Garante continuidade da série de satélites SPOT, oferencendo vários dados ao longo do tempo.	- Tamanho de memória de imagens grande e pesada; - As limitações de vida e o seu consequente desgaste ao longo do tempo podem resultar na diminuição da qualidade das imagens e eventual falha operacional; - Baixa qualidade de imagens influenciadas pelas condições atmosféricas.	(Coeurdeley, *et al.* 2013; ESA, 2023b)

PRISMA	PRISMA	2019	PAN/HYP	Observação dos recursos naturais para estudo das principais alterações ambientais, por exemplo interações entre a atmosfera, a biosfera e a hidrosfera, observação das transformações ambientais e climáticas e as consequências da atividade humana nos ecossistemas. Fornece informações importantes para o apoio à prevenção de riscos naturais e de origem humana, monitorização do património cultural, tomadas de medidas de ajuda para crises humanitárias, atividades agrícolas e exploração dos recursos minerais.	- Está equipado com o sensor hiperespectral mais poderoso criado até hoje; - Análise detalhada e discriminação pormenorizada de características nas imagens adquiridas; - As imagens recolhidas são ferramentas valiosas para investigadores em diversas áreas.	- O acesso a dados é restrito pela identidade oficial, uma vez que são apenas disponibilizados aos utilizadores com conta no *site* oficial e mesmo assim podem não ser divulgados se a identidade assim entender; - Depende das condições atmosféricas; - O satélite PRISMA apresenta restrições na cobertura da superfície terrestre, em virtude da sua natureza singular com um satélite individual e das exigências de observação que dependem exclusivamente da sua órbita e posicionamento.	(Loizzo, *et al.* 2018; Guarini, *et al.* 2018; ESA, 2012c)

Para a investigação, os dados de base para classificação de imagens serão os produtos disponibilizados pelo Sentinel-2, uma vez que são as imagens utilizadas no caso de estudo de Fonte, *et al.* (2020). Sendo que este estudo pretende melhorar a metodologia desenvolvida no respetivo trabalho, a seleção das mesmas imagens é de elevada relevância.

A alta resolução dos dados de imagem produzidos pelo Sentinel-2, com bandas que variam a resolução espacial de 10 a 60 m, implicam que as imagens tenham um enorme potencial para exploração pormenorizada da superfície terrestre. Além disso, a sua alta resolução temporal de 5 dias torna-se valiosa, apresentando uma melhoria de 10 para 5 dias a seguir ao lançamento do segundo satélite gémeo (Vuolo, *et al.* 2018; Phiri, *et al.* 2020). Graças às altas resoluções e ao conjunto das bandas do visível e do infravermelho próximo, Sentinel-2 oferece elevados contributos para a monitorização detalhada da ocupação do solo a uma escala muito pequena, assim como o desenvolvimento de áreas urbanas, áreas arbustivas e regiões florestadas. Por exemplo, no caso de estudo de Yang, *et al.* (2018), foram identificadas as inundações sazonais em áreas urbanas a partir da utilização de imagens Sentinel-2. Outro caso de estudo bem-sucedido, com utilização de imagens Sentinel-2, foi elaborado por Nomura & Mitchard, (2018), na monitorização de culturas florestais na região de Tanintharyi em Mianmar a uma escala muito especifica (Phiri, *et al.* 2020).

As imagens do satélite Sentinel-2, representam um amplo leque de oportunidades para a monitorização de características da superfície. No entanto, tal como os outros sensores passivos abarcados nos mais variados satélites, as limitações que envolvem os efeitos atmosféricos é uma realidade ainda presente. Problemas desta natureza estão ligados à aquisição de imagens com a manifestação de cobertura de nuvens e sombras, com interferência na qualidade das imagens. Nestas situações, a verdadeira informação dos objetos na superfície terrestre, encontra-se em discordância com a realidade levando a imperfeições na classificação.

Com o intuito de superar inconvenientes relacionados com as limitações dos efeitos atmosféricos, pretende-se utilizar, em conjunto com imagens Sentinel-2, a classificação automática de imagens adquiridas por outro sensor. Uma vez que se definiu como objetivo de estudo, aumentar as fontes de dados para classificação relativamente ao artigo Fonte, *et al.* (2020), tal procedimento auxilia na transposição para a definição de outro satélite de aquisição de imagens. Além disso, a partir da classificação em combinação de duas fontes de dados, é possível aumentar as séries temporais das imagens e consequentemente maior detalhe relativamente características de ocupação do solo.

Em consequência do aumento da disponibilidade de sensores (Tabela VII), instalados nos diferentes satélites, foi possível ampliar o leque de oportunidades para a monitorização de alta frequência de longo prazo. Neste sentido, concentrou-se a definição do segundo satélite na reconhecida compatibilidade entre as plataformas de Sentinel-2 e Landsat-8, dado que transportam sensores multiespectrais, fornecendo uma qualidade radiométrica aprimorada e o aumento das aplicações para a monitorização das características de ocupação do solo (Mandanici & Bitelli, 2016)

O satélite Landsat-8 foi lançado no ano de 2013 com o sensor OLI a bordo, com capacidade para fornecer imagens multiespectrais com resolução de 30 metros (15 metros banda pancromática) e uma resolução temporal de 16 dias. As imagens foram projetadas para oferecer várias possibilidades de aplicação em diferentes áreas da deteção remota, com capacidade de monitorizar objetos à superfície, calcular índices radiométricos e produzir informações relativas à temperatura (Mandanici & Bitelli, 2016; Roy, *et al.* 2014). A missão dos satélites Sentinel-2 é constituída por uma combinação de dois satélites (Sentinel-2A e Sentinel-2B), transportando sensores MSI, com capacidade de adquirir imagens compostas por 13 bandas de diferentes resoluções (10 m e 60m). Comparativamente ao satélite Landsat-8, instrumento multiespectral do Sentinel-2 abarca um campo de visão extremamente amplo, de 290 km de largura de faixa, significativamente superior aos 185 km do sensor OLI. A missão tem como objetivo dar continuidade às missões SPOT (Drusch, *et al.* 2012; Mandanici & Bitelli, 2016).

A utilização das imagens de ambos acarreta possível sucesso, dado à sua disponibilidade e acesso de aquisição de imagens de forma totalmente gratuita, sistemas de coordenadas geográficas idênticos e a interoperabilidade e capacidade de monitorizar grandes superfícies (Wang, *et al.* 2017). Contudo, a exploração combinada de diferentes satélites apresenta uma vasta série de desafios técnicos e conceptuais, uma vez que as configurações espaciais e espectrais das imagens diferem (Mandanici & Bitelli, 2016). Será necessário submeter os dados observados pelo Landsat-8, de 30 m, a um processo de reamostragem do pixel da imagem, de maneira a ajustá-lo à resolução espacial das imagens Sentinel-2 (10 m). Este processo desencadeia uma maior pormenorização dos dados Landsat-8, e muitos recursos que não são possíveis de ser observados claramente nas imagens originais, podem passar a ser identificados explicitamente numa escala mais reduzida (Wang, *et al.* 2017).

3.1.1 Imagens Sentinel-2

O desenvolvimento do Programa Copernicus pela ESA e pela EU tem vindo a contribuir de forma efetiva através da disponibilização de imagens multiespectrais, fornecendo dados de satélite de alta resolução para a monitorização da superfície, alterações climáticas e deteção de catástrofes (Phiri, *et al.* 2020).

O lançamento dos satélites do programa Copernicus impactou de forma revolucionária na evolução da observação da superfície, desempenhando um papel decisivo na produção de mapas. Desde o primeiro envio do satélite Sentinel nos anos de 2010 e, subsequente, implementação de uma missão constelações de dois satélites, registou-se um marco significativo no que diz respeito à aquisição de imagens de alta resolução (espacial, espectral e temporal) e consequente criação de mapas de ocupação do solo com exatidão (Phiri, *et al.* 2020; Drusch, *et al.* 2012).

Composto por várias constelações de satélites, o programa Copernicus, adequou os seus objetivos a iniciativas de monitorização da superfície terrestre, dos mares e oceanos, da vegetação, solos e regiões costeiras (Phiri, *et al.* 2020). A missão Sentinel-2, iniciou-se com o lançamento do seu primeiro satélite Sentinel-2A a 23 de junho de 2015, e o lançamento do segundo, Sentinel-2B realizou-se a 7 de março de 2017. Estes realizam uma órbita sincronizada com o sol, separados a 180º e a uma altitude de 786 km, com um campo de visão de 290 km (Phiri, *et al.* 2020).

A bordo dos satélites Sentinel-2, estão instalados instrumentos MSI, com a capacidade para captar imagens em 13 bandas distintas. Os seus sensores são caracterizados como passivos, ou seja, o seu funcionamento ocorre apenas durante o dia, quando a superfície é exposta à radiação solar (Drusch, *et al.* 2012). A resolução espacial varia entre quatro bandas com uma resolução de 10 m, seis bandas de 20 m e três restantes de 60 m. A resolução espectral compreende os 443 nm e os 2 190 nm,e engloba toda a região do espectro eletromagnético da parte visível, dos infravermelhos de onda curta e os infravermelhos de onda longa. Já a resolução temporal, ou seja, período de tempo que o satélite necessita para realizar o seu percurso orbital e passar novamente no mesmo ponto, é de 5 dias (Drusch, *et al.* 2012; Yang, *et al.* 2018; Phiri, *et al.* 2020).

Tabela VIII – Bandas multiespectrais do satélite Sentinel-2. Fonte: Phiri, *et al.* 2020, p. 7.

		Sentinel-2A		Sentinel-2B	
Resolução Espacial (m)	Bandas	Comprimento de onda central (nm)	Largura de banda (nm)	Comprimento de onda central (nm)	Largura de banda (nm)
10	Banda 2 - Blue	492.4	66	492.1	66
	Banda 3 - Green	559.8	36	559	36
	Banda 4 - Red	664.6	31	664.9	31
	Banda 8 - NIR	832.8	106	832.9	106
20	Banda 6 - Red edge	740.5	15	739.1	15
	Banda 7 - Red edge	782.8	20	779.7	20
	Banda 8A - Narrow NIR	864.7	21	864	22
	Banda 11 - SWIR	1613.7	91	1610.4	94
	Banda 12 - SWIR	2202.4	175	2185.7	185
60	Banda 1 - Coastal aerosol	442.7	21	442.2	21
	Banda 9 - Water vapour	945.1	20	943.2	21
	Banda 10 - SWIR-Cirrus	1373.5	31	1376.9	30

Os produtos Sentinel-2 passam por etapas de correção e processamento que estabelecem os diferentes níveis antes da sua disponibilização. Durante estes processos, as imagens adquiridas são submetidas a pré-processamentos exigentes de maneira a garantir a qualidade necessária. Os níveis das imagens são caracterizados por produtos de Nível-0, Nível-1A, Nível-1B, Nível-1C e Nível-2A. Os produtos de Nível-0 e de Nível-1A, não são disponíveis para os utilizadores, uma vez que são imagens sem qualquer pré-processamento aplicado, compactadas no formato de origem (Phiri, *et al.* 2020).

O Nível-1B, é um produto disponível para os utilizadores, e constitui dados de imagens com correções radiométricas para valores de radiância no TOA. Os produtos Nível-1C, resultam do acesso a modelos digitais de elevação para obter as coordenadas geográficas UTM/WGS 84 projetadas na imagem, com valores de refletância acima da atmosfera. Dentre todos os produtos oferecidos pela missão Sentinel-2, os mais comuns à utilização são os dados de Nível-1C, com informações de refletância TOA, bem como os produtos Nível-2A, com informações de refletância do fundo da atmosfera (Phiri, *et al.* 2020).

Contudo, é importante considerar que os produtos Sentinel-2 sofrem de inconsistências na aquisição de imagens, assim como em muitas situações, devido à frequência de cobertura de nuvens e sombras, questionando a qualidade e utilização das imagens em diferentes aplicações.

3.1.2 Imagens Landsat-8

O programa de satélites Landsat, com mais de 40 anos de execução, fornece um período de registo temporal mais longo da superfície terrestre. Desde o lançamento do seu primeiro satélite, Landsat-1 no ano de 1972, foram enviados de forma consecutiva vários satélites, os Landsat-2, 3, 4, 5 e 7, de maneira a adquirir e fornecer informação da superfície com o aumento gradual das resoluções espectrais e espacial das imagens (Roy, *et al.* 2014). O recorde registado pelos anos deste programa, estende-se pelo lançamento de sucesso do satélite Landsat-8, a 11 de fevereiro do ano de 2013, com uma colaboração conjunta da NASA e do USGS. O objetivo principal da missão de lançamento do Landsat-8, foi assegurar a continuidade das observações Landsat, a fim de promover a consistência dos dados e permitir comparações com imagens capturadas pelas séries de satélites anteriores. Ainda assim, a missão tem como objetivo, a continua aquisição de dados multiespectrais com resolução média, para caracterizar os objetos presentes à superfície (Roy, *et al.* 2014; Deliry, *et al.* 2021).

A bordo do satélite Landsat-8, encontram-se os dois sensores, OLI e TIRS, estabelecendo avanços técnicos que visam a melhoria do desempenho relativamente aos sensores dos Landsat precedentes. Ambos detêm de um sistema de varrimento, com planos alinhados por matrizes ao longo do seu percurso, proporcionando uma maior resolução radiométrica.

As bandas espectrais dos sensores OLI e TIRS são compatíveis com o sensor ETM+ do Landsat-7 e as respetivas bandas. Todavia, o sensor OLI revelou algumas novidades com o aumento da resolução espetral das bandas, pela introdução de uma nova banda BLUE de comprimento de onda mais curto, de forma a melhorar a sensibilidade à clorofila da vegetação e outras partículas encontrados nos corpos de água. Para além disso, é ainda reconhecia uma nova banda NIR de onda curta. O sensor TIRS, emite radiância a partir de duas bandas de infravermelho térmico de 100 m, possibilitando a correção atmosférica no comprimento de onda térmico, bem como uma determinação exata da temperatura à superfície (Roy, *et al.* 2014).

O satélite Landsat-8, encontra-se a 705 km de altitude, realiza uma órbita circular quase polar, em sincronicidade com o sol, no antigo posicionamento do satélite Landsat-5, recentemente desativo. Os produtos do Landsat-8, assentam em faixas de 185 km e são segmentados em imagens multiespectrais de resolução média de 185 km x 180 km, compostas por 9 bandas espetrais e duas bandas térmicas (Deliry, *et al.* 2021; Roy, *et al.* 2014). O período de revisita do mesmo local é de 16 dias. Em confronto com a missão anterior, o Landsat-8 obtêm,

aproximadamente, mais 60% de imagens, o que resulta no aperfeiçoamento de dados da cobertura de quase todo o planeta, com a mínima incidência de nuvens (Roy, *et al.* 2014).

Tabela IX – Bandas multiespectrais do satélite Landsat-8. Fonte: Li, *et al.* 2015, p. 5986.

Landsat-8		
Bandas	Comprimento de onda (nm)	Resolução espacial (m)
Banda 1 - Blue	0.43–0.45	30
Banda 2 - Blue	0.45–0.51	30
Banda 3 - Green	0.53–0.59	30
Banda 4 - Red	0.64–0.67	30
Banda 5 - Near infrared	0.85–0.88	30
Banda 6 - Shortwave infrared	1.57–1.65	30
Banda 7 - Shortwave infrared	2.11–2.29	30
Banda 8 - Panchromatic	0.50–0.68	15
Banda 9 - Cirrus	1.36–1.38	30
Banda 10 - Thermal infrared	10.60–11.19	100
Banda 11 - Thermal infrared	11.50–12.51	100

De modo geral, os dados Landsat-8 são processados em produtos L1TP, com correções de georreferenciação e radiométricas, compactados em ficheiros *GeoTiff*. Todas as bandas espetrais do sensor OLI e TIRS são guardadas no mesmo ficheiro compactado, em DN de 16 bits, seguidas de um ficheiro associado, relativo aos metadados, com informações detalhadas de cada banda espetral e os valores de deslocação com parâmetros de escala. Estes parâmetros são imprescindíveis para a conversão direta de radiância para refletância TOA (Roy, *et al.* 2014; Deliry, *et al.* 2021). Todos os produtos são projetados no fuso UTM do sistema de coordenadas WGS84, garantindo compatibilidade com os dados Landsat adquiridos pelas missões anteriores (Roy, *et al.* 2014).

3.2 Modelos Digitais de Elevação

O MDE, alude para uma representação da informação da superfície de forma quantitativa, divulgando conhecimentos base sobre o relevo. Inclui vários atributos, nomeadamente inclinação, redes de drenagem, exposições, etc., parâmetros importantes para extração de noções valiosas na avaliação e análise de áreas da superfície terrestre. Variáveis desta natureza são aplicadas em diversos âmbitos dos SIG, como modelação de redes hidrográficas, análise de drenagens à superfície, previsão de riscos de inundação, movimentos de vertentes, produção de mapas hipsométricos, e estudos meteorológicos do solo (Mukherjee, *et al.* 2013).

O MDE é criado através da aplicação de diferentes técnicas, tais como métodos ligados à fotogrametria, aplicada na extração de fotografias, por interferometria, técnica que utiliza, no mínimo a sobreposição de duas ondas, para extrair informação emitida pelas características do relevo, frequentemente aplicada em astronomia. Ademais, são também conhecidas as técnicas de varrimentos topográficos através de *lasers*, métodos de aquisição de fotografias aéreas, levantamentos topográficos e interpolação de mapas de curvas de nível (Mukherjee, *et al.* 2013).

Nestas informações, como em qualquer outro conjunto de dados, é necessário ter em conta os possíveis erros a serem encontrados. Assim sendo, os efeitos ligados a posições geográficas e as condições de relevo, podem influenciar a veracidade dos dados representados pelo MDE. Falhas ao longo do processo de recolha de dados, imprecisões de orientação de imagens e combinações desconhecidas de erros que muitas vezes são inevitáveis são também fatores de desencadeamento de erros (Mukherjee, *et al.* 2013). Contudo, a adaptação para técnicas computacionais de manipulação, análise e distribuição de dados geográficos tem contribuído para a maior rapidez de produção do MDE aliada a menores custos de execução de processos de levantamento, reduzindo a possibilidade de defeitos e incerteza da informação.

Nos últimos anos, o MDE tem contribuído de forma acrescida na elaboração de mapas de uso e ocupação do solo, desempenhando um papel crucial em tomadas de decisão e distinção de classes de ocupação do solo. No território continental, a informação ligada à elevação é essencial, na compreensão dos tipos de uso e ocupação do solo em diferentes áreas geográficas. A título de exemplo, observa-se que no topo da Serra da Estrela prevalece uma vegetação de porte mais baixo e rasteiro devido à influência das grandes altitudes e, por conseguinte, de fatores atmosféricos. Essas condições restritas favorecem exclusivamente o desenvolvimento deste tipo de vegetação. À medida que se desce de altitude, começa-se a evidenciar da presença

de outras espécies vegetais, como arbustos ou árvores, que não são encontradas em grandes elevações, como no alto da Serra da Estrela.

Isto para referir que se verifica uma correlação significativa entre a elevação e determinadas classes do solo presentes na superfície. O MDE fornece, assim, informações indispensáveis para o desenvolvimento de trabalhos de amplitudes similares. Além disso, pode vir a verificar-se que a inclusão de dados relativos a altitudes pode determinar de forma exata classes de ocupação do solo, e ainda, acarretar uma maior contribuição do que algumas bandas de imagens de satélite.

3.2.1 Produtos de Modelos Digitais de Elevação

Na aquisição de dados para MDE deve ter-se em consideração a qualidade e a resolução da informação altimétrica em função do objetivo e posterior aplicação. A qualidade do modelo é estabelecida pela precisão dos dados, já a resolução diz respeito à concisão da informação, particularmente em relação ao tamanho da célula na imagem. Células de dimensões mais pequenas, resultam numa maior resolução espacial e detalhe sobre a informação altimétrica. Além disso, é importante ter em consideração a variação vertical dos valores de elevação, que desempenham um papel importante, uma vez que indicam a variação de altitudes entre os pontos dos dados adjacentes. Uma variação mais baixa assinala uma resolução vertical mais precisa, possibilitando a determinação de variações de altitude menos assentes e um detalhe superior (Garbrecht & Martz, 2000).

A seleção de MDE de alta qualidade e precisão informativa é um processo complexo, uma vez que estes critérios estão intrinsecamente ligados à forma e aos métodos de produção dos dados da superfície. Estes fatores são motivos de preocupação, contudo, os MDEs de detalhe superior, são também os de acesso mais restrito, dado aos altos custos de aquisição e limitações de dados em determinadas áreas. Todavia, os MDE de livre acesso têm vindo a ganhar alguma importância, relativamente a modelos de altos custos, pela capacidade de substituição em determinadas aplicações sem necessidade de informações rigorosas (Saleem, *et al.* 2019).

Os MDEs de livre acesso disponíveis, como o *Shuttle Radar Topography Mission* (SRTM) e o *Advanced Spaceborne Thermal Emission and Reflectation Radiometer* (ASTER), com resoluções mais baixas, de 30 m em comparação com modelos digitais de elevação pagos, desempenham um

papel importante em várias aplicações. Além disso, são também reconhecidos e estão disponíveis de livre acesso, os dados do ALOS PALSAR RTC, EU-DEM e GMTED2010.

É crucial destacar que a disponibilidade e a resolução espacial dos MDEs de livre acesso podem registar variações dependendo da localização geográfica e da fonte dos dados. Portanto, é essencial realizar uma análise prévia das características de diferentes produtos. Todos eles apresentam vantagens e desvantagens em relação à sua cobertura, resolução espacial e rigor informativo e, portanto, é importante reconhecer se atendem às necessidades do trabalho. Nesse sentido, é relevante realizar uma descrição das características dos diferentes produtos, a fim de determinar o que melhor se adequa desenvolvimento da presente dissertação.

Tabela X – Exemplos de Modelos Digitais de Elevação, respetivas vantagens e desvantagens.

MDE	Agências	Data	Origem dos dados	Métodos e interpolação	Resolução espacial	Sistema de referência e *datum*	Precisão vertical	Extensão Geográfica	Limitações e inconvenientes	Referência bibliográfica
SRTM 1 *Arc-Second Global*	-NGA -NASA	Fevereiro de 2000	Dois instrumentos InSar: -Banda C Radar -Banda X Radar	Regista as características do solo através imagens de radar relativas à superfície terrestre, recolhidas por dois satélites em órbita. Duas imagens de radar são adquiridas para a mesma área, mas em momentos diferentes para que possam ser comparadas. Qualquer movimento da superfície do solo (exemplo: erupção vulcânica), durante o espaço de tempo de deslocação dos satélites, pode ser medido e calculado como uma imagem, ou seja, as diferenças entre os dois sinais permitem a avaliação da elevação da superfície. Os dados SRTM adquiridos podem conter imprecisões e, por esse motivo, são submetidos a processo de correções de erros através de técnicas de calibração. Além disso, para lacunas na informação, são aplicados métodos de interpolação, tendo em vista o preenchimento de espaços vazios. Adicionalmente, são utilizados procedimentos de filtragem com o intuito de remover ruídos e informações indesejadas. É realizada uma comparação com dados provenientes de outras missões ou levantamentos terrestres para avaliar a precisão dos dados obtidos.	1 *arc-second for global coverage* (30 metros)	WGS84 e EGM96	90% (erros inferiores a 10 metros)	Cobertura de dados radar para 80% da superfície terrestre entre 60° norte e 56° sul	-Equívocos presentes no SRTM apresentam um crescimento proporcional à ampliação da densidade de árvores. A banda C utilizada pelo SRTM, não tem a capacidade de penetrar a vegetação e atingir o solo; -A precisão da superfície terrestre representada pelo SRTM, depende da topografia local, com erros maiores em áreas montanhosas do que em superfícies mais planas; -Cobertura limitada em áreas polares; -A precisão vertical pode variar consoante a área.	(O'Loughlin, *et al.* 2016; EROS Center, 2018; VHP, s.d.)
ASTER GDEM	-NASA -METI	Agosto de 2019	-SRTM1 V3 -Alaska DEM -CDED -GMTED2010	Processamento automático de vários ficheiros, utilizando técnicas de correlação estéreo na criação dos modelos digitais de elevação, com base em imagens individuais, após a aplicação de uma máscara de nuvens. Agregação de modelos digitais de elevação incluído produtos com e sem identificação de nuvens, para a eliminação de valores imprecisos. Áreas com informação limitada, utilizam-se vários MDEs de referência para completar os dados ASTER e corrigir eventuais anomalias persistentes. Os dados selecionados foram calculados para obter os valores das células finais, posteriormente divididos em pixéis de 1° de latitude por 1° de longitude, com a sobreposição de um pixel.	1 *arc-second for global coverage* (30 metros)	WGS84 e EGM96	Precisão vertical de 10 metros de erro	Cobertura geográfica de 99% da superfície terrestre compreendendo 83° norte e 83° sul	-Sobreposição de um pixel na parte norte, sul, este e oeste da quadrícula. Na maioria dos casos, as células de bordas sobrepostas têm valores idênticos, mas em algumas situações os valores podem ser diferentes; -Os utilizadores devem utilizar os produtos disponibilizados com a consciência das limitações; -É considerado como um produto sem falhas, exceto para a Gronelândia e para a Antártida; -A qualidade dos dados pode variar de acordo com a localizações geográficas, com maior precisão e incerteza em algumas áreas.	(Abrams, *et al.* 2020; NASA, *et al.* 2019)
EU-DEM v1.1	-EEA no âmbito do programa Copernicus	Abril de 2016	-ICESat -SPOT 2011 -EU-HYDRO -EU-DEM v1.0	Processos de correção de problemas de posicionamento da versão anterior (EU-DEM v1.0) utilizando como referência imagens recolhidas do SPOT 2011. EU-DEM v1.0 é derivado de um processo automático de fusão de dados de modelos digitais de elevação existentes, SRTM e ASTER GDEM. Calibração dos dados utilizando pontos de referência obtidos através da missão ICESat. Identificação, avaliação e eliminação de inconsistências e erros nos dados EU-DEM v1.0. Assegurar dados de redes hidrográficas consistentes e alinhados com os padrões estabelecidos pelo EU-Hydro com o objetivo de criar uma representação da realidade com precisão e de alta qualidade das redes hidrográficas.	25 metros	EPSG: 3035 ETRS89, LAEA	Precisão vertical de mais ou menos 7 metros RMSE	Europa	-Limitações de cobertura, uma vez que a informação abrange apenas o território europeu, excluindo o resto do mundo; -Embora a precisão vertical seja excelente, é necessário considerar cuidadosamente a precisão das informações, uma vez que podem ocorrer erros e imprecisões em determinadas áreas; -Apesar de uma resolução espacial notável, podem existir particularidades em algumas áreas que não detêm da mesma resolução.	(Copernicus, s.d.)

GMTED 2010	-USGS -NGA	Janeiro de 2011	-DTED2 *-Antarctica satellite radar and laser altimeter* DEM -DTED1 -CDED3 -CDED1 *-Greenland satellite radar altimeter* DEM -NED – Alaska *-15-arc-second* SPOT 5 *Reference3D* -GTOPO30 (versão anterior) -GEODATA 9 *second* DEM *version 2*	Derivado de 11 fontes de informações relativas à elevação (*raster*), utiliza uma técnica de fusão de vários modelos digitais de elevação. Os métodos de agregação utilizados, envolvem a elevação mínima, elevação máxima, elevação média, elevação mediana, o desvio padrão da elevação, subamostra sistemática e a identificação de limites de quebra.	30 *arc-seconds* (1 quilómetro) 15 *arc-seconds* (500 metros) 7.5 *arc-seconds* (250 metros)	WGS84	*30 arc-seconds,* a precisão vertical está entre 25 e 42 metros RMSE *15 arc-seconds*, a precisão vertical está entre 29 e 32 metros RMSE *7.5 arc-seconds*, a precisão vertical está entre 26 e 30 metros RMSE	Cobertura global compreendendo a partir da latitude 84° norte até 56° sul para a maioria dos produtos e cobertura de 84° até 90° para vários produtos	-Dependendo da fonte de dados utilizada como entrada, algumas limitações técnicas são encontradas nos produtos de elevação; -Qualquer área geográfica que tenha sido preenchida pela informação do GTOPO30 pode ser problemática devido à qualidade relativamente baixa desses de origem dos respetivos dados; -As elevações mínimas e máximas absolutas podem não coincidir com a realidade para determinados continentes; -Várias funções e algoritmos de generalização aplicados ao longo do processamento dos dados em ambiente SIG (ArcGis) resultam em coordenadas com ligeiras variações nos dados de saída; -Embora o GMTED2010 disponibiliza informações importantes sobre a elevação global, a resolução espacial pode deixar muito a desejar para algumas aplicações que exigem alta resolução; -A precisão vertical pode variar dependendo da localização geográfica e da fonte de dados de entrada; -Limitações temporais, uma vez que os dados foram recolhidos até 2010 e não abrangem alterações topográficas recentes, como erosão do solo, atividades vulcânicas ou movimentos de placas tectónicas.	(Danielson & Gesch, 2011; CCI, s.d.)
ALOS PALSAR RTC	-NICT -JAXA EORC	Janeiro de 2006	-SAR L *band* -ALOS PALSAR L1.1 -NED13 -SRTM GL1 -SRTM US1 -NED1 -NED2	A partir dos produtos ALOS PALSAR de nível 1.1, imagens radar adquiridas em qualquer condição atmosférica, dia e hora (SAR), são aplicadas correções radiométricas da superfície para a produção dos dados finais ALOS PALSAR RTC, desenvolvidos pela ASF. Estas correções envolvem fases de pré-processamento com o auxílio de modelos digitais de elevação adicionais e das imagens ALOS PALSAR L1.1. Durante o pré-processamento, é selecionado o melhor MDE disponível para garantir a cobertura adequada da área de interesse. A partir do MDE selecionado são produzidas simulações de imagens SAR, permitindo o alinhamento e a correção das distorções geométricas das imagens SAR originais. Além disso, as correções das imagens envolvem a criação de uma imagem de fatores de correção, que tem em consideração a diferença da área entre os pixéis das imagens antes e depois da correção. Também é calculada a razão entre o tamanho do pixel das imagens corrigidas e não corrigidas. Estas informações são guardadas posteriormente para o cálculo de multiplicação com as imagens SAR, resultando na imagem final.	12.5 metros	WGS84 e EGM96	Precisão vertical com erros de 5 metros RMSE	Cobertura global que compreende os 87,8° de latitude norte e 75,9° de latitude sul	-O design flexível do ALOS PALSAR permite ângulos fora do seu campo de visualização, podendo afetar a consistência e a compatibilidade dos dados adquiridos; -Devido à complexidade do projeto, o ALOS PALSAR requer procedimentos muito longos de pré e pós-calibração, aumentado o tempo de tratamento e os requisitos de recurso; -A precisão dos processos de correção do terreno dos produtos ALOS PALSAR RTC, dependem da qualidade e resolução dos MDEs utilizados; -Erros de geolocalização podem ser causados por fatores de imprecisões do MDE ou erros nas informações de posição dos satélites; -Podem existir incertezas inerentes ao processo de calibração, acabando por induzir em erros nas imagens finais radiometricamente corrigidas; -A correção da superfície depende essencialmente da disponibilidade de MDEs de alta qualidade e resolução, que podem não estar disponíveis para determinadas áreas.	(Ferreira & Cabral, 2021; Laurencelle, *et al.* 2015; ASF, s.d.)

O modelo EU-DEM v1.1, oferecido pela EEA no âmbito do programa Copernicus, determina-se como uma potencial hipótese, dado ao seu ajuste às necessidades da investigação. Além de incluir produtos das versões anteriores, acaba por ter em consideração as limitações e problemas dos respetivos e, portanto, versão mais recente 1.1, pressupõe a dissolução dos erros. Neste caso específico, as correções implementadas nesta nova versão, visam a utilização de dados de referência, tais como ICESat, SPOT 2011, EU-HYDRO, e como referido, o EU-DEM v1.0 (versão anterior). O EU-DEM v1.0 é derivado de um processo automático de fusão de modelos digitais de elevação SRTM e ASTER GDEM, que de certa forma, integram a nova versão. Os novos dados de apoio, possibilitaram correções ao nível do posicionamento, calibração de pontos de referência, e ainda garantem consistências e alinhamentos dos dados das redes hidrográficas, com o prossuposto objetivo de representação da realidade com algum rigor e qualidade.

Para além disso, beneficia de uma resolução de 25 metros, assim como, um erro de precisão vertical de aproximadamente 7 metros, estabelecendo uma análise de valores de elevação precisos e muito completos. Há que acrescentar ainda, a preferência por uma produção de dados altimétricos para território europeu, assenta na estipulação de que esta abrange todo território de Portugal continental e insular, criando dados mais concisos e seguros por esta restringir a uma extensão limitada. Como referido, a escolha deve-se à preferência por uma cobertura de dados de elevação, focada exclusivamente no território europeu, enquanto outros MDEs focam-se em áreas mais extensas. A maioria destes, têm como objetivo compreender uma cobertura mundial, envolvendo um nível de complexidade superior, ao nível dos métodos de tratamento da informação permitindo a ocorrência de limitações e inconsistências superiores, levantando questões sobre a qualidade dos dados em determinadas áreas.

O EU-DEM v1.1[2], apresenta uma precisão e resolução espacial adequada, com foco principal no território europeu, e mesmo não detendo a resolução mais alta, a estabelecida acaba por ser satisfatória e comparativamente elevada para produtos disponibilizados de acesso livre. Após considerar os pontos mencionados, é possível afirmar que este MDE está pronto para ser utilizado, pois oferece uma resolução espacial adequada para o desenvolvimento do respetivo trabalho.

[2] O EU-DEM v1.1 foi recentemente descontinuado e não está disponível. O substituto encontra-se em: https://spacedata.copernicus.eu/collections/copernicus-digital-elevation-model. Embora exista um novo MDE, o utilizado para o trabalho foi o primeiro, pois era o único disponível quando se realizou a parte prática.

3.3 OpenStreetMap (OSM)

O conjunto de informações geográficas, ou seja, dados que implicam posicionamento geográfico associado, tem desenvolvido um crescimento exponencial, bem como a necessidade de aprendizagem relativa a numerosas aplicações. Dentro destes dados são considerados aqueles criados por entidades oficiais, tal como a DGT, mas também provenientes de IGV concebida por qualquer pessoa que intervenha na sua produção.

No que diz respeito à IGV, destaca-se o OSM. As informações OSM são produzidas por utilizadores que realizam a análise da superfície terrestre, a partir de classes temáticas, com o foco para o uso e ocupação do solo de forma a mapear e dar a conhecer os objetos à superfície. Conforme mencionado anteriormente, a IGV consiste no conjunto de dados georreferenciados produzidos de forma voluntária por cidadãos. E é através do avanço das tecnologias *web*, que se tem observado um crescente aumento da interação dos utilizadores, resultando em uma maior contribuição e disponibilização, bem como no OSM.

A IGV dispõe de dois tipos principais de informação, a implícita, onde o posicionamento georreferenciado é associado a suportes de difusão de informação, nomeadamente imagens e vídeos, ou seja, quando partilha uma publicação auxiliada da fotografia da localização ou quando cria uma publicação e associa a localização geográfica qualquer rede social. O segundo tipo, a IGV explicita, onde o utilizador se foca em atividades que contribuem no mapeamento, através de dados associados a conteúdos geográficos, como a criação de geometrias cartográficas do tipo ponto, linha e polígono no OSM (Senaratne, *et al.* 2017; Jesus, 2020). Esta informação apresenta um elevado grau de importância na prevenção e resolução de riscos e desastres, uma vez que, a recolha de informação geográfica em tempo real, possibilita o desenvolvimento de respostas eficazes para solucionar problemas desta natureza. A disponibilização destes dados por parte de qualquer utilizador que pretenda intervir, está associada a avaliação de qualidade, pois nem toda a informação pode ser considerada correta, ou próxima da realidade (Jesus, 2020).

Posto isto, foram apresentadas três formas de avaliação da IGV, através de aproximações *crowd-sourcing*, social e geográfica. A aproximação *crowd-sourcing*, assume que os dados são fornecidos por qualquer utilizador, sem considerar a formação ou o nível de ensino, para que seja feita correção de erros. São muito comuns erros geográficos diretamente associados à localização geográfica, uma vez que, pode não existir conhecimento na área e as imagens aéreas utilizadas não se encontram ortorretificadas (Goodchild & Li, 2012; Jesus, 2020). A

aproximação social atribui reputações aos utilizadores voluntários, pelo seu contributo, criando uma hierarquização, considerando que as pessoas com maiores cooperações e avaliações positivas tenham conteúdos mais concretos e de confiança superior. Por fim, na aproximação geográfica aplicam-se regras de confirmação para validar a IGV criada e editada a partir do reconhecimento geográfico e da legislação da região em questão. Por exemplo, seria impensável numa região classificada pela classe corpo de água existirem edifícios (Goodchild & Li, 2012; Jesus, 2020).

O OSM é um dos projetos colaborativos mais conhecidos na área dos dados geográficos que permite a introdução, criação e edição de IGV. Com a sua conceção no Colégio Universitário de Londres em 2004, por Steve Coast, este projeto engloba atualmente milhões de pessoas pelo mundo inteiro, com principal foco para desenvolvimento de informações de zonas urbanas, originando mapas de dados vetoriais com geometrias próprias do tipo ponto, linha e polígonos (Patriarca, *et al.* 2019). Para além das geometrias, são conhecidos nos dados OSM as relações lógicas, que podem ser utilizadas para a conexão de elementos que não se encontram fisicamente conectados. A estas relações são associados atributos ou *tags*, formadas por uma chave e um valor, onde cada elemento pode ter uma ou várias *tags*, que se encontram incutidas numa longa lista da comunidade OSM. Estas destacam a vantagem de não impor restrições aos voluntários, acabando por contribuir para o enriquecimento da informação do OSM (Patriarca, *et al.* 2019). Por outro lado, torna a utilização dos dados do OSM mais complexa, sendo que os significados das *tags* pode variar consoante o local da informação produzida pelo utilizador. Além disso, a informação da *tag* pode trazer alguma incerteza da informação, assim como diferentes *tags* podem representar a mesma característica da superfície (Patriarca, *et al.* 2019).

Este projeto tem como objetivo principal, efetivar informação para o mundo inteiro com disponibilização de livre acesso aos dados por parte de qualquer utilizador, assim como de toda a população que pretenda trabalhar com esta informação do tipo vetorial. Esta realidade acaba por lutar e contrariar as empresas de organizações comunitárias que desenvolvem e disponibilizam dados geográficos que requerem compensação financeira (Jesus, 2020). O aumento da sua popularidade possibilitou o incremento na quantidade e qualidade dos dados, tornando-os aplicáveis em diversas áreas. A utilização dos dados OSM para a produção de conjuntos de dados de treino é um exemplo das suas aplicações, assim como na classificação subsequente de imagens e criação de mapas de uso e ocupação do solo, compensando a criação exaustiva de dados de treino manualmente por parte do operador.

3.4 Área de estudo

Era necessário definir uma área de estudo para desenvolver a investigação, e deste modo, foi selecionada a ilha Terceira, que faz parte do conjunto de ilhas do Arquipélago dos Açores. Este localiza-se no Oceano Atlântico Norte, entre as latitudes 37º e 40º N e as longitudes 25º e 31º W, sendo composto por um total de nove ilhas. O conjunto de ilhas têm uma distribuição espacial ao longo de aproximadamente 600 km, segundo a orientação de WNW-ESSE, com a formação de três grupos diferentes. Subdividem-se no grupo ocidental organizado pelas ilhas do Corvo e das Flores, o grupo central composto pelas ilhas do Faial, Pico, São Jorge, Graciosa e a Terceira e por fim o grupo oriental, integrado pelas ilhas de São Miguel e Santa Maria.

A ilha Terceira, selecionada como área de estudo, é reconhecida como a mais oriental do grupo central e encontra-se situada entre as latitudes 38º 37' e 38º 48' N, e as longitudes 27º 02' e 27º 23' W. É uma ilha caracterizada pela sua origem vulcânica, apresentando um contexto geodinâmico complexo resultado da interação tripla das placas tectónicas Euroasiática, Africana e Americana. A Terceira ocupa uma área de sensivelmente 400 km², formando um prolongamento geral na orientação de E-W, determinando um cumprimento máximo de cerca de 30 km e uma largura próxima dos 19 km. Caracteriza-se, de um modo geral, como uma ilha com altitudes relativamente baixas e o seu ponto de cota mais elevado, é representado com 1021 m, situado no marco geodésico de Santa Bárbara, no bordo S desta caldeira.

Dividida por dois concelhos, Angra do Heroísmo e Praia da Vitória, esta ilha encontra-se repleta de natureza, apresentando uma densidade populacional baixa, população dispersa, com predominância de campos agrícolas e pastos e pequenos aglomerados de edifícios. É evidente que a ilha Terceira apresenta uma predominância significativa de áreas verdes, constituídas essencialmente por florestas, matos, parcelas de culturas relacionadas com a atividade agrícola e pastos. Aglomerados populacionais e consequentes áreas de instalação não são, de fato tão vistosos, uma vez que predominam apenas em regiões das cidades com mais densidade. Contudo é possível afirmar, que a região é uma área maioritariamente rural.

Em relação à cobertura dos dados OSM sobre a área de estudo, percebe-se que existem dados para algumas zonas da ilha, principalmente em regiões urbanas. Em contrapartida, são consideradas também algumas regiões na área de estudo com uma ausência de dados,

identificada numa parte mais central, e outra a Este do território, onde a falta de dados não passa por despercebido.

A consideração da ilha Terceira como área de estudo para o desenvolvimento desta investigação, não foi feita de forma aleatória. Alguns aspetos importantes definiram a Terceira como área de trabalho, devido a aspetos ligados à qualidade das imagens de satélites, assim como a cobertura de nuvens sobre a superfície terrestre. Este aspeto é justificado mais à frente, no capítulo 4.2.1.

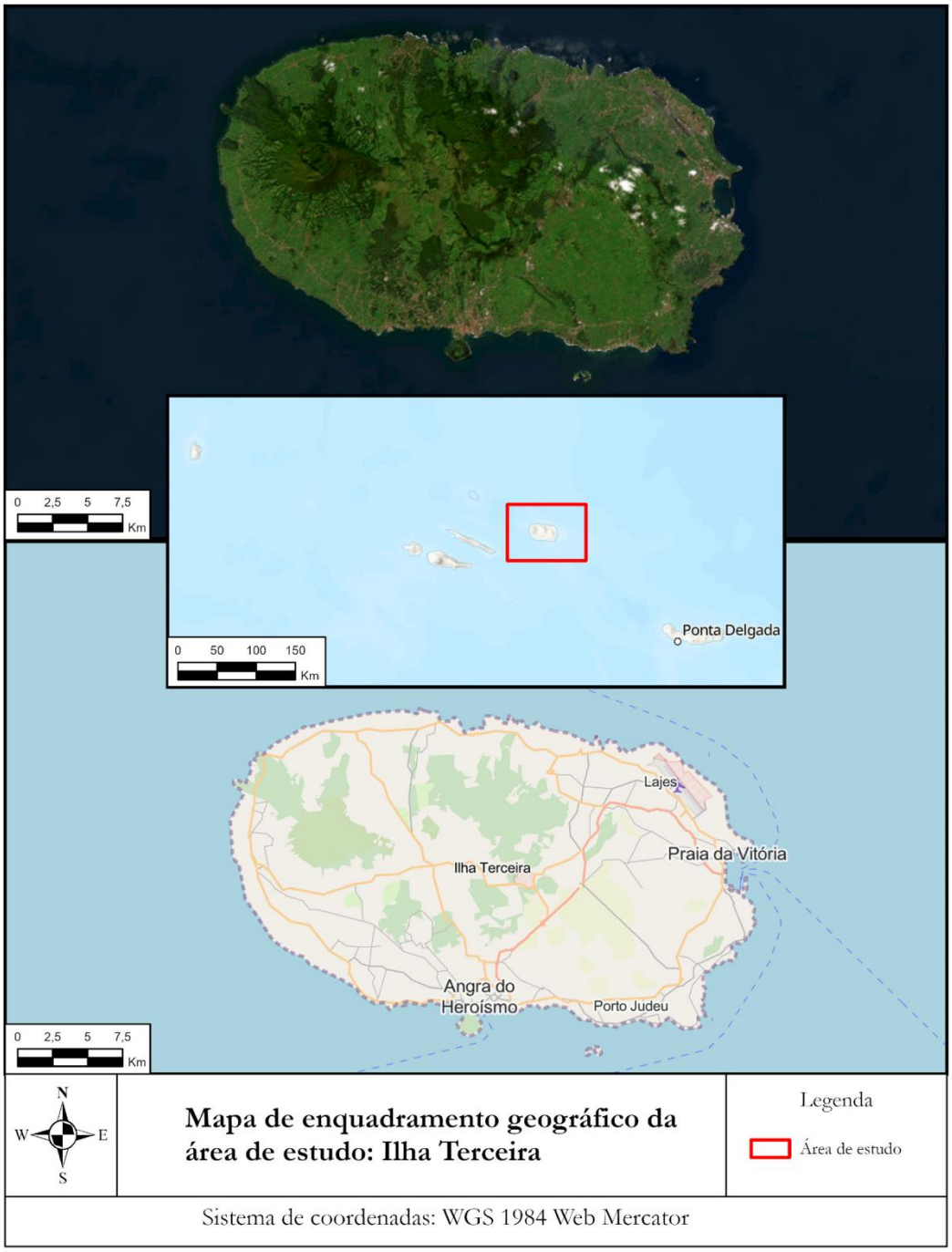


Figura 1 – Enquadramento geográfico da área de estudo.

4 Metodologia

No desenvolvimento de uma metodologia para a classificação automática de imagens de satélite, através da utilização de dados de treino criados sem a intervenção do operador, é fundamental concentrar-se na conceção de diferentes métodos de filtragem desses dados. O objetivo principal é garantir a sua integridade e a aquisição de informações espetrais precisas, a fim de serem utilizadas na classificação das imagens de satélite. A criação de dados de treino de forma manual, é sem dúvida um processo muito demorado e subjetivo e, em contrapartida, a criação automática destes dados é um método desejado para os mais variados projetos de classificação automática supervisionada, como é o caso da presente investigação.

A metodologia desenvolvida no artigo de Fonte, *et al.* (2020) é utilizada como abordagem de base, considerando-se como ponto inicial do desenvolvimento da presente metodologia, contudo, serão implementados novos procedimentos, em função dos métodos utilizados, com o propósito de adquirir resultados mais concretos. É de realçar que os procedimentos metodológicos englobam mais do que considerações relativas a novas formas de filtrar dados de treino. Avaliar o impacto de utilização de mais dados, ou seja, outras imagens de satélite e informações relativas à altimetria na classificação, são também pontos importantes a analisar na execução do projeto.

Com o objetivo de aprimorar o processo de classificação, a partir da metodologia executada no artigo de referência, pretende-se aumentar o número de bandas Sentinel-2 utilizadas, uma vez que não foram implementaram todas em Fonte, *et al.* (2020), incluir novos dados, como um MDE e imagens de satélite de outras missões, como Landsat-8. Estes aspetos adicionais são considerados relevantes para a implementação da atual metodologia. Para além da inclusão de novas fontes de dados para classificação, são ainda seguidas novas etapas, implementadas nos processos de filtragem dos dados treino.

Ainda, em referência à metodologia descrita no artigo de Fonte, *et al.* (2020), destaca-se a utilização de uma abordagem de filtragem de dados apoiada em grelhas vetoriais. Essa abordagem é também utilizada no presente exercício, seguida do cálculo da percentagem ocupação de determinadas classes nas diferentes células, com o objetivo de identificar células com ocupação mista, ou seja, pixéis que são ocupados por mais de uma classe, assim como classes que não ocupam a totalidade da célula. Pressupõe, assim, a exclusão inicial destes pixéis, no sentido de mitigar as interferências nas informações dos respetivos dados de treino e, portanto, será utilizada como modelo para o desenvolvimento da presente metodologia.

Além disso, em Fonte, *et al.* (2020), são efetuados cálculos de índices radiométricos, incluindo NDVI, NDWI e NDBI, no sentido de estabelecer critérios de categorização para cada classe tendo por base os valores obtidos por cada índice correspondente. Deve destacar-se que a definição dos critérios não seguiu parâmetros detalhados, porém, os resultados obtidos consideraram-se valiosos e promissores.

Uma outra alteração relativa a Fonte, *et al.* (2020), pressupõe uma análise mais aprofundada dos critérios relativos aos índices radiométricos. Assim, pretende aumentar-se o cálculo de índices radiométricos, de forma a aumentar o rigor das regras de filtragem, implicando saber os valores que estão, normalmente, associados a cada classe de ocupação do solo. Este último aspeto é conseguido, fazendo a alusão para uma determinada área na COS 2018, seguida de uma identificação manual de uma amostra estratificada de polígonos que contêm apenas elementos pertencentes à classe. Por exemplo na classe de floresta, deseja-se alcançar pixéis puros, sem qualquer outro tipo de ocupação ou até mesmo, em classes de territórios artificializados sem interferência de áreas verdes entre os edifícios. Considerando os polígonos puros da amostra, é importante avaliar, para cada classe, a distribuição dos valores nos vários índices radiométricos, implementados no artigo Fonte, *et al.* (2020) e os mais recentemente definidos na presente metodologia.

Deve referir-se que aquilo que foi mencionado, se trata de uma breve nota sobre os novos procedimentos técnicos que serão implementados na execução da presente metodologia. Definiram-se um grupo de subcapítulos, onde se encontram implementados os métodos e técnicas que, pela sua especificidade, merecem uma explicação fundamentada e profunda.

Numa primeira fase da implementação da presente metodologia, o subcapítulo 4.1, contextualiza a definição da nomenclatura adotada, permitindo estabelecer as classes temáticas de ocupação do solo, posteriormente representadas na nomenclatura dos resultados dos mapas de ocupação do solo. De seguida, no subcapítulo 4.2, destina-se à descrição que remete para a recolha dos dados, com a secção 4.2.1 dedicada à seleção e divulgação das características das imagens de satélite Sentinel-2 e Landsat-8, bem como do MDE. Além disso, ainda no subcapítulo 4.2, é ponderada a conversão dos dados obtidos do OSM para classes CLC, por meio do programa OSM2LULC (secção 4.2.2). Posteriormente, na secção 4.2.3, são descritas as abordagens utilizadas na harmonização das classes resultantes da conversão, de acordo com a nomenclatura definida.

No subcapítulo 4.3, segue-se a descrição de um procedimento básico, aliado a um problema clássico na classificação de imagens através da utilização de dados de diferentes fontes, em

que dados recolhidos de fontes originais, necessitam de ser submetidos a processos de preparação e pré-processamentos, antes de alcançarem a etapa de classificação. Este subcapítulo, destina-se à descrição de toda a fase de pré-processamento das imagens de satélite e do MDE, assim como discussão de tomadas de decisão para a seleção dos métodos de interpolação mais adequados ao longo do processamento dos diferentes dados.

Em seguida, avança-se para os processos de filtragem dos dados de treino, onde na secção 4.4 são descritos pontos determinantes para a metodologia, como os dados obtidos a partir da conversão do *software* OSM2LULC, são submetidos às fases de filtragem e de seleção, aliada à utilização conjunta de imagens Sentinel-2 para o cálculo de índices radiométricos.

O subcapítulo 4.5 é dedicado à explicação detalhada de todo processo exaustivo adotado para definição das novas regras para cada índice radiométrico, de maneira a definir o intervalo de valores que permitem a seleção pormenorizada das informações por classe, que devem ou não, ser incluídas nos dados de treino. Esta fase tem um grande peso no processo de filtragem dos dados de treino, da secção anterior.

Na seguinte subsecção 4.6, apresentam-se princípios ideológicos da metodologia, onde no subcapítulo 4.6.1 se justifica e explica a utilização de determinados índices radiométricos, de forma a garantir o seu peso na individualização de classes de ocupação. Segue-se a secção 4.6.2 para definição da combinação de *features* (Sentinel-2, Landsat-8 e MDE) e descrição dos dados de treino, e distinção entre eles, mediante das regras de filtragem implementadas. Como última secção deste conjunto de subcapítulos, a 4.6.3 tem o objetivo de descrever e justificar o classificador definido para a classificação das imagens e posterior criação de mapas de ocupação do solo.

Por fim, resta apenas assinalar no subcapítulo 4.7, os métodos utilizados para a validação da classificação, com referência para duas estratégias adotadas, que envolvem a construção de matrizes de confusão e de contingência, assim como o cálculo de medidas de validação.

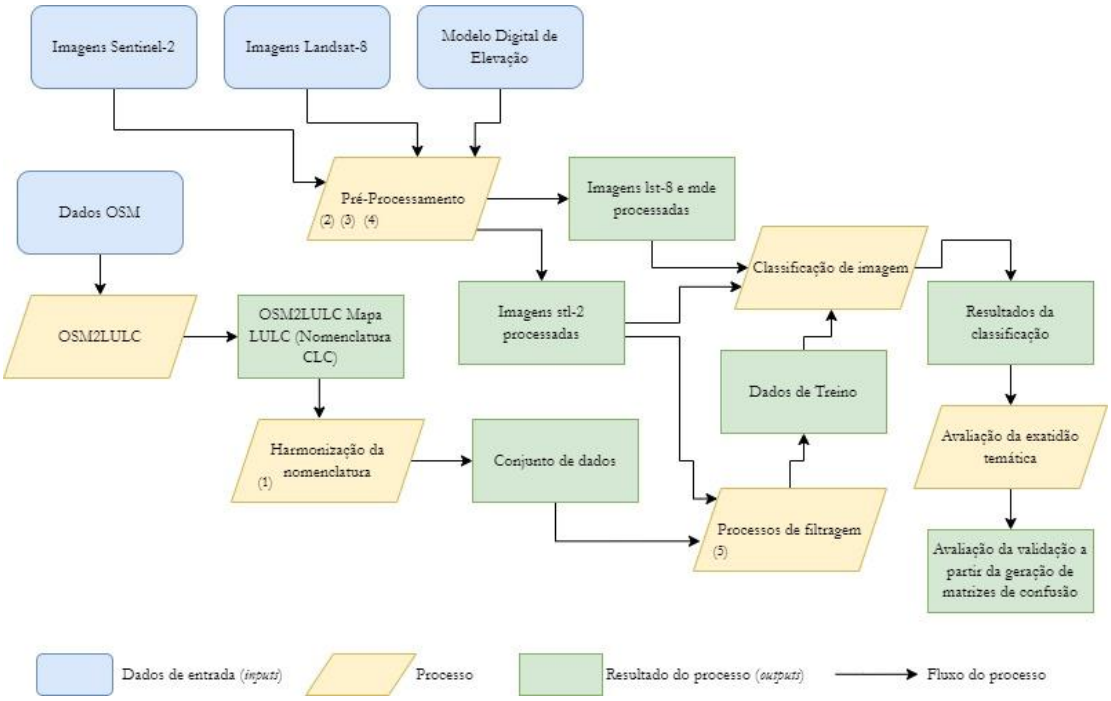


Figura 2 – Fluxograma das etapas da metodologia e fases de análise mais aprofundada.
(1) Harmonização da nomenclatura baseada na abordagem CLC+ *Backbone*;
(2) Pré-processamento das imagens Sentinel-2;
(3) Pré-processamento das imagens Landsat-8;
(4) Pré-processamento do modelo digital de elevação;
(5) Aplicação de filtros aos dados produzidos a partir das imagens Sentinel-2 e dados OSM.

4.1 Seleção da Nomenclatura

O primeiro passo da classificação de imagens de satélite, assenta na definição da nomenclatura a utilizar. Algumas nomenclaturas consideram massas de água e zonas húmidas como informações idênticas, ao ponto de serem englobadas na mesma classe, sendo que outras fazem a distinção. A título do exemplo de certos casos, o uso de nomenclaturas pormenorizadas cria dificuldades só podendo ser utilizadas com todos os instrumentos e ferramentas necessárias para tal.

Neste sentido, nomenclaturas exatas necessitam de alguns trabalhos auxiliares para além da classificação de imagens de satélite, como por exemplo visitas de campo, fotointerpretação ou confirmação da informação geográfica com as comunidades desses territórios para garantir a fiabilidade da classificação. Por outro lado, é importante considerar o tipo de dados que são utilizados e trabalhados. Neste estudo, como são utilizadas e classificadas imagens de satélites, sem qualquer trabalho de campo ou fotointerpretação, imprescindível para a

análise de informações sobre o uso do solo, a nomenclatura dedica-se apenas à ocupação do solo, dado à metodologia adotada para a produção de mapas.

Para a classificação dos mapas, e em alusão às nomenclaturas descritas anteriormente, revelou-se pertinente realizar a classificação de imagens auxiliada as classes da nomenclatura CLC+ BB. Esta diz respeito à segunda geração de novos produtos do *Corine Land Cover* (CLC+), tendo sido considerado como ponto de partida para a verdadeira alteração no paradigma da monitorização da ocupação e uso do solo na Europa (Copernicus, 2022).

O CLC+ BB consiste num grupo de dois produtos, vetorial com um nível de detalhe de 18 classes temáticas e um produto *raster* de 10 metros de resolução espacial para 11 classes de ocupação do solo, produzidos a partir de uma combinação de dados geoespaciais e da EO. Incluído no conceito e componente do CLC+, para além do CLC+BB, é ainda conhecido CLC+ Core, um conjunto de dados, ainda em desenvolvimento, que oferecerá um repositório de dados mais completo, caracterizado por informações de uso e ocupação do solo. Este incorporará produtos *Corine Land Monitoring Service* existentes e futuros, assim como informações de uso e ocupação do solo de produtos nacionais, tendo como base as ideias e o modelo de dados do EAGLE. Por fim, encontra-se também em desenvolvimento, a componente do CLC+ Instances, que se refere a uma série específica de dados criados a partir do CLC+ Core, de maneira a criar informação particular para adaptação às necessidades de políticas específicas (Copernicus, 2022).

O CLC+ BB, primeira componente da nova era CLC+, é considerado como o único elemento particularmente derivado de dados EO, relativamente às três componentes restantes. Esta nomenclatura parte para um detalhe geométrico especialmente elevado, publicamente disponível para os trinta e oito países membros da EEA, os países cooperantes e Reino Unido. O seu inventário de larga escala é fornecido no sistema de referência ETRS 1989 na projeção LAEA, para dados *raster* apenas, uma vez que a informação vetorial ainda não se encontra disponível (Copernicus, 2022; Probeck, *et al.* 2021).

O produto *raster* do CLC + BB tem na sua base, uma metodologia baseada nas informações dos pixéis de imagens de satélite. A principal fonte de dados para o conjunto de informações *raster* é derivada de imagens Sentinel-2, no entanto, na ausência desta informação, os dados *raster* completam-se através de dados dos satélites Sentinel-1 e Landsat-8, como auxílio da classificação. Os dados vetoriais seguem outra metodologia, combinando segmentos lineares conhecidos como *"hardbones"* e segmentação de imagens para identificação de limites naturais na superfície, conhecidos como "*softbones"*. O conjunto de dados *"hardbone"* é derivado de

dados publicamente disponíveis de redes hidrográficas europeias (rios, lagos, canais) e vias de circulação submetidas a processos de seleção, correção e harmonização, garantindo uma unidade de largura mínima de 20 metros. A segmentação *"softbone"* define objetos representativos da superfície baseados nos recursos da série temporal do Sentinel-2, distinguidos pelas suas características espetrais e a sua variação ao longo do tempo. A unidade mínima cartográfica dos dados vetoriais é de 0,5 hectares e distingue 18 classes de ocupação do solo com características derivadas do produto *raster*, bem como outras características adicionais (Probeck, *et al.* 2021).

Uma vez que se pretende realizar o inventário relativo às informações de ocupação do solo para os mapas, torna-se crucial empregar uma terminologia que faça referência a classes temáticas relacionadas com a ocupação do solo. Indubitavelmente, o produto *raster* proveniente da componente CLC+BB, possui as características necessárias para atender às exigências do atual exercício. Este produto, conforme mencionado anteriormente, é um mapa que representa a ocupação da superfície com base nas respostas espetrais dos pixéis das imagens Sentinel-2 para o ano de referência de 2018. O conceito da nomenclatura correspondente, adota uma abordagem plausível, em que a maioria das classes são definidas pelo valor limite de ocupação de 50% no respetivo pixel, sendo estabelecida como classe dominante (Copernicus, 2022).

Em casos de uso e ocupação do solo em simultâneo na escala dos 10 metros de resolução, é implementada uma metodologia de DT para complementar as definições das classes que, geralmente, tem como objetivo atribuir, inequivocamente, os pixéis à classe de ocupação, independentemente do uso atribuído à respetiva área. A classe herbácea permanente pode ser considerada como exemplo, mesmo que o uso do solo dominante seja de um pomar ou de qualquer outra cultura permanente (Copernicus, 2022).

A nomenclatura do produto *raster* correspondente, composta por 11 classes, caracteriza as seguintes classes temáticas (Copernicus, 2022):

1. *Sealed* (Áreas seladas): todas as áreas impermeáveis e fechadas, incluindo edifícios, estruturas artificiais e superfícies planas revestidas por materiais impermeáveis, tais como asfalto e cimento;
2. *Woody – needle leaved trees* (Floresta de árvores coníferas): árvores do grupo botânico *Gymnospermae*, que geralmente possuem folhas em forma de agulha;

3. *Woody – broadleaved deciduous trees* (Floresta de árvores de folha caduca e de folhas largas): árvores de folhas largas que caducam durante uma determinada estação do ano;
4. *Woody – broadleaved evergreen trees* (Árvores perenes de folhas largas): árvores de folhas permanentes durante todo o ano;
5. *Low-growing woody plants (bushes, shrubs)* (Plantas lenhosas de baixo crescimento (arbustos)): plantas perenes de crescimento arbustivo, com múltiplos caules, geralmente, com uma altura inferior a 5 metros;
6. *Permanent herbaceous* (Herbáceas permanentes): áreas com uma cobertura contínua de vegetação ao longo do ano, sem qualquer presença de solo exposto, como pastagens naturais não administradas, pastagens permanentemente administradas e áreas aráveis com cobertura vegetal permanente;
7. *Periodically herbaceous* (Herbáceas periódicas): áreas que sofrem de pelo menos uma alteração na ocupação do solo no período de um ano, entre solo exposto e vegetação herbácea, podendo passar por várias alterações entre os dois tipos de ocupação do solo durante o mesmo período;
8. *Lichens and mosses* (Líquen e musgos): engloba três elementos, líquen musgos e uma classe de vegetação típica da tundra presente no norte da Europa;
9. *Non- and sparsely-vegetated* (Áreas de pouca vegetação ou solo descoberto): superfícies de materiais consolidados e não consolidados, bem como solos permanentemente carenciados de vegetação, onde o solo descoberto cobre mais de 70% da superfície;
10. *Water* (Água): presença de água no estado líquido, independentemente da sua localização, forma, salinidade e origem, bem como corpos de água em movimento, cursos de água, corpos de água estáticos, lagos naturais, açudes, piscinas, etc.;
11. *Snow and ice* (Neve e gelo): áreas cobertas por neve e gelo e a combinação dos dois.

A cobertura total do produto *raster* CLC+ BB, é de 6 002 168 km² em território europeu e, para Portugal, as informações relativas às classes temáticas cobrem todo o território continental, incluindo as áreas insulares do país. No que diz respeito à exatidão temática, esta é de 90%, com erros de omissão e comissão inferiores a 15% por cada classe, porém, para classes de alguma especificidade, como as classes de *"Low-growing woody plants"* e *"Lichens and mosses"* os limites de erro podem ser excedidos (Copernicus, 2022).

4.2 Recolha de dados

A primeira etapa para recolha de dados consiste na definição daqueles que serão utilizados como dados para o desenvolvimento do estudo. Dos diferentes grupos de dados serão consideradas imagens, adquiridas dos satélites Sentinel-2, Landsat-8 e, ainda, o EU-DEM v1.1, bem como dados relativos ao OSM.

Ao longo da aquisição dos dados, as imagens passam por um processo de seleção, tendo em consideração a qualidade, comparência de nuvens, incidência de sombras, recortes, disponibilidade, anomalias e distorções. Fatores desta natureza demonstram um forte peso na definição das imagens, tornando-o, uma tarefa exaustiva e desafiadora. Em algumas ocasiões o período temporal das imagens pode ser um fator influente, condicionando as datas que inicialmente se pretendem trabalhar, assim como a definição da extensão geográfica. Operações de pré-processamento de imagens são muitas vezes necessárias, no entanto, mediante da qualidade dos dados, esta etapa preliminar pode ser menos ou mais minuciosa. Estas envolvem etapas de correção geométrica, correção radiométrica, correção atmosférica e, ainda, em casos de necessidade, o preenchimento de lacunas de informações em falta, por meio de dados provenientes de outras imagens.

Assim, a etapa de seleção das imagens de satélite assume fundamental importância, uma vez que exerce influência sobre o subsequente tratamento requerido ou dispensado nas imagens selecionadas. Os dados compostos pelas imagens de satélite, serão apresentados na secção 4.2.1, onde são descritos em conjunto o acesso à sua plataforma *web*, a etapa de seleção e os principais fatores que influenciaram a sua escolha. Quanto às informações OSM, exigem o mesmo acesso à página *web* e a seleção do conjunto de dados.

O OSM é, de facto, conhecido pela sua complexa cobertura de informação geográfica a uma escala global e tem como principal objetivo, alcançar o maior grau de pormenorização e de detalhe da informação para a mesma extensão, o que vai de encontro com o princípio desta investigação.

Indiscutivelmente, vai desempenhar um importante papel no desenvolvimento da metodologia. Devido à característica que lhe concede a abundância de informações e o considerável volume de dados de treino, resultará em informações discriminatórias para a definição das classes temáticas nas imagens de satélite. Todavia, apesar da sua cobertura global, os dados OSM sofrem, igualmente, de carência de dados em determinadas regiões.

Nessa perspetiva, as áreas predominantemente urbanas são reconhecidas por apresentar um detalhe e complexidade informativa mais elevada, enquanto áreas rurais, apresentam-se algumas lacunas informativas e insuficiência de dados.

Da mesma forma que as imagens de satélite, a qualidade dos dados OSM deve ser discutida no decorrer do desenvolvimento do exercício. É importante submeter as informações adquiridas do OSM a processos de filtragem e seleção, de maneira a alcançar, exclusivamente, informações fidedignas. Destitui-se de qualquer confiabilidade, a utilização de dados provenientes do OSM em bruto, para a classificação de imagens de satélite, uma vez que podem conter inconsistências, resultando em produtos desfavoráveis. Deste modo, no presente exercício, a informação será submetida a uma etapa de conversão para classes temáticas de uso e ocupação do solo. É com esta finalidade que se recorre aos dados OSM, com o propósito de estabelecer classes de ocupação do solo, que serão posteriormente submetidas a processos de harmonização e filtragem. Contudo, no subcapítulo 4.2.2 será descrito a fase de conversão dos respetivos dados para classes de ocupação do solo CLC, com recurso ao *software* OSM2LULC e no subcapítulo 4.2.3, será exposta a etapa de harmonização das classes obtidas por meio da conversão para a nomenclatura estabelecida.

4.2.1 Seleção de Imagens de Satélite

O conjunto de imagens provenientes do satélite Sentinel-2 foi descarregado da plataforma *Copernicus Open Acces Hub* da ESA. Os produtos são resultados de processamento de imagens de L2C, adquiridos pelos dois satélites Sentinel-2A e 2B, no período entre os meses de fevereiro e setembro do ano de 2020. As imagens de satélite Landsat-8 foram descarregadas da plataforma *web Earth Explorer* da USGS, constituído por produtos resultantes do processamento das imagens L1TP. As datas das duas imagens enquadram-se no período temporal dos meses de janeiro e setembro do ano de 2020.

Com base nos dados disponíveis, a seleção das imagens não ocorreu de maneira aleatória assim como a determinação da área de estudo, a ilha Terceira do arquipélago dos Açores. Fatores ligados à qualidade das imagens, são sem dúvida um ponto essencial a ter em consideração na escolha das imagens a serem utilizadas. A ampla cobertura de nuvens é outro aspeto relevante a ser considerado, proporcionando dificuldades na seleção final, não só das imagens, assim como na área de estudo. Originalmente, a intenção era trabalhar a ilha mais icónica do arquipélago, São Miguel, no entanto, devido à baixa qualidade das imagens e ao

pequeno número de captações sem cobertura nebulosa sobre a superfície terrestre, foi necessário restabelecer a área de estudo.

De uma forma geral, a cobertura de nuvens sobre o território açoriano é um problema para os satélites, contudo, os produtos dos satélites Sentinel-2 e Landsat-8 albergam uma qualidade superior, com interferência de nuvens, comparativamente reduzida, no caso da ilha Terceira. Também o período temporal em que se encontram inseridas as diferentes imagens, acabou por ser justificado pela qualidade de imagens e cobertura de nuvens. Objetivava-se trabalhar com um período temporal mais amplo, no intervalo compreendido entre 2020 e 2022, com a aquisição de uma imagem de satélite para cada mês, contudo, devido às grandes massas de nuvens sobre a superfície terrestre de que o território açoriano é alvo, ao longo de todo o ano, este objetivo considerou-se inconcretizável.

O conjunto dos produtos retirados do satélite Sentinel-2 é constituído por três imagens diferentes. Relativamente à cobertura de nuvens das imagens, considera-se com o valor mais baixo de 1.51%, a imagem relativa ao dia 24 de julho do ano de 2020 registada às 12h40. Por outro lado, a imagem adquirida com a percentagem mais alta de cobertura de nuvens, remete para 14.39% obtida no dia 15 de setembro de 2020 às 12h50. Os produtos de dados capturados pelo satélite Landsat-8 sofrem da mesma forma com a cobertura de nuvens e, portanto, a imagem do dia 3 de janeiro de 2020, regista o valor mais baixo de 6.71%. De outro modo, a imagem que remete para a data do 15 de setembro de 2020 sofre de uma percentagem de intervenção de nuvens superior, de aproximadamente 22.49%.

Igualmente importante, foi a aquisição do EU-DEM v1.1, com o propósito de auxiliar no processo de classificação de imagens, este foi conseguido com acesso à plataforma online *Copernicus Land Monitoring Service* EU-DEM v1.1. Este produto tem como objetivo representar a informação quantitativa da superfície, divulgando conhecimentos base sobre o relevo. Ao contrário das imagens de satélites, o período do MDE não é tão relevante, mas o sistema de coordenadas atribuído ao respetivo dado difere em relação às imagens de satélite. Enquanto as imagens de satélite se encontram no mesmo sistema de coordenadas WGS84*,* associado à projeção UTM no fuso 26 N, o MDE tem como sistema de referência, o ETRS89 na projeção LAEA. Indubitavelmente, na fase de pré-processamento é necessário ter em consideração os sistemas de referências de ambos os dados e realizar as conversões necessárias, de maneira a igualar o mesmo sistema de referência. Para uma análise mais detalhada dos produtos adquiridos dos satélites Sentinel-2 e Landsat-8 e ainda com referência ao MDE, segue-se a seguinte Tabela XI.

Tabela XI – Características das imagens de satélite e MDE selecionados.

Imagens Sentinel-2							
Nº	**Data**	**Satélite**	**Sensor**	**Nível de processamento**	**Cobertura de nuvens**	**Sistema de referência**	**Estação do ano:**
1º Imagem	12h40 do dia 20/02/2020	Sentinel-2A	MSI	L2C	1.59 %	UTM/WGS84	Inverno
2º Imagem	12h40 do dia 24/07/2020	Sentinel-2B	MSI	L2C	1.51 %	UTM/WGS84	Verão
3º Imagem	12h50 do dia 15/09/2020	Sentinel-2B	MSI	L2C	14.39 %	UTM/WGS84	Verão
Imagens Landsat-8							
1º Imagem	12h29 do dia 03/01/2020	Landsat-8	OLI/TIRS	L1TP	6.71 %	UTM/WGS84	Inverno
2º Imagem	12h29 do dia 15/09/2020	Landsat-8	OLI/TIRS	L1TP	22.49 %	UTM/WGS84	Verão
Modelo Europeu de Elevação Digital							
Imagem	20/04/2016	ICESat, SPOT 2011, EU-HYDRO, EU-DEM v1.0	HRV, HRVIR	-	0%	ETRS89-LAEA	-

4.2.2 Conversão OSM para classes de ocupação do solo

A fim de adquirir as classes de ocupação do solo como dados de treino provenientes da informação geográfica do OSM, é imperativo utilizar o *software* OSM2LULC. Este é um pacote desenvolvido pela Universidade de Coimbra para a conversão automática dos dados de OSM em classes ou mapas LULC (Patriarca, *et al.* 2019). O presente programa incorpora uma interligação lógica e sequencial de diversas ferramentas que pertencem ao FOSS4G e a outros pacotes. É possível aplicá-lo a diferentes modelos de dados vetoriais e matriciais, contando com um largo desenvolvimento de diferentes versões para melhor performance do *software*.

O OSM2LULC é considerado uma poderosa ferramenta de conversão de dados por incorporar no seu programa (Patriarca, *et al.* 2019):

1. Relação direta e única com uma respetiva classe LULC;
2. Transformação de dados OSM do tipo linha para polígonos com a criação de "áreas *buffer*", calculada pela distância entre as linhas e os polígonos ao seu redor, com melhor representação da realidade;
3. Atribuição de uma determinada classe LULC consoante a área e o tamanho do polígono OSM;
4. Transformação das *tags* ferroviárias e redes hidrográficas (geometria de linhas) do OSM, em geometrias de dados do tipo polígonos, com criação de áreas de amortecimento;
5. Completa informação associada aos polígonos OSM que possuem o valor "sim" para a chave "edifício" com informações complementares desse mesmo edifício. Quando não se obtém informação acerca do edifício, todos os polígonos são assumidos na classe LULC como residencial.

Atualmente, o *software* incorpora também algumas regras pertinentes, para o processo de conversão dos dados. A regra da Prioridade, é aplicada após agregação de todas as classes LULC numa única camada, resolvendo problemas associados à sobreposição de polígonos. Adicionalmente, o programa também incorpora a regra de Hierarquia, aplicada em situações de sobreposição de elementos no terreno, como por exemplo vias de circulação sobre linhas de água ou espaços verdes. Esta abordagem atribui diferentes níveis de prioridade às classes LULC consideradas no processo de conversão dos dados OSM, para resolver problemas desta natureza (Patriarca, *et al.* 2019).

A conversão de dados a partir do OSM2LULC, é concretizada a partir das *tags* que se associam aos elementos por uma chave e um valor, acabando por ser benéfico no impedimento de restrições às edições manuais que enriquecem os dados. Pelo contrário, este cenário torna a utilização dos dados mais complexa, uma vez, que diferentes *tags* podem ser associadas a áreas diferentes, assim como *tags* distintas podem ser aplicadas para representar o mesmo tipo de classe (Patriarca, *et al.* 2019).

Contudo, o *software*, considera apenas as *tags* listadas na plataforma online do OSM *Map Features*. Este está preparado para aplicar o mesmo conjunto de tarefas para preparar e concretizar a conversão dos dados OSM. Tem capacidade de criar três nomenclaturas diferentes e segundo o desenvolvedor, é exequível obter resultados das classes CLC, UA e ainda GLC. Na Figura 3 é apresentada, genericamente, a relação entre o conjunto de dados de entrada e os *outputs* desenvolvidos pelo *software* OSM2LULC.

Na presente investigação, utiliza-se o OSM2LULC, com o objetivo de extrair a partir dos dados OSM, classes temáticas CLC em formato vetorial, empregadas posteriormente na fase da metodologia destinada aos processos de filtragem dos dados de treino. No entanto, antes de se prosseguir com o mencionado, é pertinente que as classes resultantes da conversão passem por uma etapa de correspondência com a nomenclatura previamente estabelecida. Essa abordagem será exposta no subcapítulo seguinte, intitulado como 4.2.3.

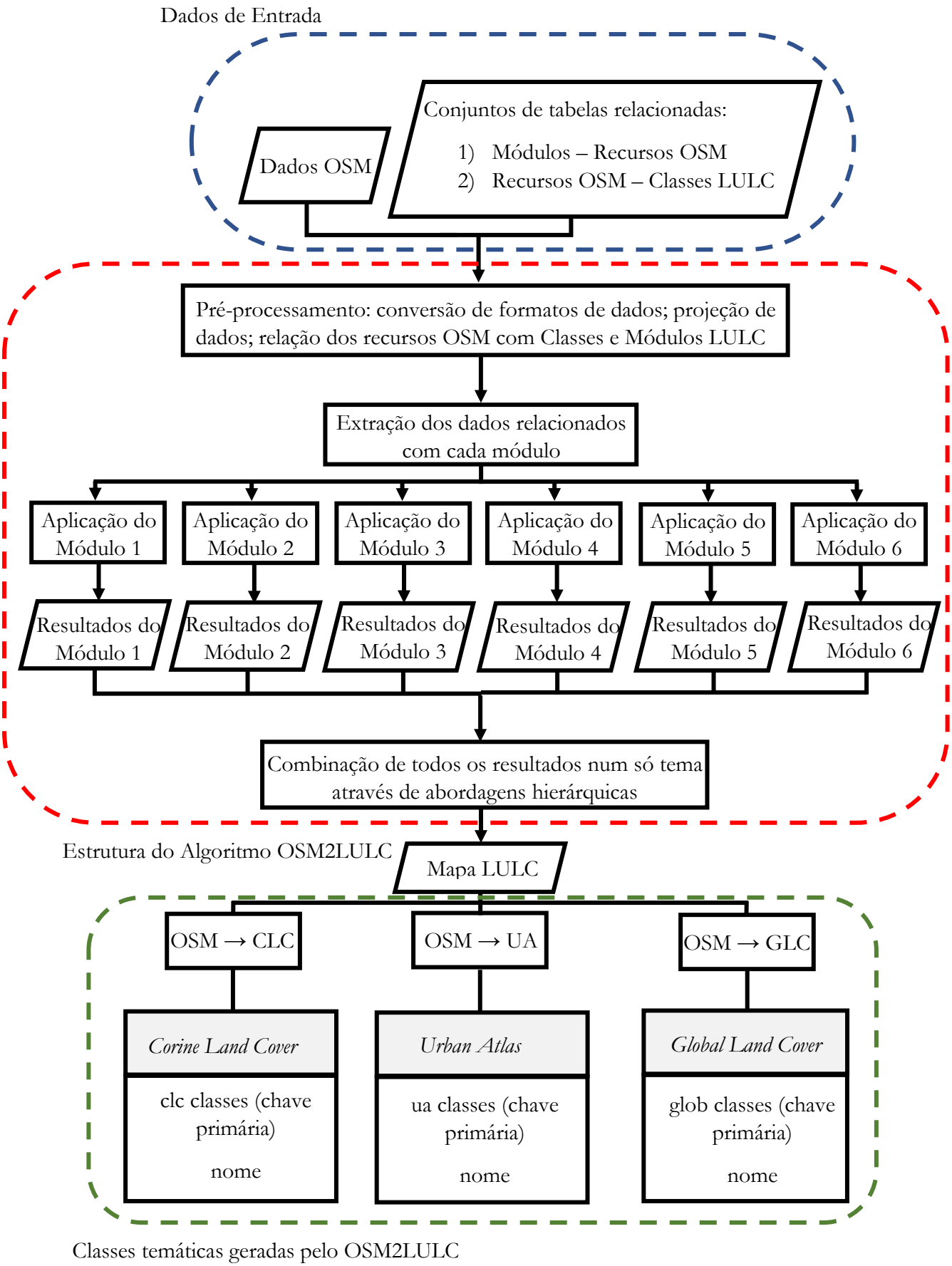


Figura 3 - Esquema da implementação do software OSM2LULC e conjuntos de dados de saída suportados pelas várias versões. Adaptado de: Patriarca, *et al.* 2019, p. 5 e 8.

4.2.3 Harmonização da Nomenclatura (I)

O conceito de nomenclatura, alude para uma lista de classes e à fase de atribuição de nomes às mesmas, que descrevem determinadas características, tendo por base as regras de aplicação dos nomes às classes. A legenda, conceito que corresponde à aplicação da nomenclatura para um propósito específico, lista todas as classes que subsumem todos os objetos presentes nos mapas temáticos para individualização e distinção dos elementos à superfície. A respeito da harmonização de nomenclatura, é relevante para a definição da mesma, realizar uma correspondência, capaz de descrever e conter todas as características da superfície, preservando a facilidade e coesão entre ambas as nomenclaturas.

Neste contexto, foi estabelecida, no subcapítulo 4.1, a nomenclatura CLC+ BB para classificação dos mapas temáticos. Torna-se, ainda assim, imprescindível promover a harmonização entre nomenclaturas, uma vez que a informação OSM extraída, foi integrada na nomenclatura CLC. Para essa finalidade, conforme mencionado anteriormente, recorreu-se ao *software* OSM2LULC, para a extração de um mapa com as classes temáticas relativas à ocupação do solo no formato vetorial. As classes temáticas CLC, são apresentadas no Anexo A1.

É relevante corresponder as classes entre nomenclaturas, seguida de uma análise aprofundada das respetivas discrições das classes de cada terminologia. Considera-se como um processo de extrema importância, uma vez que a associação entre as classes não deve gerar confusão de informações. Assim sendo, a etapa de harmonização assume um papel de extrema importância, representando as classes e respetivas informações da maneira mais precisa possível.

Numa primeira fase, pretendeu-se levar a cabo, a harmonização dos dados vetoriais CLC para a nomenclatura EAGLE. Pressuponha-se como referência, o bloco LCC integrante da estrutura de matriz EAGLE e as respetivas classes do nível 2 de desagregação para classificação dos mapas. Esse processo foi realizado, porém, ao longo dessa etapa, verificou-se que harmonização entre as nomenclaturas, CLC e EAGLE, não era de possível realização, pois a associação entre as várias classes acaba por ser inconsistente.

O EAGLE produz classes e mapas, fazendo uma classificação de uso e ocupação do solo ao mesmo tempo, o que significa que para uma determinada classe, pode ser atribuída uma classificação de uso e ocupação do solo em simultâneo, ou seja, em algumas situações não é

feita a distinção entre os dois conceitos que caracterizam a superfície. De acordo com o objetivo de concretizar uma classificação aliada à ocupação do solo, o conceito EAGLE não se encontra em conformidade com os parâmetros estabelecidos, na medida em que é caracterizada por uma nomenclatura multiclasses, por aludir aos dois termos. Com base na tentativa de harmonização para a nomenclatura EAGLE, o processo acabou por tornar-se difícil. Algumas classes acabaram por não ter qualquer impacto na harmonização e exemplo disso, foi a desconformidade das classes vegetação suculenta e cactos e líquen, musgos e algas. Concluiu-se que a harmonização para as classes EALGE, apresenta uma baixa correspondência, como se pode verificar na Tabela XII e, por conseguinte, optou-se por uma nomenclatura de mapas mais tradicionais, como é na situação da terminologia das classes CLC+ BB.

Tabela XII – Inconsistência da harmonização entre nomenclatura CLC e nível 2 das classes LCC da nomenclatura EAGLE.

<table>
<tr><th colspan="3">Correspondência entre nomenclatura CLC e nível 2 das LCC da nomenclatura EALGE</th></tr>
<tr><th>Classes da nomenclatura CLC</th><th>Nível 2 das classes LCC da nomenclatura EAGLE</th><th>Avaliação</th></tr>
<tr><td>1.1.1 Tecido urbano contínuo</td><td rowspan="7">1.1 Superfícies artificiais e construídas</td><td rowspan="7">Correspondência total</td></tr>
<tr><td>1.1.2 Tecido urbano descontínuo</td></tr>
<tr><td>1.2.1 Indústria, comércio e equipamentos gerais</td></tr>
<tr><td>1.2.2 Redes viárias e ferroviárias e espaços associados</td></tr>
<tr><td>1.2.3 Áreas portuárias</td></tr>
<tr><td>1.2.4 Aeroportos e aeródromos</td></tr>
<tr><td>1.3.1 Áreas de extração de inertes</td><td rowspan="3">1.2 Superfícies de materiais naturais</td><td rowspan="3">Correspondência incompleta: -Áreas de deposição de resíduos, não apresentam materiais no seu estado natural.</td></tr>
<tr><td>1.3.2 Áreas de deposição de resíduos</td></tr>
<tr><td>1.3.3 Áreas em construção</td></tr>
<tr><td>1.4.1 Espaços verdes urbanos
1.4.2 Equipamentos desportivos, culturais e de lazer, e zonas históricas</td><td>1.1 Superfícies artificiais e construídas</td><td>Correspondência incompleta: -Espaços verdes não incluem superfícies artificiais e construídas</td></tr>
</table>

2.1.1 Culturas temporárias de sequeiro 2.1.2 Culturas temporárias de regadio	2.2 Vegetação herbácea	Correspondência incompleta: -Classes que descrevem vegetação herbácea como ocupação, mas também o uso de ocupação do solo, como agrícola.
2.1.3 Arrozais	-	Sem correspondência
2.2.1 Vinhas 2.2.2 Pomares 2.2.3 Olivais	2.1 Vegetação florestal	Correspondência incompleta: -Classes que descrevem vegetação florestal e arbustiva como ocupação, mas também o uso de ocupação do solo, como agrícola.
2.3.1 Pastagens permanentes 2.4.1 Culturas temporárias e/ ou pastagens associadas a culturas permanentes	2.2 Vegetação herbácea	Correspondência total
2.4.2 Sistemas culturais e parcelares complexos 2.4.3 Agricultura com espaços naturais e seminaturais 2.4.4 Sistemas agroflorestais	2.2 Vegetação herbácea	Correspondência incompleta: -Classes que descrevem o uso do solo agrícola, e a vegetação florestal não podem ser inclusas
3.1.1 Florestas folhosas 3.1.2 Florestas de resinosas 3.1.3 Florestas mistas	2.1 Vegetação florestal	Correspondência total
3.2.1 Vegetação herbácea natural	2.2 Vegetação herbácea	Correspondência total
3.2.2 Matos	2.1 Vegetação florestal	Correspondência incompleta: -Agrega a vegetação florestal e arbustiva (os arbustos não são distinguidos de árvores)
3.2.3 Vegetação esclerófila	2.2 Vegetação herbácea	Correspondência total
3.2.4 Florestas aberta, cortes e novas plantações	2.1 Vegetação florestal	Correspondência incompleta: -Cortes rasos de vegetação florestal pode não ser considerado como vegetação
3.3.1 Praias, dunas e areias 3.3.2 Rocha nua 3.3.3 Vegetação esparsa 3.3.4 Áreas ardidas	1.2 Superfícies de materiais naturais	Correspondência incompleta: -A vegetação não pode ser incluída
3.3.5 Neves e Glaciares	3.2 Corpos de água sólidos	Correspondência total
4.1.1 Paúis 4.1.2 Turfeiras 4.2.1 Sapais	-	Sem correspondência
4.2.2 Salinas e aquicultura litoral	3.1 Corpos de água líquidos	Correspondência total
4.2.3 Zonas entremarés	-	Sem correspondência
5.1.1 Cursos de água 5.1.2 Planos de água 5.2.1 Lagoas costeiras 5.2.2 Desembocaduras fluviais 5.2.3 Oceano	3.1 Corpos de água líquidos	Correspondência total

O processo de harmonização das classes CLC foi concretizado com as classes da nomenclatura CLC+ BB, resultando na superação das dificuldades anteriormente sentidas. O processo para a respetiva nomenclatura, desencadeou diversas técnicas, incluindo a abordagem espontânea, admitindo uma correspondência direta entre as classes e a abordagem de generalização de classes, que se refere ao processo de tornar os termos das classes mais amplos ou abrangentes, mediante das características de determinadas classes.

Na atual etapa de harmonização para classes CLC+ BB, mesmo sucedendo como uma versão atualizada do CLC, com um maior detalhe e atualização, possibilitando uma compressão mais aprofundada dos recursos à superfície, surgiu uma evidente complexidade na harmonia entre as classes. Ao longo da abordagem, as classes CLC+ BB, necessitaram de uma generalização informativa entre as 11 classes correspondentes. Posto isto, foram necessárias várias agregações, como o respetivo caso das três classes de descrição da vegetação arbórea, classes *Woody* (*needle leaved trees*), *Woody* (*broadleaved deciduous trees*) e *Woody* (*broadleaved evergreen trees*), conjugadas em apenas uma definida por *Forest*. Ademais, tal método, incluiu ainda, as duas classes de referência aos tipos de herbáceas, definidas como classe *Herbaceous,* na nomenclatura harmonizada.

Além do procedimento em questão, duas classes CLC+ BB foram submetidas a supressão, nomeadamente a classe *Lichens and mosses* e a classe *Snow and ice*. Tal decisão deveu-se ao fato de no território português serem consideradas como informações com influência praticamente nula e, portanto, não fazia sentido integrarem-se na nomenclatura, dada à sua inexistência. Considerando a correspondência entre as classes das nomenclaturas no contexto do exercício, e tendo em questão o significado das classes, considerou-se como abordagem espontânea, a relação direta de determinadas classes CLC para classes CLC+ BB, como *Sealed Surface*, *Herbaceous*, *Partly vegetated or Non-vegetated land* e *Water*, considerando-se como uma simples fase de harmonização. Por sua vez, a harmonização de classes CLC para a nomenclatura definida ocasionou dificuldades, em *Forest* e *Shrubland,* onde algumas classes CLC desencadearam incertezas e, portanto, recorreu-se a uma generalização.

Por fim, para que classes CLC remetessem para a descrição de áreas húmidas, regiões com presença de água, permanente ou sazonal, integrou-se uma classe destinada à caracterização deste tipo de ocupação do solo, de modo a alcançar maior consistência. A cabo do processo de harmonização entre as nomenclaturas CLC e CLC+ BB, verifica-se, na Tabela XIII, a correspondência para um conjunto de sete classes distintas a utilizar, posteriormente, na classificação de imagens.

Tabela XIII – Harmonização entre nomenclatura CLC e classes CLC+ *Backbone*.

Harmonização da Nomenclatura CLC e CLC+ *Backbone*	
1.1.1 Tecido urbano contínuo 1.1.2 Tecido urbano descontínuo 1.2.1 Indústria, comércio e equipamentos gerais 1.2.2 Redes viárias e ferroviárias e espaços associados 1.2.3 Áreas portuárias 1.2.4 Aeroportos e aeródromos 1.4.2 Equipamentos desportivos, culturais e de lazer e zonas históricas	*Sealed Surface* (Espaços fechados ou selados)
2.2.2 Pomares 2.2.3 Olivais 2.4.1 Culturas temporárias e/ ou pastagens associadas a culturas permanentes 2.4.2 Sistemas culturais e parcelares complexos 2.4.4 Sistemas agro-florestais 3.1.1 Florestas folhosas 3.1.2 Florestas de resinosas 3.1.3 Florestas mistas 3.2.4 Florestas aberta, cortes e novas plantações 1.4.1 Espaços verdes urbanos	*Forest* (Floresta)
2.2.1 Vinhas 3.2.2 Matos 3.2.3 Vegetação esclerófila 2.4.3 Agricultura com espaços naturais e semi-naturais	*Shrubland* (Arbustos)
2.1.1 Culturas temporárias de sequeiro 2.1.2 Culturas temporárias de regadio 2.3.1 Pastagens permanentes 3.2.1 Vegetação herbácea natural	*Herbaceous* (Herbáceas)
2.1.3 Arrozais 4.2.3 Zonas entre-marés 4.1.1 Paúis 4.1.2 Turfeiras 4.2.1 Sapais	*Wetlands* (Áreas húmidas)
1.3.1 Áreas de extração de inertes 1.3.2 Áreas de deposição de resíduos 1.3.3 Áreas em construção 3.3.4 Áreas ardidas 3.3.1 Praias, dunas e areias 3.3.2 Rocha nua 3.3.3 Vegetação esparsa	*Partly vegetated or Non-vegetated land* (Áreas de pouca vegetação ou solo descoberto)
5.2.1 Lagoas costeiras 5.2.2 Desembocaduras fluviais 5.2.3 Oceano 3.3.5 Neves e Glaciares 4.2.2 Salinas e aquicultura litoral 5.1.1 Cursos de água 5.1.2 Planos de água	*Water* (Corpos de água)

4.3 Pré-processamento (Harmonização de dados)

Segundo os critérios apresentados no fluxo de trabalho da Figura 2 do capítulo 4, com finalidade de mostrar o esquema genérico para a execução do presente exercício, apresenta-se um fluxograma de análise de tarefas. Neste são inicialmente avaliadas as etapas de pré-processamento dos diversos dados utilizados. Essas etapas têm como objetivo harmonizar e igualar as características das informações adquiridas, garantindo a uniformidade relativas à resolução espacial, sistema de coordenadas e projeção, extensão geográfica e mitigações de erros, bem como os efeitos atmosféricos encontrados nas imagens (2), (3) e (4).

A operacionalização desta primeira fase metodológica, de pré-processamento dos diferentes dados espaciais, só pode ser completada com recurso a *software* capaz de executar ferramentas de processamento estatístico, manipulação, análise e visualização de informações geoespaciais. Para o efeito, explorou-se o conhecimento de uma solução SIG com capacidade para proceder a este tipo de avaliação, por via da programação de várias ferramentas, quando possível, de modo a automatizar tarefas, simplificar processos e melhorar a eficiência geral do trabalho, mitigando erros de processamento e desenvolvendo conhecimentos de programação, lógica e resolução de problemas.

De modo geral, a presente fase de preparação dos diferentes dados de imagens, reside na submissão de diversas fases de tratamento, nomeadamente em etapas de reamostragem, reconhecido como um método estatístico para a transformação das imagens, relativamente à alteração do tamanho das células que compõem as imagens, operações de corte pela área de estudo, correções atmosféricas com a exclusão de pixéis associados a nuvens e ainda correções topográficas.

Contudo, antes de proceder à análise da fase de pré-processamento de imagens, é importante fundamentar conceitos referentes aos métodos de reamostragem aplicados nos diferentes dados, uma vez que se considera como um passo essencial da correção de distorções. Assim, é fundamental compreender os princípios básicos que determinam o método mais adequado de utilização. Isso permitirá o conhecimento das particularidades e, consequentemente, auxiliará no entendimento e definição dos novos valores de pixéis das imagens reamostradas. É neste sentido que se seguem os próximos subcapítulos, explicando o que se entende por reamostragem e os diferentes métodos de interpolação que a compõem.

4.3.1 Reamostragem (*Resampling*)

A aquisição de imagens de satélite requer, frequentemente, a aplicação de métodos de *resampling* para os sinais contínuos das imagens, como por exemplo, a radiação refletida pela superfície terrestre. Estes dados digitais são apresentados em forma de grelha, onde cada célula ou pixel da imagem contém um DN que representa diferentes tonalidades, das mais escuras para as mais claras, mediante das características do solo (Wolf, *et al.* 2014). A reamostragem é uma técnica utilizada para manipular uma imagem digital e transformá-la noutra, de acordo com as necessidades e os objetivos pretendidos. A preparação de imagens pode ser executada por diversas razões, como a alteração da resolução espacial ou da orientação, ou seja, a rotação, a alteração do posicionamento dos pixéis, entre outras (Gurjar, *et al.* 2005). Esta técnica é, extensivamente, utilizada para processos de imagens em diversas aplicações, como na medicina, na indústria e, claro, na deteção remota.

Os processos de reamostragem de imagens ocorrem porque a imagem, conforme a sua captação, acaba por incorporar algumas limitações causadas pela geometria da imagem e pela capacidade do sensor (Gurjar, *et al.* 2005). Para realizar as correções geométricas de uma imagem de satélite original, o *resampling* é utilizado para determinar valores digitais e colocá-los em novas posições da imagem corrigida, a partir das informações da imagem original (Gurjar, *et al.* 2005).

Em função do método de reamostragem, o resultado obtido pode traduzir-se numa redução do número de pixéis da imagem, implicando a perda de alguma informação dos valores de DN. Por outro lado, é possível afetar o tamanho da representação da imagem com o aumento da resolução dos pixéis da imagem (Manjunatha, *et al.* 2018). Este método envolve a interpolação entre diferentes pixéis que constituem a imagem para sintetizar a informação e para transformação de coordenadas (Wolf, *et al.* 2014).

São vários os métodos disponíveis para executar o *resampling* de imagens de satélite, embora existam três, sendo os mais conhecidos e utilizados: interpolação NN, *Bicubic* e *Bilinear*. Há uma grande diversidade de métodos, no entanto, tendem a ser sensíveis às interferências nas imagens (Wolf, *et al.* 2014). Posto isto, nos subcapítulos seguintes, serão descritos os três métodos com o principal objetivo de analisar as suas características no processamento das imagens de satélite, a fim de entender os seus benefícios e as suas limitações.

4.3.1.1 *Neareast Neighbor*

O processo de *resampling* de imagens por meio da aplicação do método NN consiste na utilização do valor do pixel na imagem original que se encontra mais próximo da nova localização do pixel na imagem corrigida (CCSR, 2013). Do ponto de vista computacional, a aplicação deste método resume-se em arredondar os valores fracionários das linhas e colunas para o valor inteiro mais próximo (Wolf, *et al.* 2014).

Este método é considerado como o mais simples, por preservar os valores originais e ainda por levar menos tempo de processamento computacional em comparação com as outras técnicas (Manjunatha, *et al.* 2018). Todavia, é importante salientar que a aplicação do método NN pode resultar na deslocação de imagem em relação à área de extensão, caracterizadas por bordas recortadas, e podendo afetar tanto a redução como o aumento de pixéis na imagem (Manjunatha, *et al.* 2018).

4.3.1.2 Interpolação *Bilinear*

O método *Bilinear* calcula o valor médio de quatro pixéis da imagem original na nova localização do pixel da imagem corrigida (CCSR, 2013). É importante destacar que este envolve uma complexidade computacional superior e a sua interpolação é realizada na vertical e na horizontal (Manjunatha, *et al.* 2018).

Durante o processo de cálculo, os valores originais dos pixéis podem ser alterados, criando informações digitais completamente novas. Isso pode ser indesejável, quando se pretende realizar uma maior análise nas imagens, como classificação baseada na resposta espetral. Neste caso, é preferível realizar primeiro a classificação e, somente após este processo, aplicar o método de interpolação *Bilinear* (CCSR, 2013).

4.3.1.3 *Bicubic* ou *Cubic Convolution*

A interpolação *Bicubic*, também conhecida como *Cubic Convolution*, é uma técnica de reamostragem amplamente utilizada. Este método é considerado uma evolução das limitações da interpolação *Bilinear*, pois tem em consideração os 16 pixels mais próximos

num total de 4x4 pixels circundantes. (Manjunatha, *et al.* 2018). Tal como a interpolação *Bilinear*, este modelo produz valores de pixéis completamente novos, resultando em superfície interpoladas mais suaves e nítidas em comparação aos métodos de interpolação *Bilinear* e NN, que podem apresentar imagens em blocos (Manjunatha, *et al.* 2018).

Contudo, é importante destacar que a interpolação *Bicubic* requer uma maior capacidade de processamento computacional em comparação com a interpolação *Bilinear*, mas acaba por ser um bom método para as imagens de satélite (CCSR, 2013).

4.3.2 Pré-processamento de imagens Sentinel-2 (2)

No que diz respeito ao pré-processamento (2) das imagens Sentinel-2, a série temporal das várias bandas que as compõem foram submetidas, inicialmente, a um processo de reamostragem, aplicada pela técnica NN ou vizinho mais próximo. Este serviu para avaliar os novos valores dos pixéis, com o objetivo de ajustar o tamanho das bandas de resolução espacial mais baixa para bandas de resolução maior, de 10 metros, utilizando como referência uma banda com a respetiva resolução espacial.

A interpolação NN considerou-se a mais indicada, dado que a nível computacional, é considerado muito eficiente, combinado com a preservação dos valores do pixéis da imagem de entrada, e, ainda, a conservação de detalhes oportunos, como por exemplo as informações relativas a nuvens (Roy, *et al.* 2016).

O estudo de Forkuor, *et al.* (2017), explora a utilização conjunta de dados Landsat-8 e Sentinel-2 para a produção de mapas de uso e ocupação do solo numa determinada região do Burkina Faso, percebendo em simultâneo o contributo de bandas do infravermelho próximo do Sentinel-2 nos resultados dos mapas. No respetivo estudo, revelaram a utilização do método de interpolação do vizinho mais próximo, para realizar a reamostragem de imagens Sentinel-2 e Landsat-8 com resolução original de 30 para 10 metros de resolução. Assim como no estudo em referido, o processo de reamostragem foi útil para a correspondente fase do exercício, visto que se necessitou de ajustar o tamanho do pixel das bandas, de maneira que permitissem ser comparadas e combinadas adequadamente em etapas seguintes da metodologia.

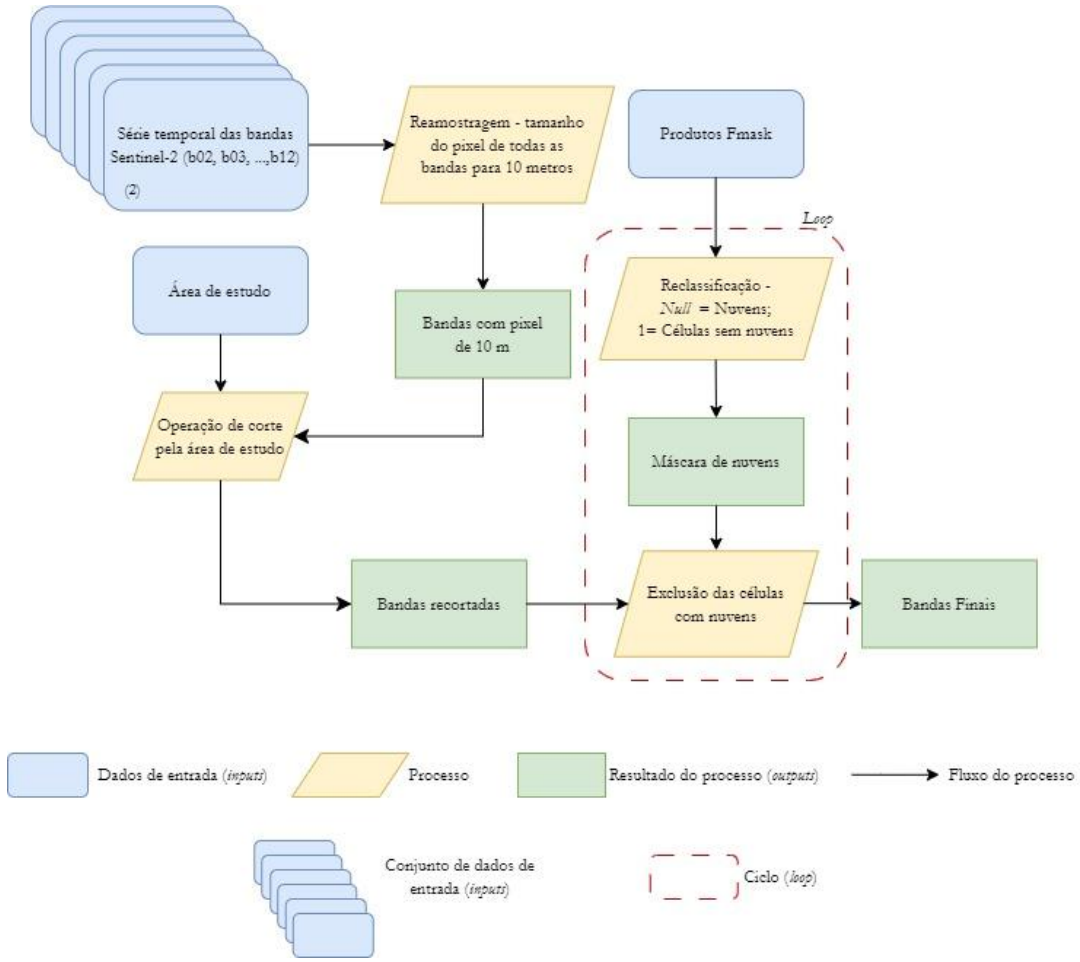


Figura 4 – Fluxograma das fases de pré-processamento das imagens Sentinel-2.

Após obter as bandas que compõem as imagens Sentinel-2 para as diferentes datas, com uma resolução espacial de 10 metros, estas foram submetidas a um recorte pela área de estudo, de maneira a excluir informação desnecessária, limitando os dados apenas para a ilha Terceira. Adquiridas as bandas, estas passaram por um processo de exclusão das células de nuvens, através da conceção de máscaras produzidas a partir do programa Fmask, desenhado por Zhe Zhu e Shi Qiu do Departamento de Recursos Naturais e Ambiente da Universidade de Connecticut. Este *software* é utilizado na deteção de nuvens, sombras de nuvens, e neve, de forma totalmente automática, para uma panóplia de imagens de satélite desde a série Landsat 4-9 até Sentinel-2, com resultados muitos satisfatórios.

Inicialmente, pensou-se conceber produtos SCL, reconhecidos como mapas informativos sobre a ocupação da superfície em diferentes classes temáticas, das quais se individualizam pixéis com informações referentes a nuvens e sombras a partir das bandas das imagens Sentinel-2. No entanto, percebeu-se que os produtos SCL deixaram muito a desejar, visto que apenas detetavam algumas nuvens, não todas, e reconheciam, também, células de classes

de ocupação (*Sealed Surface*) como sombras ou nuvens, com a perda de informação importante. A título de exemplo das lacunas desvendadas pela implementação de máscaras de nuvens através dos produtos SCL, segue-se a Figura 5, cuja inconsistência se relaciona com o aeroporto da área de estudo. Este conduziu a confusão nos produtos de máscaras criados, acabando por ser excluído, sendo considerado como informação relacionada a nuvens ou sombras, proporcionando a exclusão de informação pertinente.

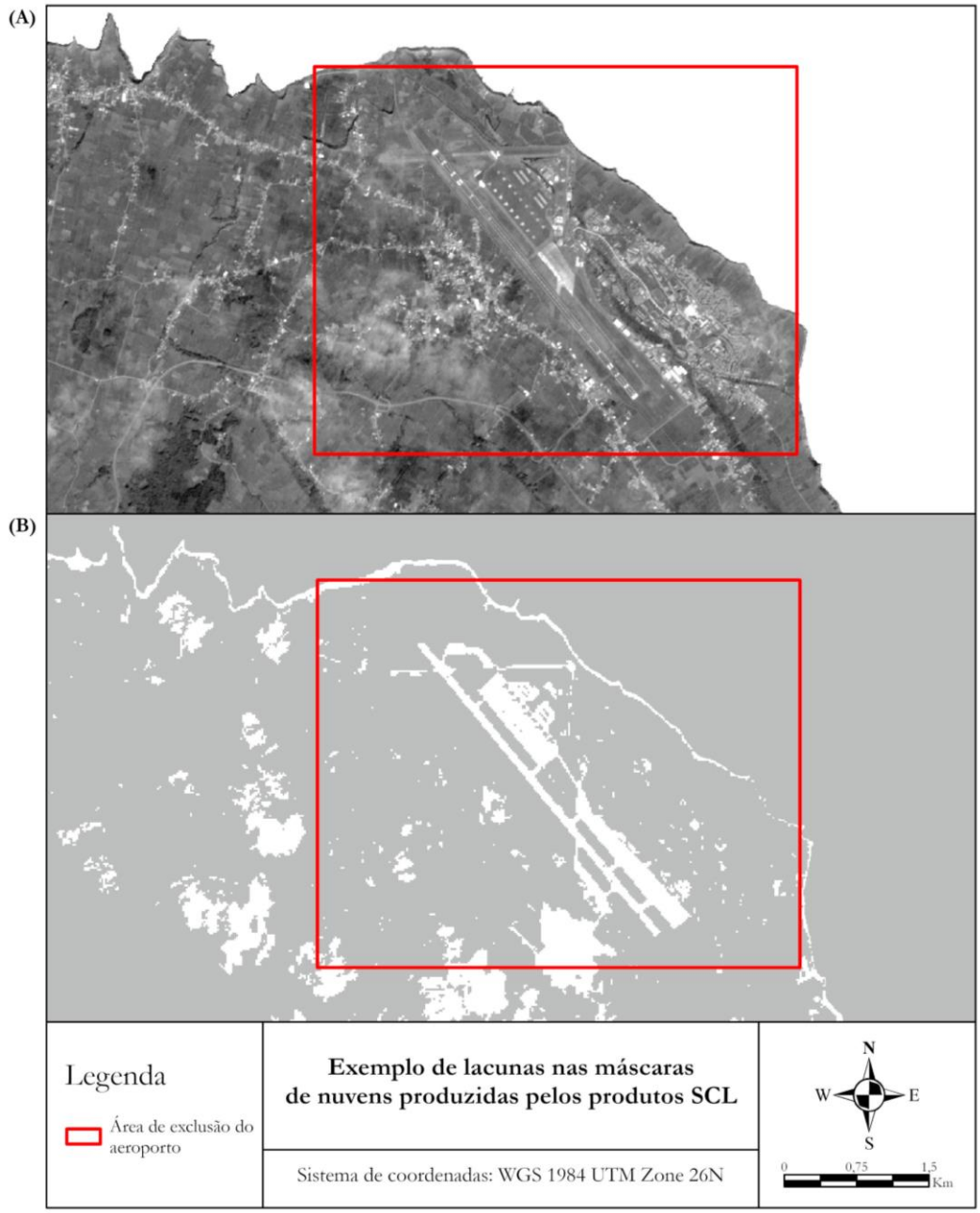


Figura 5 – Localização da área do aeroporto eliminada pelo produto SCL.
(A) Banda 2 da imagem Sentinel-2 sem correções atmosféricas;
(B) Produto SCL com valores *Null* para as perturbações atmosféricas e valor de 1 para a informação sem nuvens.

De maneira a contornar o respetivo *handicap*, em alternativa aos produtos SCL procedeu-se à utilização do *software* Fmask, para identificação e eliminação dos efeitos atmosféricos, considerando a utilização de imagens Sentinel-2 no nível de processamento L1C para as mesmas as datas. A principal razão que levou à aplicação das imagens neste nível, deve-se ao fato destes dados conterem informações da refletância isentas de correções atmosféricas complexas, fator elementar para a deteção de nuvens e sombras. A utilização do segundo nível das imagens Sentinel-2, originado de correções atmosféricas, altera as características das imagens originais, comprometendo o desempenho do algoritmo Fmask na identificação das nuvens e sombras (Qiu, *et al.* 2019). Neste seguimento, aplicaram-se as imagens L1C no processamento de criação de máscaras de nuvens, pelo motivo de preservarem as condições originais das particularidades das imagens, detetando perturbações, enquanto as imagens de processamento L2C direcionaram-se para as etapas de treino e classificação.

É importante referir que a tomada de decisão para a utilização do algoritmo Fmask, advém de uma tentativa de criação de máscaras de nuvens por outra via, apercebendo-se que esta meta não seria alcançável no tempo disponível, pela disposição de habilidades a nível académico para a execução e pela incerteza do melhor rumo a tomar para a solução mais clara. Dada a limitação, foi equacionada e repensada a concretização da meta recorrendo ao programa em questão, como uma forma acessível e fluente de produzir as máscaras de nuvens, com a garantia de que o resultado seria eficiente e confiável.

Após a produção das máscaras, estas são reclassificadas, atribuindo o valor *Null* a células de nuvens e outros efeitos atmosféricos, com o propósito de excluir das bandas e preservar as informações com referência à ocupação do solo atribuídos com valor de 1. Como consta no fluxo de trabalho da Figura 4, depois de criadas as diferentes máscaras de nuvens, estas são agregadas em apenas um ficheiro *raster* binário, para a posterior aplicação em todas as bandas da série temporal das imagens Sentinel-2 de forma sequencial. Este ciclo mostrou-se deveres importante para que toda a informação indesejada, fosse excluída do conjunto de bandas das imagens. Vale ressaltar que no respetivo processo, a extensão da série temporal das imagens, equaciona a informação perdida, devido à área de estudo definida, onde é frequente identificar grandes porções de cobertura de nuvens em qualquer data. Estas quantidades de pixéis referentes a nuvens e sombras, originam naturalmente uma maior exclusão de células de ocupação do solo, reduzindo assim a informação das imagens.

Por fim, as bandas Sentinel-2 encontram-se em condições de avançar para as seguintes etapas das fases da metodologia, uma vez que o conjunto de imagens igualam uma resolução

espacial de 10 metros, apresentam informação restritamente para a área de estudo e encontram-se livres de inconsistências relacionadas à cobertura de nuvens ou de sombras.

4.3.3 Pré-processamento de imagens Landsat-8 (3)

No que concerne às imagens Landsat-8, é imprescindível considerar que o pré-processamento das mesmas teve como principal objetivo, obter uma resolução espacial igual às bandas de 10 metros das imagens Sentinel-2. Contudo, neste estudo, foram utilizados dados Landsat-8 processados do L1TP. Considerando as características inerentes ao respetivo nível de processamento, que não abrange dados corrigidos tanto ao nível geométrico como radiométrico e não considera as características do relevo à superfície, torna-se inevitável a necessidade de condicionar fatores de preparação independentes para a devida aplicação.

Posto isto, e em referência à Figura 6, a primeira fase de pré-processamento das imagens Landsat-8 considera uma correção atmosférica e topográfica. Uma vez que os dados provenientes da série de processamento L1TP fornecem um conjunto de fatores de escala armazenados no ficheiro de metadados, incluído coeficiente de ângulos, tais elementos revelaram-se de elevada importância para a fase inicial de preparação das imagens.

Nesse sentido, a correção atmosférica das imagens, teve em consideração a conversão dos DN das imagens para refletância TOA. Este parâmetro considera-se importante na análise das imagens, pelo facto de eliminar os efeitos da atmosfera que podem afetar a intensidade da radiação capturada pelo sensor. Conta ainda com o aumento das exatidões informativas das imagens relativamente à superfície terrestre.

Por sua vez, tornou-se indispensável realizar em simultâneo uma correção topográfica, inevitavelmente condicionada pela preparação prévia do MDE e aplicações informativas relativas ao ângulo zenital e azimute disponibilizados pela coleção dos dados. O ajuste das condições de iluminação do MDE, com o propósito de determinar a inclinação e orientação das superfícies, foi alcançado por meio da reprojeção do sistema de coordenadas para os dados das imagens correspondentes (UTM/WGS84 zone 26N), e da reamostragem para a resolução espacial de 30 metros, equivalente à resolução das imagens Landsat-8. Estas pequenas fases foram essenciais para complementar a correção de distorções e variações topográficas da área de estudo das imagens Landsat-8. A presente correção serve para

amenizar as condições topográficas que influenciam a quantidade de luz solar refletida ou emitida pela superfície de alcançar o sensor do satélite.

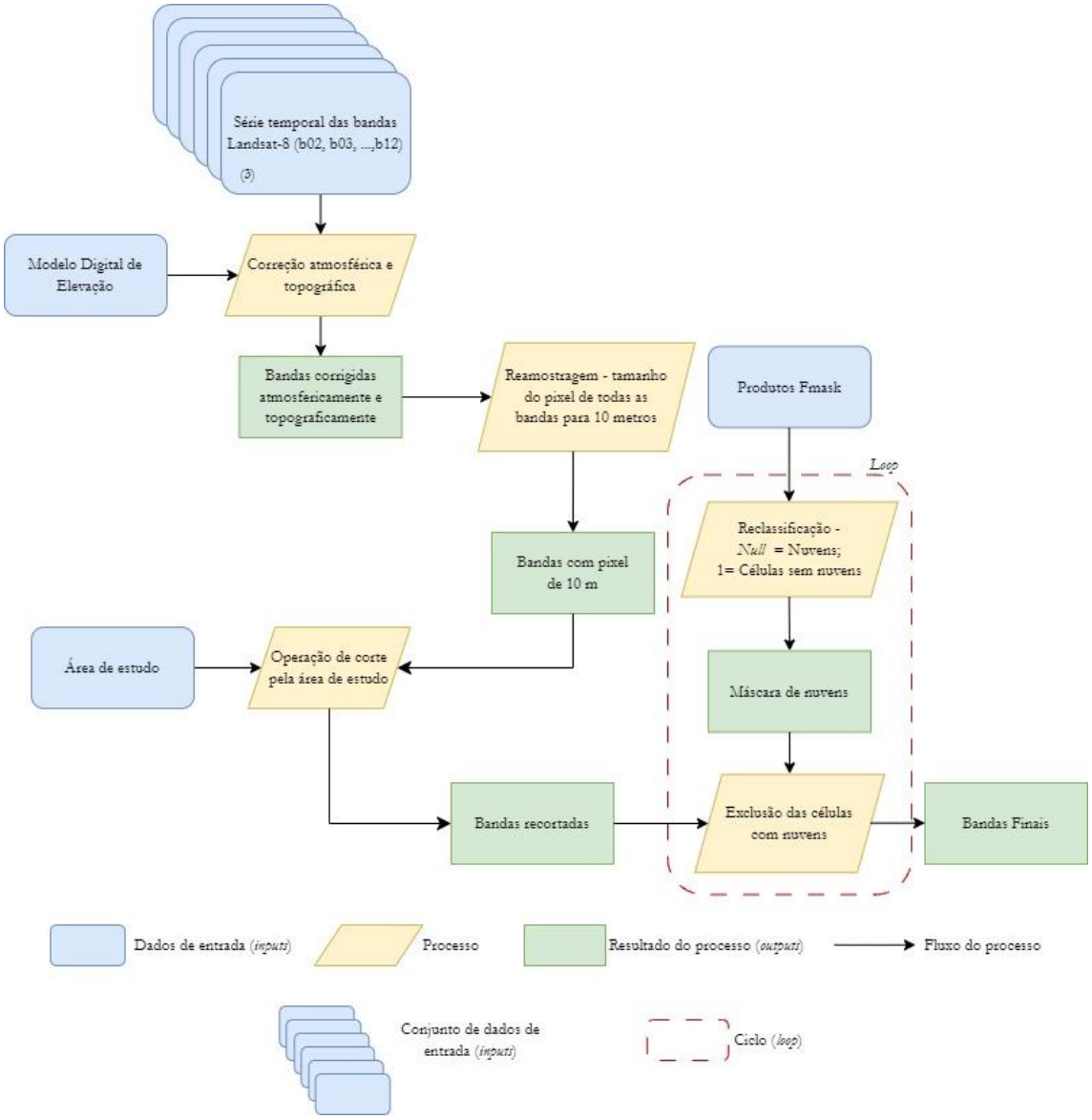


Figura 6 – Fluxograma das fases de pré-processamento das imagens Landsat-8.

Após a execução de correções atmosféricas e topográficas nos dados de imagens Landsat-8, assim como implementação para os dados Sentinel-2, o método de reamostragem NN repetiu-se às respetivas bandas com o objetivo de transformar o tamanho da célula. Em alusão a referências bibliográficas, no trabalho de Forkuor, *et al.* (2017), verificou-se a reamostragem da resolução Landsat-8 original de 30 metros para 10 metros de resolução a partir da técnica do vizinho mais próximo, assim como os autores do estudo de Williamson,

et al. (2018) recorreram a imagens Sentinel-2 e Landsat-8 para monitorizar a profundidade de lagos supraglaciais na Gronelândia, e ao longo do pré-processamento implementaram o mesmo método de interpolação, para a alteração do tamanho dos pixéis das imagens Landsat-8 para 10 metros.

Um dos objetivos da reamostragem das bandas Landsat-8 era garantir que, ao reduzir o tamanho do pixel de 30 para 10 metros, o valor do pixel mais comum fosse definido com o valor correspondente do novo pixel da imagem. Nesse sentido, o método de interpolação NN demonstrou a capacidade de alcançar tal propósito. Adicionalmente, além da justificação baseada na bibliografia, a escolha deste método de interpolação também se fundamentou por esta razão adicional.

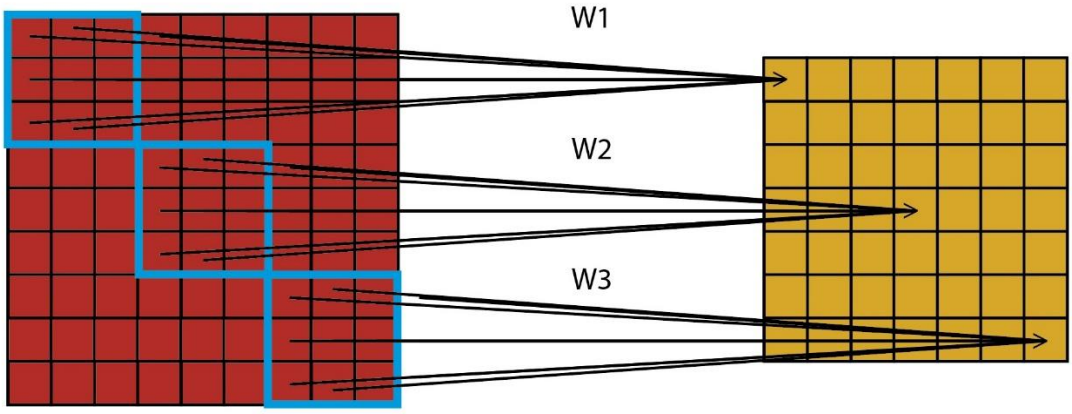


Figura 7 – Ilustração esquemática dos valores dos pixéis de 10 metros subdivididos na imagem Landsat-8 com uma resolução de 30 metros (retângulos azuis) e os valores dos pixéis mais comuns (valores de peso W1, W2 e W3) na imagem reamostrada de 10 metros de resolução (retângulo à direita). Adaptado de: Shao, *et al.* 2019, p. 3.

Posteriormente, realizou-se o corte pela área de estudo, sendo que a presente etapa representa um objetivo central, ao qual estão intrinsecamente associados a sobreposição geográfica e a uniformidade informativa, visando eliminar qualquer incompatibilidade entre os dados e possibilitar a adequada comparação e combinação das diversas imagens.

Na criação de máscaras de nuvens, estas foram mais uma vez estabelecidas de forma automática através do *software* Fmask para a identificação de nuvens, sombras de nuvens e neve nas imagens Landsat-8. Dada à inexistência de produtos SCL para as imagens, como acontece nos dados Sentinel-2, bem como outros fatores adjacentes, foi necessário circundar as adversidades nas imagens, relacionadas com efeitos atmosféricos, de outra forma, para posterior exclusão de células intercetadas a partir da mesma reclassificação realizada às

imagens senitnel-2. Só assim se evitaram células afetadas por nuvens, circunstância pontual na área de estudo definida para esta investigação.

Depois de concretizadas todas as fases de pré-processamento, abrangendo as bandas das duas séries temporais das imagens Landsat-8, estas encontram-se em condições de ser utilizadas na próxima fase metodológica, avançando imediatamente para a etapa de classificação de imagens.

4.3.4 Pré-processamento do Modelo Digital de Elevação (4)

Numa abordagem inicial, conforme foi amplamente adotado nos dados das diversas imagens de satélite, foi essencial recorrer à reamostragem do tamanho original das células do MDE para 10 metros, visando a harmonização com as características das imagens Sentinel-2. Ao contrário, do que se sucedeu com as imagens de satélite, no caso do MDE, diversas dúvidas surgiram relativamente ao método de interpolação mais indicado para aplicação. Nesta etapa, era crucial a preservação integral dos valores correspondentes às elevações originais do modelo, e, portanto, a seleção demandou uma abordagem cuidadosa.

Neste sentido foi realizada uma análise aprofundada, relativamente a dois métodos de reamostragem aplicados ao MDE, com o propósito de determinar a interpolação que melhor preservava as informações das elevações nas diferentes áreas da ilha Terceira.

Esta ideia nasceu da falta de bibliografia de desenvolvimento e definição de métodos de interpolação orientados para a adequada reamostragem de informações sobre a elevação. Para tal, nos subcapítulos seguintes serão demonstrados e caracterizados os resultados desenvolvidos pelos dois métodos de interpolação, acompanhada das conclusões retiradas dessa análise. Essa investigação, encontra-se na parte seguinte do documento, nomeadamente na subsecção enumerada por 4.3.4.1, no entanto a análise dos métodos de pré-processamento aplicados ao MDE, continua a ser detalhada nesta secção.

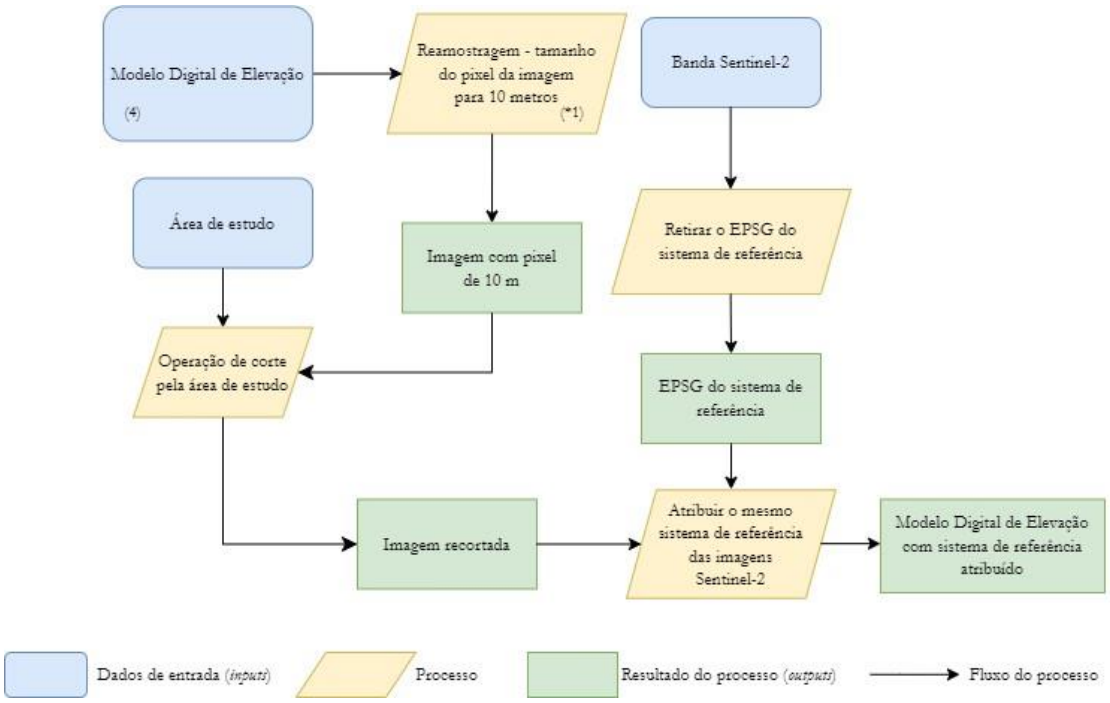


Figura 8 – Fluxograma de descrição das fases de pré-processamento do MDE. (*1) Diferenças entre os métodos *nearest* e *bicubic* e comparação com a proximidade à realidade.

Após a definição do método de interpolação e a aplicação da reamostragem das células para 10 metros, segue-se novamente, uma operação de corte pela área de estudo, assim como ocorreu com as imagens de satélite. Esta promoveu a intenção de obter informação altimétrica apenas para a ilha Terceira. Ademais, trata-se de uma operação pertinente, para igualar a extensão de área dos dados das respetivas imagens de satélite, de forma a garantir uma análise comparativa abrangente e precisa, possibilitando uma avaliação informativa consistente.

Dada a desigualdade entre o sistema de coordenadas original do produto MDE e das restantes imagens de satélite, foi necessário implementar uma etapa de alteração e atribuição de sistema de referência espacial coincidente. Neste sentido, utilizou-se uma banda de Sentinel-2 como referência, para a transformação da projeção orientada à europa do MDE, coordenadas ETRS89 LAEA para WGS84 UTM zona 26N, sistema orientado para o território açoriano. Mais uma vez, a definição do mesmo sistema de coordenadas é imprescindível na garantia de qualidade, precisão e interoperabilidade das informações, acabando por permitir uma análise confiável.

4.3.4.1 Diferenças entre os métodos *Nearest* e *Bicubic* aplicados no MDE: comparação da proximidade à realidade (*1)

Nesta secção do documento, são discutidos e apresentados os diferentes métodos implementados no processo de reamostragem do MDE, com o intuito de avaliar qual dos dois procedimentos desencadeou um melhor desempenho e aperfeiçoamento no rigor das elevações em relação à realidade. Esta fase foi de extrema importância, uma vez que o EU-DEM v1.1 foi descarregado da sua plataforma num sistema de referência distinto das imagens Sentinel-2 e uma resolução espacial de 25 metros. O principal objetivo da reamostragem do MDE foi conciliar o tamanho do pixel para uma resolução espacial de 10 metros, no qual testaram-se os dois métodos: NN e *Bicubic*.

Ao contrário dos pré-processamentos executados para as diferentes imagens de satélite, da mesma forma que as reamostragens dos dois métodos aplicados ao MDE, executados com recurso à programação de ferramentas, toda a fase de análise foi conduzida de forma manual. Para uma compreensão exata dos resultados produzidos por cada método de reamostragem, *Bicubic* e NN, utilizou-se o ArcGis Pro. No pacote de ferramentas do respetivo *software* fez-se auxílio da ferramenta *raster calculator*, para o cálculo da diferença absoluta entre os MDEs produzidos pelos métodos em questão. Este processo permitiu dar início à análise integral das técnicas de interpolação, com o propósito de compreender as disparidades entre os valores de altitude de cada modelo produzido, e localizar as discordâncias da elevação na extensão da área de estudo.

Para análise das principais variações elevativas entre os modelos, foram selecionadas áreas de análise nas quais a diferença das informações era evidente, a fim de determinar a reamostragem com melhor preservação dos valores de elevação originais. O detalhe dos valores de altitude é fundamental para a seleção do método aplicado, uma vez que se pretendia obter o maior rigor possível. Todas as informações correspondentes às áreas de análise com diferenças aparentes foram consideradas, e o estudo auxiliou da análise conjunta de um *Basemap*, Google Earth Pro e estudo de valores mínimos e máximos, para o melhor entendimento acerca do relevo encontrado nas áreas em questão.

A análise das técnicas NN e *Bicubic* aplicados no MDE, tiveram também em consideração as particularidades e os princípios básicos de cada método, descritas nas secções anteriores, para o reconhecimento dos valores de altitude definidos em pontos específicos da extensão da

área de estudo. Estas análises de resultados produzidos pelos métodos de interpolação serão apresentadas no subcapítulo seguinte.

4.3.4.1.1 Interpolação *Bicubic* e *Nearest Neighbor*

O presente subcapítulo, tem por base a análise e discussão do desempenho entre o método *Bicubic* e NN, com o princípio de conhecer a interpolação que melhor conservou as características altimétricas do relevo da área de estudo. Ao longo dos próximos pontos, serão discutidos resultados produzidos, valores, semelhanças e diversidades, entre ambos. Foi necessário recorrer ao *software* SIG, ArcGis Pro, para visualização e análise. Este SIG foi essencial para o cálculo da diferença absoluta, executado com o acesso à ferramenta *raster calculator*, aplicando a subtração entre os modelos criados pelos diferentes métodos, a fim de identificar as regiões onde os valores são concordantes e as áreas de diferenças. Na Figura 9, é possível verificar o resultado da aplicação da ferramenta *raster calculator*, em ambiente ArcGis Pro.

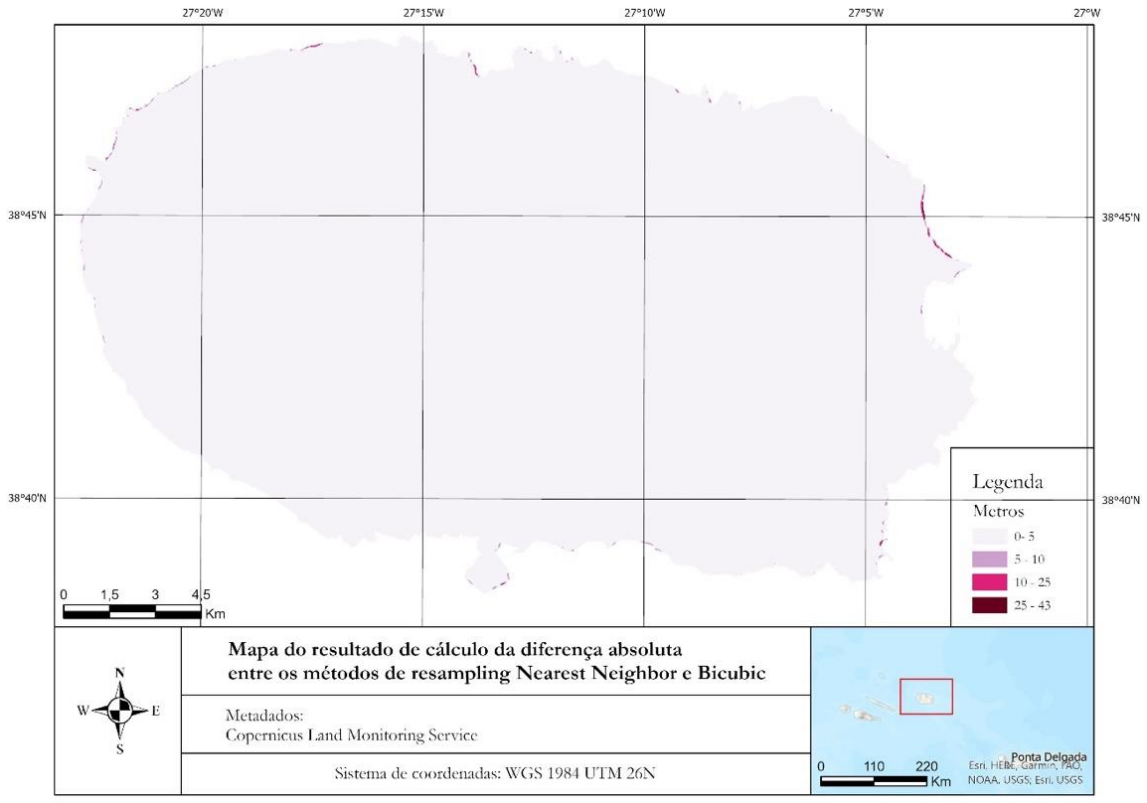


Figura 9 – Mapa da diferença absoluta das elevações pelos dois métodos de interpolação.

De um modo geral, a partir da análise cálculo da diferença absoluta entre os valores de altitude obtidos pelo método NN e *Bicubic*, é evidente uma concordância entre as informações de elevação para quase toda a extensão geográfica. Todavia, cabe ressaltar que existem variações entre os valores, mesmo que não sejam evidentes na parte mais interior da ilha Terceira. A partir da Figura 9 é possível detetar as anomalias entre os dois métodos que, curiosamente, se limitam apenas às extremidades da extensão geográfica. A análise da diferença máxima entre os valores de altitude dos métodos de reamostragem foi feita, e um registo de apenas 43.7 metros foi encontrado.

O presente fenómeno de variação das elevações entre os métodos de reamostragem pode estar atribuído à intensidade de suavização aplicada pelas diferentes técnicas, e com referência para a Figura 9, esta é mais sentida nas áreas litorais da área de estudo. Nestes locais, os valores de elevação determinados pelo MDE original podem acabar por ser suavizados para altitudes mais moderadas, resultando em altitudes intermédias relativamente à realidade. Neste sentido o respetivo exercício é conduzido, com a identificação das áreas onde as variações são claramente evidentes, acompanhada de uma análise para determinar a interpolação que aplicou uma maior suavização nos valores de elevação.

A análise do mapa da Figura 9 será iniciada do lado oeste da ilha, com uma avaliação extensiva ao longo das áreas de borda onde as diferenças são facilmente identificadas, avançando pela parte norte, oeste e sul, com a intenção de contornar toda a área de estudo. Objetiva-se realizar uma análise abrangente de diversas áreas em pontos diferentes da extensão geográfica, com o propósito de contemplar uma ampla gama de possibilidades e exemplos, visando assegurar a máxima precisão e credibilidade do estudo.

4.3.4.1.1.1 Áreas de análise

Para o avanço da análise das variações dos valores de elevação obtidos a partir dos métodos de reamostragem, selecionaram-se seis pequenas áreas em diferentes pontos da ilha. A tarefa consistiu na interpretação da inconsistência entre as altitudes adquiridos a partir do cálculo da diferença absoluta entre os dois métodos de reamostragem. Em auxílio, foram analisados os valores mínimo e máximo identificados em cada método de reamostragem para as diferentes áreas, com o intuito de aprofundar o reconhecimento relativo à melhor proximidade da morfologia do relevo. A análise tem como finalidade alcançar este objetivo, uma vez que o detalhe das elevações pode ser benéfico na definição de determinadas classes,

e, consequentemente, na geração de mapas temáticos de exatidão superior. São apresentadas, na Figura 10, o grupo de áreas selecionadas, com a sua localização exata nos vários pontos da extensão geográfica, para a avaliação dos valores de altitude e desenvolvimento de conclusões.

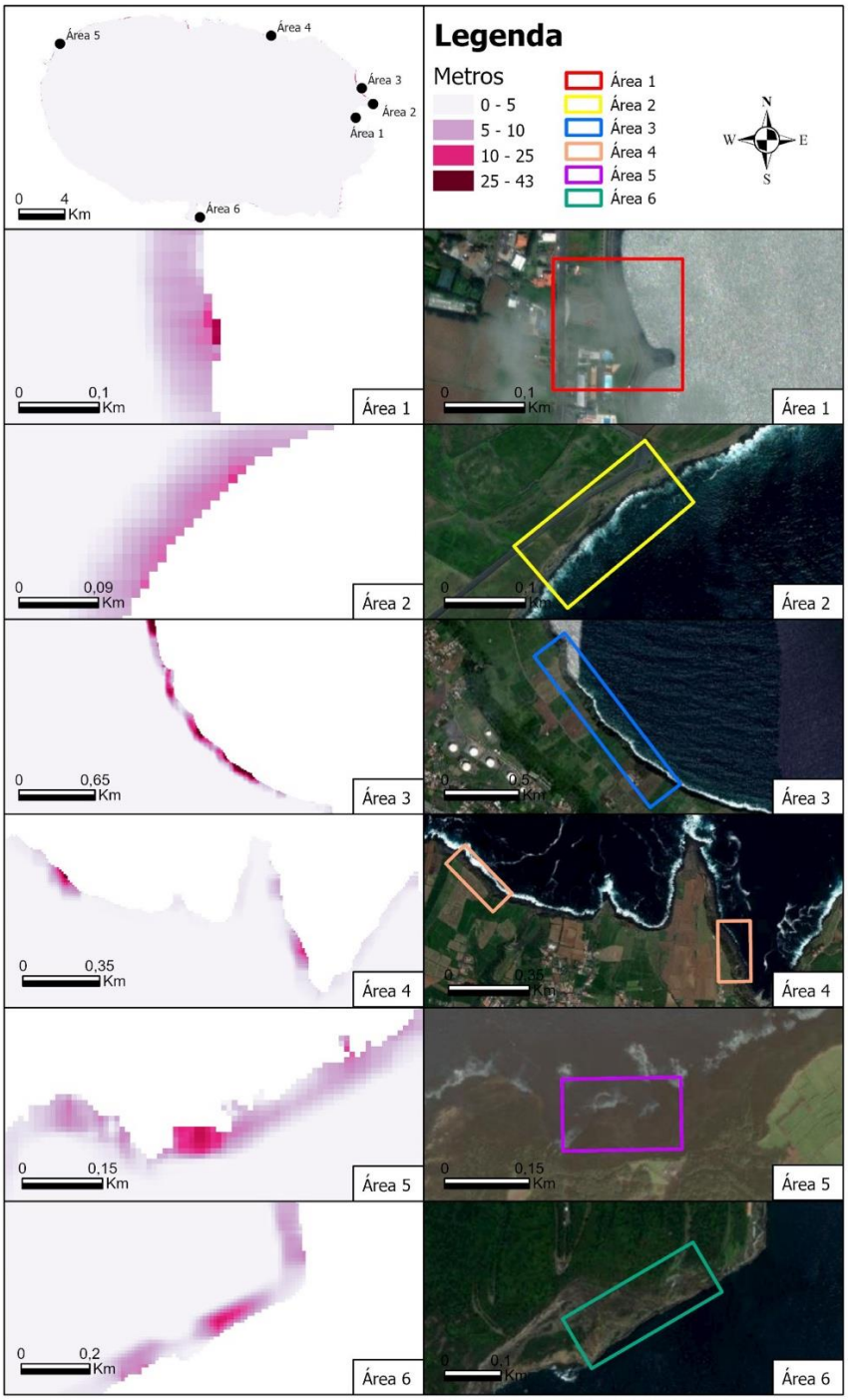


Figura 10 – Enquadramento das áreas de análise para comparação dos valores de elevações.

1ª Área: É importante notar que esta se localiza na zona ocidental da ilha Terceira, em grande proximidade ao Miradouro da Boavista. Trata-se de uma área costeira caracterizada por uma baía, com a inclusão de praias, como a Praia da Riviera. Além disso, é importante salientar que esta 1ª Área se encontra muito próxima ao oceano, o que lhe confere uma morfologia costeira predominantemente baixa. Ao longo da baía, a proximidade ao mar é evidente, conferindo valores de elevação relativamente baixos.

De acordo com o mapa referente à variação absoluta dos valores de elevação entre os dois métodos de reamostragem aplicados na 1ª Área, estes apresentam uma diferença de valor máximo de 17.41 metros. É digno de nota que se verifica uma certa diferença entre os valores apresentados pelos dois métodos, sendo este motivo de análise.

2ª Área: Localiza-se próxima da primeira área de estudo, porém mais a norte. Esta área alberga o Miradouro da Ponta da Má Merenda, uma região ilustre pela sua paisagem costeira acidentada, caracterizada por várias formações rochosas com vista para o oceano. Em comparação com a primeira área, esta segunda apresenta uma altitude superior, caracterizada por um relevo de arribas de inclinação acentuada.

No que concerne às diferenças entre as duas técnicas de reamostragem, reconhece-se uma certa limitação, onde o valor máximo da variação é de apenas 13 metros. As diferenças entre os resultados dos dois métodos são relativamente sentidas, indicando que existe uma discordância entre os métodos.

3ª Área: Considerada uma das regiões mais interessantes, devido à evidente diferença entre os dois métodos. Localizada a norte da localidade da Praia da Vitória, esta costa litoral do lado este da ilha, é caracterizada por uma região escarpada, com arribas altas. Estas elevações são responsáveis por uma paisagem única e impressionante, com falésias íngremes que se estendem ao longo da costa. Curiosamente, não possui vias de circulação próximas da linha de costa, ao contrário das áreas anteriores analisadas. Esta particularidade sugere que se trata de uma linha de costa muito recortada e perigosa, com arribas extremamente elevadas, podendo apresentar risco para a população.

Observou-se que os resultados obtidos correspondem à morfologia do relevo esperada. Esperavam-se valores de elevação muito altos, sendo que as altitudes acima dos 100 metros confirmaram esta expectativa. Foi possível perceber, através do mapa da variação absoluta, que a maior diferença entre os valores de elevação foi registada nesta área, com uma diferença de 36 metros.

4ª Área: Constitui elevada relevância para a análise comparativa dos diferentes MDEs de elevação produzidos pelos métodos NN e *bicubic*. A diferença entre os valores de elevação registados é evidente, e esta 4ª Área caracteriza-se por uma costa alta. Situada nas proximidades da Rota do Mar, destaca-se por uma costa litoral diversificada, embora, predominantemente, composta por arribas ao longo de toda a sua extensão. Esta área oferece acesso a praias isoladas, mas teve-se uma especial atenção para as diferenças de elevação, conhecidas por amplas irregularidades.

No que diz respeito à disparidade dos valores de elevação obtidos pelos diferentes métodos de reamostragem, obteve-se uma diferença de 22 metros, um valor significativo. Evidencia-se, portanto, que não existe concordância entre os dois métodos aplicados. Conforme previsto, esta 4ª Área descreve uma costa recortada com arribas inclinadas, de altitudes consideravelmente elevadas.

5ª Área: Situa-se na parte mais oeste da ilha, e foi selecionada para análise, uma vez que se procurou uma ampla representatividade de resultados, como forma de comparar semelhanças ou diferenças com as outras áreas analisadas. Localiza-se no Miradouro do Raminho e apresenta características típicas da área de estudo, como uma costa litoral de falésias e arribas de altitudes acentuadas.

A variação entre os valores de altitude obtidos pelos métodos está claramente presente, e, portanto, foi necessário determinar a melhor interpolação. O cálculo da diferença absoluta resultou num valor de 16.66 metros. É considerado como um valor médio, o que indica que os métodos de reamostragem apresentaram uma certa diferença, não deixando de ser relevante para análise.

6ª Área: Este último ponto localiza-se na parte mais a sul da área de estudo. Foi relevante adicionar à análise, uma vez que foi importante ampliar a amostra e os exemplos utilizados, a fim de verificar se mantêm congruentes com as áreas previamente examinadas. Esta 6ª Área está localizada no Monte Brasil. A costa litoral da área apresenta características de um relevo extremamente acidentado, composto essencialmente por escarpas elevadas. Esta morfologia litoral é influenciada pela Caldeira, a grande depressão vulcânica desenvolvida pelo Monte Brasil, caracterizada por um litoral excessivamente elevado e irregular.

Sobre a variação do valor absoluto das altitudes que distingue as duas técnicas de reamostragem, 15.5 metros é a elevação que os separa. Semelhante à área analisada anteriormente, esta 6ª Área apresenta um valor significativo e comparativamente mais baixo,

no entanto, é sempre relevante estudar os mais variados exemplos, a fim de alcançar o objetivo proposto.

Considerações do exercício de análise dos diferentes métodos de interpolações para a reamostragem do MDE

Os resultados das elevações obtidas a partir da execução dos dois métodos de reamostragem, NN e *bicubic*, integrados no estudo, mostraram ser possível determinar o melhor desempenho na precisão dos valores de altitude, em função das verdadeiras características do relevo da área de estudo. A partir dos exemplos demonstrados, é viável afirmar que as desigualdades entre os métodos de reamostragem localizam-se essencialmente em pontos litorais da ilha Terceira, especificamente em regiões de costa alta, caracterizada por um relevo constituído por arribas de altitudes e inclinação extremamente sentidas. O único exemplo que retratou uma costa litoral mais baixa foi a 1ª Área de análise, com uma linha de costa muito próxima do nível do oceano e as elevações assinaladas remontaram para altitudes mais baixos.

De uma forma geral, os valores obtidos pelo método *bicubic* ostentaram valores mínimos mais altos, em comparação com o método NN. Reconheceram-se concordâncias na altitude máxima, onde os valores foram repetitivamente idênticos, ou muito próximos com uma variação insignificante de pequenos metros. Sintetizou-se que o leque de valores de elevação foi consideravelmente maior para o método de reamostragem NN conferindo um detalhe da informação superior, em comparação com a interpolação *bicubic*. Uma vez que o MDE produzido a partir da técnica NN*,* o método atribui o novo valor informativo da célula reamostrada através da informação do vizinho mais próximo. Esta técnica conduziu a valores de altitude tecnicamente mais variados, dada à utilização de uma amostra de vizinhos mais reduzida. Para os valores de elevação desenvolvidos pela técnica de reamostragem *bicubic,* conclui-se que a utilização de uma amostra de valores vizinhos superior, de 16 células, acaba por determinar altitudes menos diversificadas, resultando em valores de altitudes dissolvidas nas células reamostradas.

Ainda assim, agregando as peças sobre as elevações obtidas por cada método de reamostragem e as características das diferentes áreas analisadas, reconhece-se o melhor desempenho da técnica executada pelo método NN. Em contrapartida o método *bicubic* realizou uma maior suavização da realidade, consumando altitudes mais diluídas.

4.4 Filtragem dos Dados de Treino (5)

No que concerne ao presente subcapítulo, é imprescindível afirmar que esta fase da investigação constitui, uma base fundamental para o desenvolvimento integral do trabalho e produção dos dados de treino necessários à classificação de imagens de satélite e, consequentemente, elaboração dos mapas de ocupação do solo com maior ou menor exatidão global.

Esta etapa do trabalho abrange as fases de filtragem de dados de treino, fundamentada na metodologia estabelecida no artigo de Fonte, *et al.* (2020). Este é o ponto de partida para a introdução de inovações, por meio de novas abordagens de filtragem, para compreender de que maneira a introdução de novos índices radiométricos e definição de novas regras de corte para individualização das características espetrais das classes podem potencializar os resultados alcançados na classificação.

Posto isto, deve dizer-se que o desenvolvimento deste exercício, privilegiou da conceção e utilização de procedimentos programados, facilitando o período de tempo despendido em *softwares* SIG, de forma a facilitar a cadeia lógica de geoprocessos, eliminando períodos ociosos de preparação de ferramentas. Estes períodos são caracterizados pela seleção de parâmetros de entrada e da saída de ferramentas. Ademais, como nem todos os procedimentos auxiliaram da programação de ferramentas SIG, perde-se a condicionante da qualidade e disponibilidade de tecnologias SIG, e assim, a carência de *tools* para determinados resultados pretendidos, não condenam todo o processo, sendo possível superar as restrições dos *softwares*.

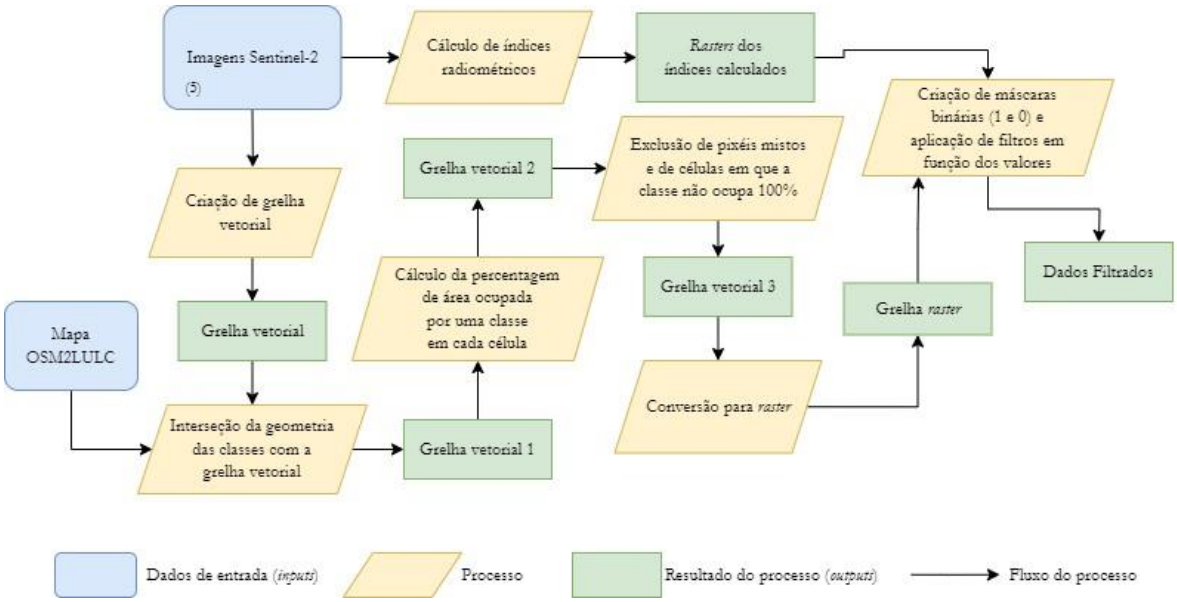


Figura 11 – Fluxograma da fase de filtragem e criação de conjuntos de dados de treino.

No âmbito das tarefas apresentadas, o fluxograma da metodologia executada para o processo de filtragem dos dados sugere que, numa primeira fase, com recurso a imagens Sentinel-2 com bandas reamostradas para resolução de 10 metros, é exequível direcionar a fase de produção de dados filtrados por duas vias de processamento, que, no final se encontrarão e serão conjugados para obter resultados. As duas vias de processamento são dívidas pelo formato de dados, tendo em consideração um ciclo, direcionado a dados vetoriais e outro orientado a ficheiros de formato *raster*.

Tendo em consideração a ordem das fases de concretização, a explicação metodológica inicia-se pelo ciclo de trabalho focado nos dados vetoriais, nomeadamente na criação de grelhas vetoriais. Esta etapa, descreve a produção de grelhas em formato vetorial compostas por elementos quadrados, sem qualquer atributo associado, produzidos a partir das bandas *raster* da imagem Sentinel-2 com resolução de 10 metros, delimitando individualmente cada célula da extensão geográfica. A produção de grelhas vetoriais a partir de imagens *raster*, acarreta vantagens, uma vez que tornam o esforço computacional menos elevado, em comparação com a manipulação de imagens *raster*, ficheiros consideravelmente mais pesados. Ademais, facilitam a edição e atualização das informações, permitindo análises geospaciais mais complexas, providenciando o cálculo de áreas das diferentes geometrias, sobreposição e combinação de dados.

Estes fatores tornaram-se imprescindíveis na etapa seguinte, onde foi consumada a interseção das geometrias das classes, provenientes do mapa vetorial OSM. A respetiva interseção, proporcionou a correspondência das diferentes classes em determinadas células da grelha, permitindo a análise relativa à ocupação de classes informativas dentro da grelha. Possibilita ainda o posterior cálculo de estatísticas de polígonos integrados na grelha, como o número de elementos intersetados no interior das células.

O seguinte passo enfatiza os processos aliados à grelha vetorial, e dedica-se, respetivamente, ao cálculo dos correspondentes elementos produzidos da interseção realizada entre a grelha vetorial e as classes, com o propósito de contabilizar as ocorrências de classes no interior de cada célula e a percentagem de ocupação representada pelas mesmas. É de extrema importância, uma vez que na são selecionadas todas a células, cuja ocupação é considerada como mista, ou seja, quando é contabilizada mais do que uma classe no interior de cada quadrícula e ainda classes que não ocupam a totalidade, isto é, classes que não preenchem 100% dos elementos da grelha. Assim, realiza-se uma primeira análise seletiva das informações das classes viáveis, excluindo informações que possam comprometer a confiabilidade dos dados de treino utilizados posteriormente na classificação de imagens.

Após o procedimento de seleção e exclusão de informações comprometedoras presentes nas grelhas vetoriais, estas são restauradas para o seu formato inicial. Os formatos dos ficheiros vetoriais, previamente sujeitos a diversas fases de processamento, passam por uma etapa final relacionada à conversão para *raster*. Essa etapa é crucial para a posterior criação de máscaras binárias e aplicação de filtros.

No segundo ciclo, assim como o primeiro, utilizam-se como dados de entrada o conjunto de bandas das imagens Sentinel-2 reamostradas para 10 metros, no entanto, os dados não se submeteram a processos de conversão para formato vetorial e, portanto, sendo trabalhados no formato original *raster*.

Este ciclo de trabalho, destina-se inteiramente, a processos de cálculo de índices radiométricos realizados por meio de fórmulas matemáticas que combinam valores de refletância de diferentes bandas espetrais da imagem, fornecendo informações úteis para o melhor entendimento das características da superfície terrestre. É relevante salientar que o cálculo dos índices radiométricos abrangeu a totalidade das séries temporais das imagens Sentinel-2 utilizadas. Esta abordagem justifica-se pelos valores espetrais registados nas diferentes datas das imagens, que carregam informações valiosas para a definição das classes e descrevem as suas variações informativas em períodos do ano distintos.

Após a concretização dos dois ciclos de trabalho, em que, por um lado, os resultados foram obtidos por meio dos processos executados nas grelhas vetoriais, seguido da conversão das informações para formato *raster*, e por outro lado, pela realização do cálculo de índices radiométricos, foram estabelecidas condições de produção de máscaras binárias e aplicação de filtros com base nos valores obtidos a partir dos índices radiométricos.

Relativamente às máscaras binárias, estas consistem em *rasters* compostos por valores binários, representados pelo valor 0 e 1, onde cada valor indica a inclusão ou exclusão de determinados pixéis da imagem. O processo de filtragem dos dados de treino baseou-se nos valores obtidos por meio do cálculo dos índices radiométricos. Este processo levou a cabo regras de filtragem especificas para cada classe, com foco nos valores que cada classe deveria apresentar em determinado índice radiométrico, bem como na frequência com que esses valores deveriam ocorrer ao longo da série temporal nas imagens captadas pelo Sentinel-2. Aquando da implementação de regras de filtragem, as máscaras binárias entram no processo, atribuindo o valor 1 quando um pixel de uma classe atende às regras de filtragem, definindo que esse pixel deve ser considerado como uma célula a ser integrada no conjunto de dados de treino. De outra forma, quando as regras não são atingidas, o valor 0 é atribuído, resultando na exclusão do respetivo pixel dos dados de treino.

Esta abordagem foi adotada com o propósito de realizar uma classificação mais exata e eficiente, a partir de uma filtragem dos dados de treino correta. Todavia, no subcapítulo seguinte serão evidenciados os valores limite definidos para cada índice radiométrico, considerando as regras estabelecidas em Fonte, *et al.* (2020).

4.5 Definição de regras de filtragem para índices radiométricos

Para a implementação dos processos de filtragem dos dados de treino, onde são definidas regras para os valores limites das classes em função dos índices radiométricos calculados, torna-se legítimo utilizar diferentes abordagens. Para além da alternativa inerente à utilização dos índices radiométricos para filtrar os dados de treino com base na identificação de valores limites implícitos na literatura, tendo como referência as regras estabelecidas no artigo Fonte, *et al.* (2020), é igualmente relevante comparar a performance e o desempenho dos respetivos processos de filtragem, a partir de uma estratégia inovadora, envolvendo uma exploração metodológica rigorosa, com o intuito de definir novas regras para filtrar os dados de treino.

Assim, para o desenvolvimento desta dissertação, e fazendo alusão ao respetivo subcapítulo que caracteriza a definição dos limiares de corte para cada índice radiométrico, concebe-se diferentes formas de se realizarem as filtragens, considerando os critérios pré-estabelecidos no estudo de Fonte, *et al.* (2020) e um trabalho investigativo procedido de forma manual, com recorrência a um *software* em ambiente SIG, para determinar novas regras de filtragem, a partir da análise histogramas com representação do comportamento das classes em determinados índices radiométricos.

Atendendo às regras de filtragem implementadas na literatura de referência, os limites definidos utilizaram uma metodologia baseada na interpretação visual e análise de histogramas por classe produzidos para três índices de cada imagem de satélite, concretamente NDVI, NDWI e NDBI. Ainda assim, analisaram a média e o desvio padrão dos índices por classe nas áreas que tinham sido selecionadas para o caso de estudo. Propuseram para as classes de 1 a 5 os valores limites definidos nos NDVI e NDWI teriam de ser respeitados em todas as imagens de satélite, caso que não ocorreu com as classes de 6 a 8, onde seria necessário pelo menos um conjunto de imagens que respeitasse as regras dos valores limites para que se considerasse como classe para os dados de treino. Por fim, utilizaram o NDBI apenas para a classe 1. A informação detalhada sobre as regras dos valores limites para cada classe, implementada em Fonte, *et al.* (2020), pode ser analisada na Tabela XIV.

Tabela XIV – Condições de filtragem de dados de treino com valores limites para os índices NDVI, NDW e NDBI. Fonte: Fonte, *et al.* 2020, p. 12.

Classes	NDVI/*Images*	NDWI/*Images*	NDBI/*Images*
1. *Artifiacial surfaces*	<0.3/*all*	<0.0/*all*	>0.0/*at least one*
2. *Agricultural areas*	>0.3/*all*	<0.0/*all*	-
3. *Herbaceous vegetation*	>0.3/*all*	<0.0/*all*	-
4. *Forest areas*	>0.3/*all*	<0.0/*all*	-
5. *Shrublands*	>0.3/*all*	<0.0/*all*	-
6. *Open spaces with little or no vegetation*	>0.0/*at least one*	<0.0/*at least one*	-
7. *Wetlands*	>0.0/*at least one*	<0.0/*at least one*	-
8. *Water bodies*	<0.3/*at least one*	>0.0/*all*	-

Objetiva-se como intuito apresentar um conjunto de melhorias, relativamente à base metodológica implementada no artigo de Fonte, *et al.* (2020) e, portanto, a definição de uma

possível forma de realizar a filtragem dos dados de treino, é um dos pontos de peso a ser considerado, tornando-se uma etapa altamente influente.

A estratégia implementada para alcançar a segunda forma de aplicação das regras de filtragem, privilegiou do apoio de um *software* SIG, com ferramentas para armazenar, analisar, manipular e visualizar dados geográficos. Neste sentido, recorreu-se ao ArcGis Pro para consumação dos objetivos da análise, pelo maior conhecimento e por se revelar mais polivalente e confiável, disponibilizando vários materiais para processamento de dados, especialmente no processo de segmentação. O respetivo processo, envolve a divisão de imagens em vários fragmentos ou áreas de pequenas dimensões tendo por base as características espetrais, cor, textura forma e ainda, outras propriedades presentes nas imagens. O objetivo da segmentação é concentrar pixéis ou criar conjuntos, ponderando as diferentes características, de forma a identificar e separar individualmente diferentes elementos nas imagens representadas por classes informativas, como vegetação, cursos de água, edifícios, herbáceas e os restantes tipos de ocupação da superfície.

A execução da técnica de segmentação a seleção do respetivo *software*, deve-se ao fato da plataforma oferecer um método de processamento de segmentação de imagens de satélite, designada por PCA. Esta técnica, baseada na abordagem limiar (*thresholding*) é geralmente utilizada para a redução da dimensionalidade dos dados, melhorar a interpretação e destacar padrões nas imagens. No caso específico, a identificação de características, foi imprescindível para o exercício, uma vez que facilita o reconhecimento de informações relevantes. Para o funcionamento da ferramenta, é necessário fornecer como entrada, o número total de bandas que compõem as imagens e o resultado obtido, sendo um *raster* multibanda. É importante ter em conta que o resultado multibanda obtido contém o mesmo número de bandas dadas como *input*. No entanto, a variação informativa encontra-se apenas nos primeiros três *rasters*, enquanto os restantes repetem a mesma informação ou não agregam mais dados relevantes.

Antecedendo a execução do exercício, procedeu-se à delimitação de uma área de teste, Figura 12, para a qual se preparou um extrato de quatro imagens Sentinel-2 com datas distribuídas ao longo do ano de 2022, descarregadas gratuitamente a partir da plataforma *Copernicus Open Access Hub* (https://scihub.copernicus.eu/dhus/#/home). A área, situada em Portugal Continental, na região de Lisboa, é maioritariamente urbana, mas agrega também áreas agrícolas, vegetação natural, regiões naturais e zonas húmidas. A escolha desta área deve-se à grande variedade de informações relativas à ocupação do solo.

As bandas utilizadas para o exercício passaram por um momento de pré-processamento, em que se ajustou a resolução espacial para 10 metros, e ainda por um processo de exclusão das células identificadas como nuvens. À semelhança do pré-processamento das imagens Sentinel-2 da ilha Terceira, a preparação dos dados foi executada de forma automatizada.

No desenvolvimento deste exercício, e atendendo ao objetivo pretendido, auxiliou-se de mapas de referência, para uma melhor perceção relativamente à ocupação do solo presente na área de teste, e por se ter revelado essencial para a metodologia implementada. Como informação de referência, utilizou-se a COS 2018 que contribuiu com informações e características da ocupação e uso do solo do território nacional.

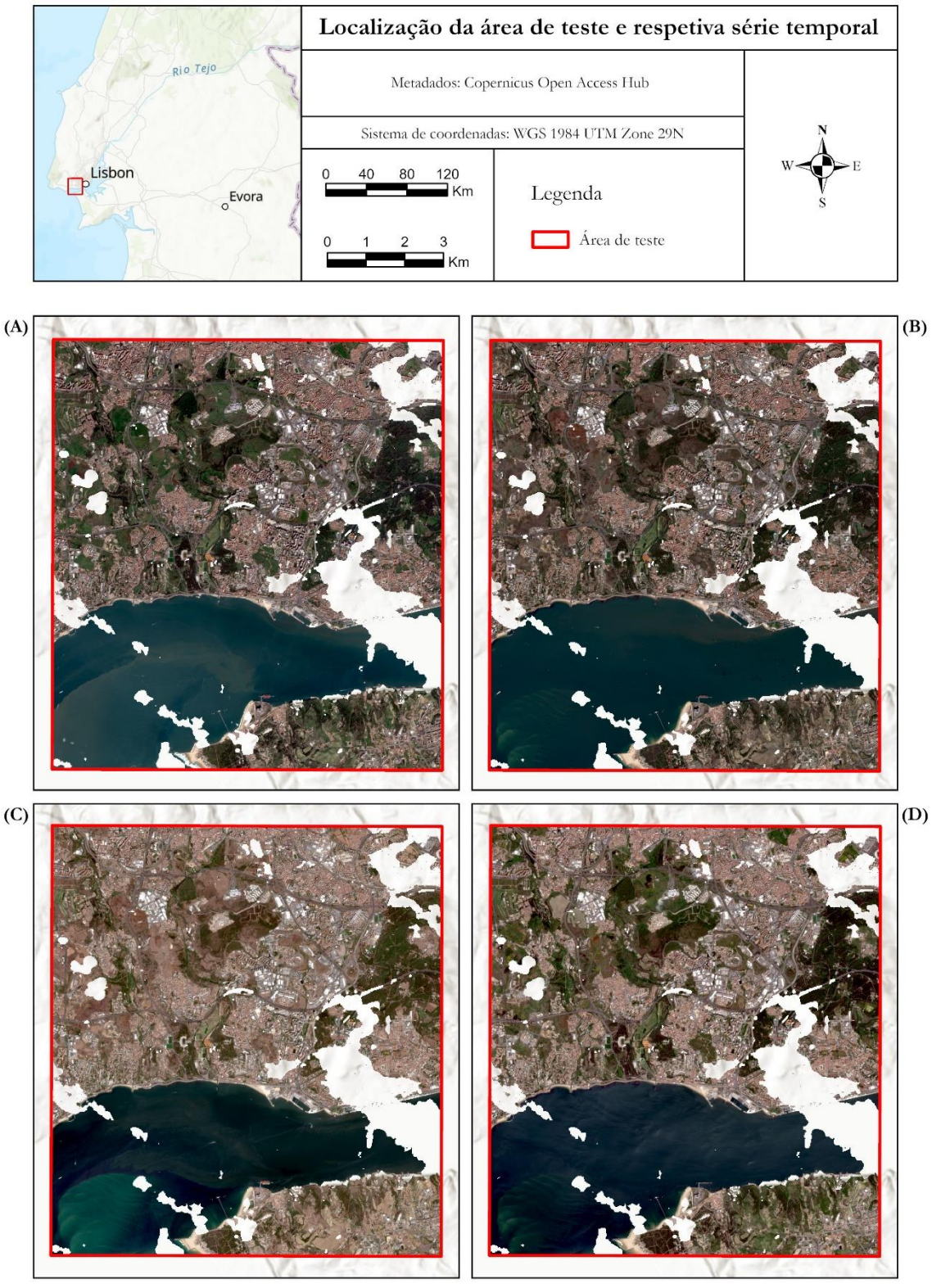


Figura 12 – Localização da área de teste para definição de novas regras de filtragem de dados.

(A) Composição da cor natural, RGB (bandas 4, 3 e 2) das imagens Sentinel-2 para a data 28-02-2022; (B) Data de 01-10-2022; (C) Data de 12-08-2022; (D) Data de 04-05-2022.

É importante salientar que, ao contrário do procedimento efetuado no pré-processamento das imagens Sentinel-2 para a área de teste, auxiliada por meio da execução de programas, toda a respetiva metodologia foi concretizada de forma manual, por meio da utilização de ferramentas no ambiente do *software* ArcGis Pro, virtude da rigorosa interpretação colocada pelo exercício. Considerando as múltiplas abordagens da metodologia, apresentada na Figura 13, foi realizada a harmonização das classes informativas da COS 2018 para a nomenclatura estabelecida na presente dissertação, CLC+ BB, resultando no mapa de classes informativas em formato vetorial. A respetiva tarefa de harmonização entre as nomenclaturas mencionadas pode ser analisada em anexo, no Anexo B (Anexo B1).

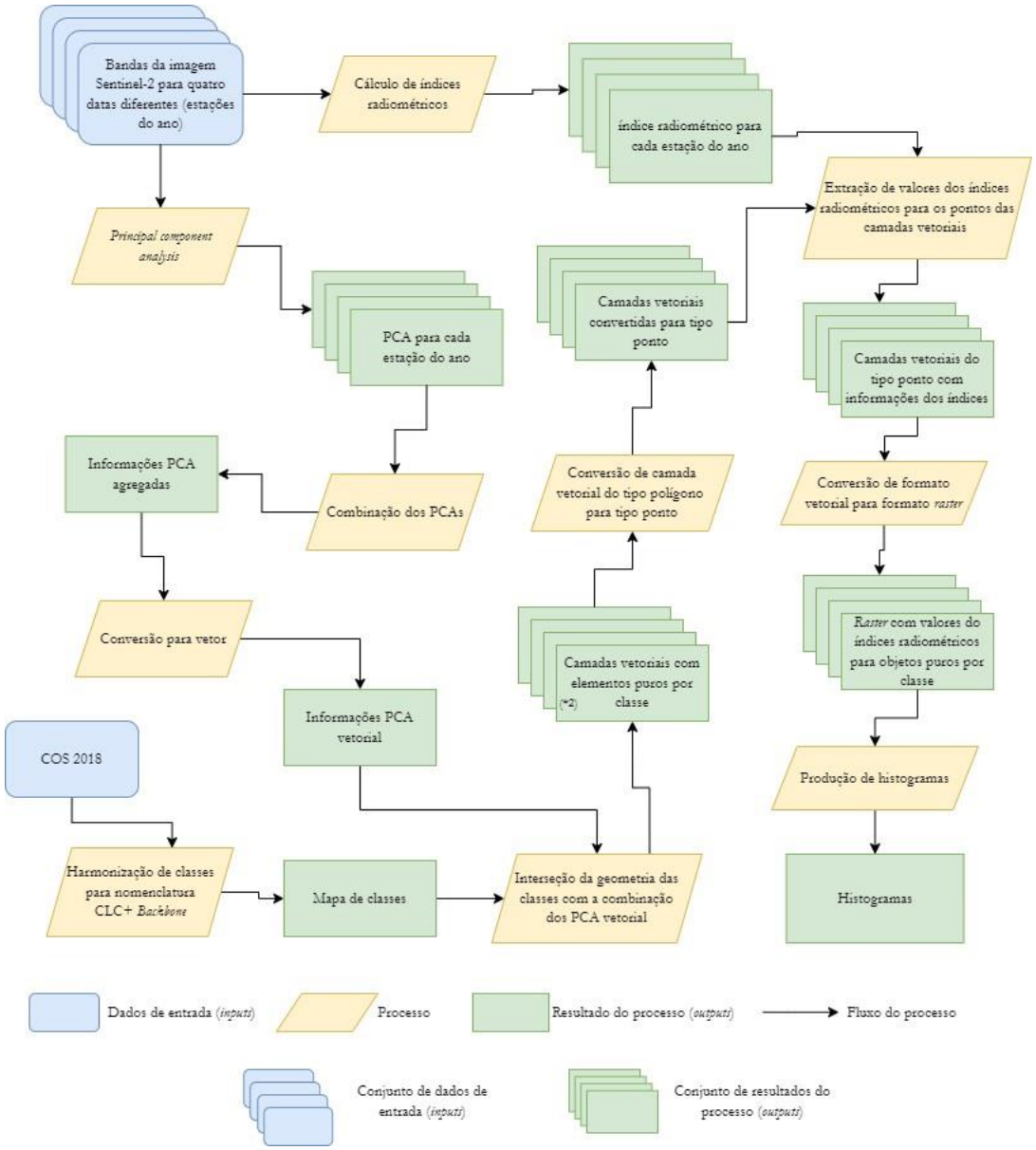


Figura 13 – Etapas de definição de valores limite para índices radiométricos com produção de histogramas. (*2) Processo de seleção de células das classes com valores espetrais puros.

Relativamente à série temporal de imagens Sentinel-2, pré-processadas anteriormente, estas foram divididas por duas fases operacionais distintas. Tendo em consideração cada série temporal, procedeu-se à combinação de bandas, utilizando a ferramenta *raster calculator* para o cálculo de expressões matemáticas, a fim de obter índices radiométricos. Em harmonização com a literatura, os índices radiométricos produzidos, aliaram-se aos índices calculados em Fonte, *et al.* (2020), bem como outros, visto que a integração de mais índices enquadra-se num dos objetivos definidos, podendo contribuir para uma estratificação mais profunda das classes, resultando, numa fase de filtragem de dados de treino mais detalhada. As diferentes bandas das séries temporais das imagens Sentinel-2 foram submetidas ao processamento de segmentação de imagem. Para tal, recorreu-se à ferramenta PCA do ArcGis Pro, para individualizar os elementos das imagens em várias classes temáticas, tendo em consideração as características encontradas.

A partir dos resultados da execução da ferramenta PCA, procedeu-se à combinação das informações contidas nos quatros *rasters* multibandas. Para o efeito recorreu-se à ferramenta *combine*, a fim de agregar todos os pixéis das camadas *raster*, para a produção de uma nova camada onde são incluídos todos os valores únicos dos *rasters* dados como entrada. Seguindo a sequência lógica das etapas da metodologia, o *raster* obtido da ferramenta *combine*, segue para uma fase de conversão de formato. Através da ferramenta *raster to polygon*, foi exequível converter as informações do ficheiro em formato *raster* para vetorial. Considera-se como uma etapa imprescindível, tendo em conta que imediatamente a seguir intersetaram-se geometrias das classes do mapa vetorial com as informações relativas à segmentação, integradas na camada vetorial das combinações dos vários PCA. A presente fase acabou por ser pertinente, uma vez que realiza a associação das informações adquiridas pela segmentação ao conjunto de classes temáticas CLC+ BB, do mapa vetorial.

Foram identificadas sete classes representativas: *forest*, *herbaceous*, *partly vegetated or non-vegetated land, sealed surface*, *shrubland*, *water* e *wetlands*. Completada a interseção e adquiridas os vários ficheiros vetoriais para as respetivas 7 classes, a etapa metodológica que se segue acarreta uma análise extremamente rigorosa. Nesta fase, com o auxílio do *basemap*, foi necessário identificar manualmente uma amostra estratificada de polígonos que contêm apenas elementos pertencentes às classes, ou seja, para considerar um elemento por exemplo de urbano, não podia ser identificado qualquer outro tipo de ocupação no interior do atributo da camada vetorial. Para um melhor entendimento sobre o que se necessitou de realizar na presente etapa, segue-se a seguinte Figura 14.

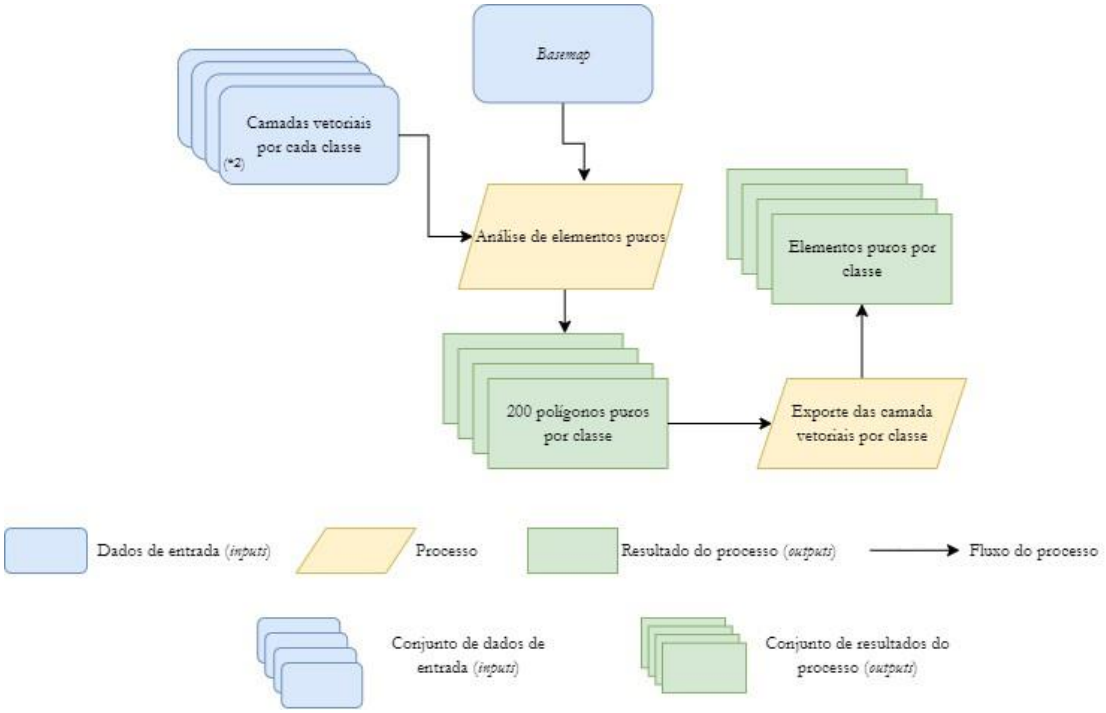


Figura 14 – (*2) Esquematização da fase de seleção de células das classes com valores espetrais puros.

Em alusão ao fluxograma, é pertinente destacar que a camada vetorial resultante da interseção mencionada, foi dividida em vários ficheiros vetoriais, estratificados por classe. Isto sucedeu em virtude da *shapefile* conter um número elevado de atributos, e tornando o processo de análise mais acessível. A fase de análise e de seleção de polígonos 100% puros, estratificados por classe, conforme mencionada anteriormente, resultou numa tarefa exaustiva, na qual foram selecionados 200 polígonos puros por classe, a fim de obter uma amostra viável das informações espetrais, aquando da análise dos valores presentes nos diferentes índices radiométricos. É de salientar que, para uma avaliação aprofundada, a classe representativa dos objetos urbanos, acabou por se dividir em grupos, onde contabilizaram individualmente, as características relativas aos asfalto, telha laranja, telha cinzenta e telha branca, uma vez que se suspeitava de variações nos valores espetrais, em detrimento destas características, acabando por tornar-se coerente perceber as respetivas oscilações.

Para permitir ao leitor uma visão mais clara sobre a seleção dos polígonos e os critérios considerados para determinar um elemento como 100% puro, apresenta-se a Figura 15.

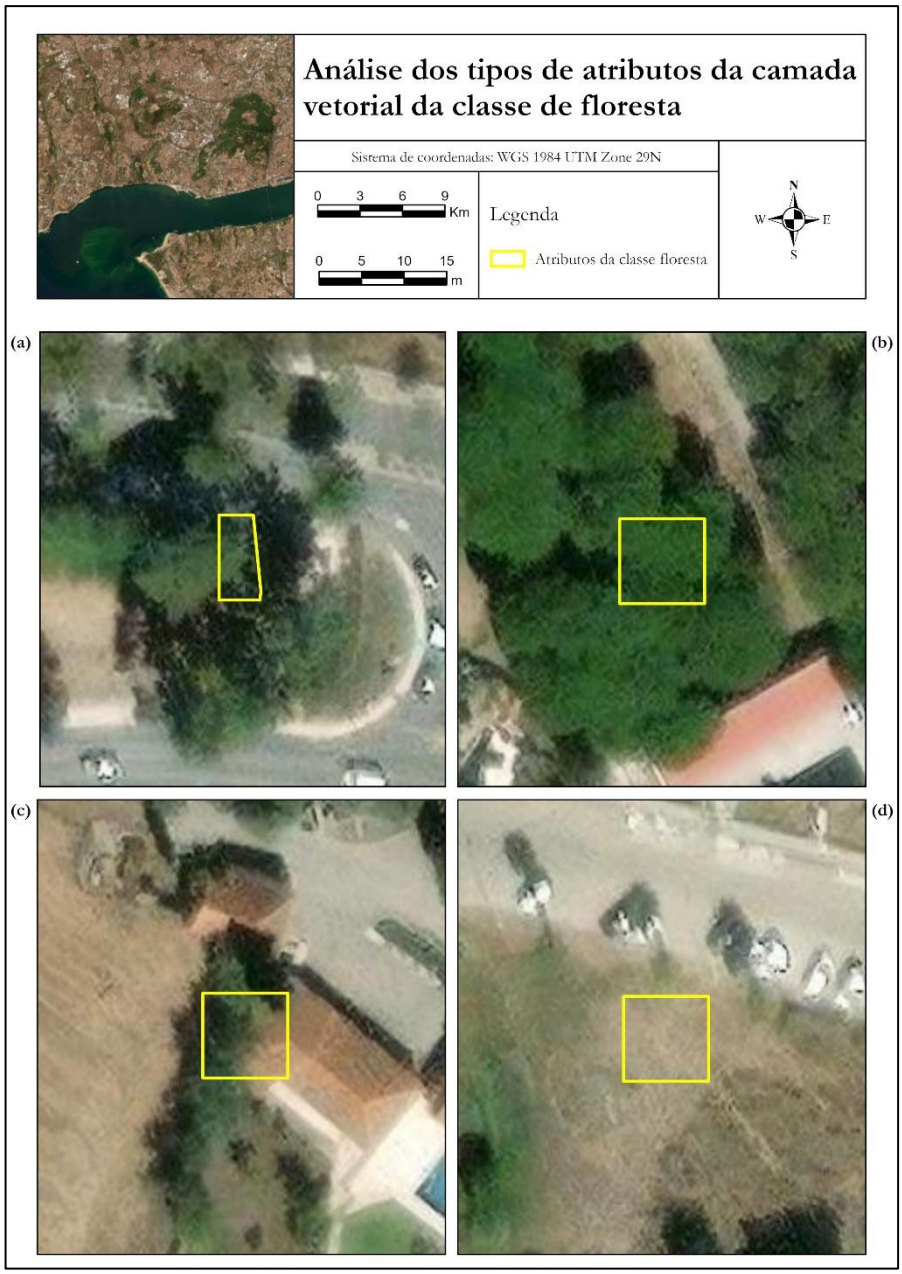


Figura 15 – Mapa de análise de atributos da camada vetorial para polígonos puros - caso específico para a classe *forest*
(a) Polígono com deformação geométrica; (b) Polígono com representatividade total da classe (100% puro); (c) Polígono misto (inclui classe de floresta e urbano); (d) Polígono desenquadrado com a classe.

Após identificar os diferentes polígonos com capacidade de individualizar a informação espetral para as classes, através da especificação de critérios de interesse explicados precedentemente, os ficheiros vetoriais com 200 atributos selecionados, foram exportados para camadas vetoriais isoladas, de forma a preservar os elementos de interesse e excluir atributos que comprometem a fidelidade dos dados.

O passo seguinte centra-se na conversão das camadas vetoriais de tipo polígono para camadas de objetos do tipo ponto. A presente etapa foi realizada para todos os ficheiros direcionados às diferentes classes, com recurso à ferramenta *feature to point*, com o objetivo de efetuar a transformação pretendida. Este método acarretou vantagens para a próxima fase, que consistiu na atribuição dos valores dos vários índices radiométricos calculados para os elementos das camadas vetoriais de objetos puros. É importante destacar que este processo foi repetido para as demais classes, e para tal, utilizou-se a ferramenta *extract value to point*, de modo a obter, em função da localização de cada ponto, os valores dos dados *rasters* correspondentes. A ferramenta tornou-se especialmente útil para obter informação relativa aos diferentes índices radiométricos para cada um dos pontos dos ficheiros vetoriais, e assim, obter os intervalos de valores que concentram as diferentes classes temáticas.

Foi imprescindível realizar mais uma conversão, para transformar a informação vetorial em dados de formato *raster*. O processo apenas foi possível, através da ferramenta *point to raster*, considerando que foi necessário repetir o procedimento para todas as camadas de cada classe. Esta fase tornou-se importante para conceber os intervalos de valores para cada índice radiométrico, nas respetivas classes, dentro de uma estrutura *raster*. Após a conversão das informações vetoriais para a estrutura *raster*, houve necessidade de dar saída aos ficheiros para fora do ambiente ArcGis Pro e guardá-los em formato *.tif*.

Deste modo, foi possível avançar para a última etapa da respetiva metodologia integrada na definição de novas regras para os limiares de corte. Nesta ótica, produziram-se histogramas, em função das informações contidas nos vários *rasters*. Quanto à estratégia de desenvolvimento de histogramas para a cada índice por classe, construiu-se um programa no âmbito deste trabalho, capaz de produzir vários resultados de forma sequencial com a mesma matriz lógica, exigindo programação da linguagem *python*. Os histogramas produzidos revelaram-se uma ferramenta importante de representação gráfica (barras verticais), para a análise da distribuição e identificação de padrões nos dados dos índices radiométricos por classe. Ademais, permitiram estabelecer valores limites e indicar os intervalos de concentração de cada classe em relação a cada índice radiométrico. Apresentado em anexo (Anexo C), encontram-se divulgados os referidos histogramas dos índices radiométricos calculados por classe.

Como forma de finalizar os métodos estabelecidos para o respetivo exercício e de alcançar os novos valores limites para filtragem dos dados de treino em auxílio dos índices radiométricos, procedeu-se a uma análise conjunta de quatro histogramas (número de datas

das imagens) por classe, para compreender o intervalo de valores em que estas se concentravam relativamente ao índice. Neste sentido, segue-se a Tabela XV, que faz referência para as novas regras de filtragem dos dados, considerando as diferentes classes em relação ao índice radiométrico e o número de imagens (datas) que devem respeitar o valor definido. É importante notar que as novas regras foram estabelecidas para os índices radiométricos pré-definidos, sendo estes o NDVI, NDWI e NDBI, da mesma forma que os novos índices introduzidos, nomeadamente MSAVI e NBR, por se considerarem como opções de grande contributo.

Tabela XV – Regras de filtragem de dados de treino com valores limites para índices ajustados às classes da nomenclatura.

Classes	NDVI/*Images*	NDWI/*Images*	NDBI/*Images*	MSAVI/*Images*	NBR/*Images*
Sealed surfaces	-0.1 to 0.3/*all*	-	-0.3 to 0.3/*all*	-0.2 to 0.5/*all*	-0.3 to 0.5/*all*
Herbaceous vegetation	0.0 to 0.6/*at least one*	-	-	0.2 to 0.7/*at least one*	-0.2 to 0.4/*at least one*
Forest	0.2 to 0.7/*all*	-	-	0.3 to 0.8/*all*	0.1 to 0.5/*all*
Shrublands	0.2 to 0.7/*all*	-	-	0.3 to 0.8/*at least one*	-0.1 to 0.5/*at least one*
Partly vegetated or non-vegetated land	-0.1 to 0.2/*all*	-	-	0.0 to 0.4/*at least one*	-0.2 to 0.2/*all*
Wetlands	>0.0/*at least one*	<0.0/*at least one*	-	-	-
Water	-	0.1 to 1/*all*	-	-	-

4.5.1 Segundo olhar na definição de novas regras

Na sequência daquilo que foi exposto, a respeito da abordagem introduzida para identificação de novas regras, houve outra pessoa que determinou as suas próprias regras, através da mesma metodologia. Ao contrário do ensaio exposto anteriormente neste estudo, o trabalho deste segundo autor passou pela análise de um número maior de áreas de estudo, resultando numa representatividade de polígonos puros superiores. Para o desenvolvimento da presente investigação, o trabalho simultâneo de duas pessoas para definição de novas regras, alberga benefícios ao nível do enriquecimento científico, uma vez que coloca à prova estas duas visões, como forma de retirar conclusões substanciais sobre o assunto.

Fruto de experiência individual, esta segunda investigação para a definição de novas regras de filtragem, envolveu uma análise mais complexa, de três regiões distintas, nomeadamente Lisboa, Coimbra e Parque Natural da Serra da Estrela, como forma de ampliar a variação

representativa do comportamento dos valores das classes em função do índice radiométrico. Do ponto de vista positivo, a definição das novas regras, desenvolvidas por esta segunda perspetiva, abarcou a atribuição dos valores de filtragem para as mesmas classes da nomenclatura definida nesta investigação e, portanto, não foi necessário realizar ajustes ou harmonização de classes informativas. Além disso, estas regras, envolvem em grande maioria, os mesmos índices radiométricos definidos para este trabalho.

Permite uma comparação direta entre os resultados obtidos na fase de treino dos modelos e classificação, admitindo averiguar qual dos dois conjuntos de regras obteve melhor desempenho, na individualização informativa dos dados mais adequados para a fase de treino do algoritmo de classificação. A Tabela XVI documenta os resultados das novas regras de filtragem, fruto do trabalho investigativo introduzido por esta pessoa.

Tabela XVI – Regras de filtragem de dados de treino com valores limites para os índices, produzidas pelo trabalho investigativo da segunda pessoa.

Classes	NDVI/*Images*	NDBI/*Images*	NBR/*Images*	NDCI[3]/*Images*	NGRDI[4]/*Images*
Sealed surfaces	0 *to* 0.1/*3 images*	0 *to* 0.4/*all*	-1 *to* 0.2/*3 images*	-0.3 *to* 0.2/*all*	-0.3 *to* 0/*all*
Herbaceous vegetation	0 *to* 0.4/*all*	-	-0.3 *to* 0.4/*2 images*	-0.1 *to* 0.5/*all*	-
Forest	0.7 *to* 1/*all*	-	0.4 *to* 1/*all*	0.3 *to* 1/*all*	-0.2 *to* 0.9/*all*
Shrublands	0.3 *to* 1/*2 images*	-	-0.2 *to* 0.4/*all*	0 *to* 0.7/*all*	-
Partly vegetated or non-vegetated land	0 *to* 0.3/*at least one*	0 *to* 1/*all*	-1 *to* 0.4/*2 images*	-1 *to* 0.1/*2 images*	-
Wetlands	0.4 *to* 1/*all*	-	-1 *to* 0.4/*2 images*	0 *to* 0.5/*all*	-
Water	-	-	-	-	-

[3] NDCI (*Normalized Difference Chlorophyll Index):* O Índice de Clorofila por Diferença Normalizada foi proposto originalmente para identificar a concentração de clorofila nas imagens de satélite (Mishra & Mishra 2012). O cálculo do respetivo índice envolve a utilização dos comprimentos de onda RED e RED EDGE 1 e a fórmula para as bandas Sentinel-2 é definida como:

$$\mathrm{NDCI} = \frac{\mathrm{b5} - \mathrm{b4}}{\mathrm{b5} + \mathrm{b4}}$$

[4] NGRDI (*Normalized Difference Green/Red Normalized Index):* O Índice de Vermelho/ Verde por Diferença Normalizada utiliza a razão da variação entre a banda GREEN e a banda RED para a discriminação da vegetação e exclusão dos efeitos da variação da radiação presente nas características espetrais (Meyer & Neto, 2008). Para as imagens Sentinel-2 a fórmula é dada por:

$$\mathrm{NGRDI} = \frac{\mathrm{b3} - \mathrm{b4}}{\mathrm{b3} + \mathrm{b4}}$$

4.6 Produção dos dados de treino e classificação das imagens de satélite

A classificação de imagens entende-se, como uma fase nuclear no desenvolvimento do trabalho, dado que inclui diversas tarefas de importância e acarreta um grande peso na determinação dos resultados. De alguma forma, tem influência na definição da nomenclatura de ocupação do solo, bem como a seleção apropriada de um algoritmo ou de vários algoritmos para a classificação de imagens. O tratamento e filtragem dos dados de treino integrados, determinação de índices radiométricos para a individualização de informação de classes e frequência de testes de classificação realizados são também pontos influenciadores da classificação. Considerando as várias componentes, tais como a seleção do algoritmo de classificação, as propriedades e a magnitude dos dados de treino, dentre outros fatores relevantes, é evidente que contribuem na determinação da abordagem metodológica mais apropriada para a concretização de uma classificação de imagens com exatidão.

O capítulo que se inicia, remete precisamente, para a fase de classificação de imagens. Além de toda as fases metodológicas devidamente elucidadas, é inconcebível negligenciar o foco relacionado com a correta aplicação dos índices radiómetros, bem como o amplo conjunto de dados de treinos, e respetivas características. Adicionalmente, não se pode deixar de considerar o algoritmo ou classificador selecionado para a execução da classificação das imagens.

É com base nestas considerações, que esta secção concentra a sua atenção, com exposição de um subcapítulo direcionado à exploração dos valores dos índices radiométricos calculados, de acordo com a classe de ocupação. De modo geral, esta secção tem como objetivo fundamentar a influência e o peso dos índices radiométricos para determinadas classes, demonstrando o seu desempenho na distinção da informação espetral correspondente.

Dada à quantidade considerável de dados de treino produzidos das diversas fases de filtragem, combinados com as várias informações procedentes das imagens de satélite, nomeadamente do Sentinel-2 e Landsat-8, juntamente com o correspondente MDE, é fundamental apresentar de forma abrangente as diferentes denominações atribuídas aos dados de treino. Além disso, é igualmente relevante esclarecer os critérios metodológicos selecionados para a filtragem dos dados, assim como identificar para que conjunto de *features* foram aplicados. Finalmente, sobre o último subcapítulo, resta estabelecer que classificador ou algoritmo de classificação foi selecionado para a classificação de imagens. A presente

secção assume, assim, um peso substancial, uma vez que a definição de classificador perfeito, capaz de executar a classificação de imagens com um rigor de 100%, é uma tarefa intangível. Neste contexto, é relevante considerar, que neste tipo de classificação, o algoritmo deve ser cuidadosamente selecionado para a recessão de dados de treino com os respetivos padrões intrínsecos. Consequentemente, o classificador deve demonstrar a capacidade de reconhecer padrões, com base nas características espetrais dos dados de treino, a fim de realizar uma classificação de imagens de satélite rigorosa.

4.6.1 Utilização dos índices radiométricos

Os índices radiométricos decorrentes do cálculo de fórmulas entre as bandas que compõem as imagens de satélite, são frequentemente utilizados como fontes de informação auxiliar, para o entendimento da distribuição dos elementos presentes à superfície terrestre. O tipo de solo pode influenciar os índices radiométricos, especialmente em regiões de baixa cobertura de vegetação, ou quando as características naturais do solo não são representadas de forma exata pelas imagens de satélite.

Por vezes, assume-se particular dificuldade em compreender, de forma separada, as respostas espetrais de determinadas propriedades do solo presentes nas imagens e, acaba por ser eficaz e promissor usufruir de métodos aliados para a decomposição delicada das informações espetrais, como forma de identificar traços específicos do solo em simples passos. A respeito desta interpretação, recorre-se a índices radiométricos como uma abordagem válida para individualização das propriedades informativas das características da superfície.

No decorrer do capítulo em particular, apoia-se nos índices radiométricos, para aplicação de processos de filtragem dos dados, cruciais na individualização das propriedades da superfície do solo. Estes representam um peso metodológico significativo para o desenvolvimento da classificação de imagens e determinação dos resultados. Revelando elevada importância no entendimento de fenómenos de relação entre o índice radiométrico e a classe informativa, o reconhecimento enfoca-se na análise visual do índice calculado em ambiente SIG, como forma de comprovar que alguns índices permitem distinguir objetos, processo realizado em conjunto com a análise da distribuição valores compreendidos nos histogramas concebidos, do Anexo C.

Considerando os valores das novas regras definidas, a Tabela XVII sintetiza os índices radiométricos calculados, e a respetiva aplicação para as classes de ocupação do solo, de forma a comprovar como se consideram úteis na identificação de classes e restrição da informação espetral.

Tabela XVII – Relação entre o índice radiométrico e as classes temáticas - Análise da variação de valores nos histogramas.

Correlação positiva entre índice radiométrico e classe temática	
Índice radiométrico	**Classe**
NDVI	***Forest***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG. ***Partly/Non-vegetated***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG. ***Sealed Surfaces***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG.
NDWI	***Water***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas, no entanto os valores podem variar consoante a concentração da clorofila; - Identificação visual da classe no índice calculado em ambiente SIG.
NDBI	***Sealed Surfaces***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG.
MSAVI	***Forest***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG. ***Partly/Non-vegetated***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas. ***Sealed Surfaces***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG.
NBR	***Forest***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG. ***Sealed Surfaces***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG. ***Partly/Non-vegetated***: - Variação regular dos valores compreendidos nas diferentes datas dos histogramas; - Identificação visual da classe no índice calculado em ambiente SIG.

4.6.2 Dados de treino e combinação de imagens

Dada a complexidade dos dados utilizados na classificação, que abrangem as várias bandas de imagens de satélite e informação altimétrica, assim como os conjuntos de dados de treino

baseados nos dados originais produzidos da conversão das informações OSM (*software* OSM2LULC), as regras de filtragem dos dados com referência para o artigo Fonte, *et al.* (2020), assim como a introdução das novas regras para a filtragem, levantam algumas questões sobre a utilização organizada dos dados relativos à fase de classificação.

Numa perspetiva coerente que exige maturação e organização do conjunto dos dados, é relevante concentrar-se em dois pontos essenciais que se encontram interligados. O primeiro ponto, com base nas regras de filtragem dos índices radiométricos utilizados no artigo Fonte, *et al.* (2020), bem como as novas regras exploradas para os mesmos índices. Assim, visa-se determinar qual é a utilização das bandas das imagens e MDE que garantem melhores resultados, respeitando os critérios de validação da classificação.

A meta de análise do conjunto de dados das imagens, pretende assegurar o grau mais elevado de qualidade dos resultados da classificação. Neste contexto, para o primeiro ponto, na Tabela XVIII são identificadas as aplicações combinadas dos dados de imagens, associados aos dois grupos de dados de treino pré-selecionados.

Tabela XIX – Identificação dos conjuntos de imagens, grupos de dados de treino e regras de filtragem.

Nome	Dados de imagens	Regras de filtragem
classi_D1_F1	D1	F1
classi_D1_F2	D1	F2
classi_D2_F1	D2	F1
classi_D2_F2	D2	F2
classi_D3_F1	D3	F1
classi_D3_F2	D3	F2
classi_D4_F1	D4	F1
classi_D4_F2	D4	F2
classi_D5_F1	D5	F1
classi_D5_F2	D5	F2
classi_D6_F1	D6	F1
classi_D6_F2	D6	F2
classi_D7_F1	D7	F1
classi_D7_F2	D7	F2
classi_D8_F1	D8	F1
classi_D8_F2	D8	F2

Tabela XVIII – Legenda dos códigos para noção dos nomes dos dados de imagens e regras.

Legenda	
D1	4 bandas Sentinel-2 (b02, b03, b04 e b08)
D2	10 bandas Sentinel-2 (b02, b03, b04, b05, b06, b07, b08, b8a, b11 e b12)
D3	4 bandas Sentinel-2 (b02, b03, b04 e b08) e Landsat-8 (B2, B3, B4 e B5)
D4	10 bandas Sentinel-2 (b02, b03, b04, b05, b06, b07, b08 b8a, b11 e b12) e 11 bandas Landsat-8 (B1, B2, B3, B4, B5, B6, B7, B8, B9, B10 e B11)
D5	4 bandas Sentinel-2 e MDE
D6	10 bandas Sentinel-2 e MDE
D7	4 bandas Sentinel-2, Landsat-8 e MDE
D8	10 bandas Sentinel-2, 11 bandas Landsat-8 e MDE
F1	Exclusão dos pixéis mistos e regras de filtragem do artigo Fonte, *et al.* (2020) para os índices radiométrico NDVI, NDWI e NDBI
F2	Novas regras de filtragem para os índices radiométrico do artigo Fonte, *et al.* (2020)

Depois de eleitas as classificações com maior destaque, deparam-se com as condições ideais para enquadrar o segundo ponto, que diz respeito às séries dos dados de treino na classificação. As respetivas unidades de dados de treino, desenvolvidos a partir de diferentes métodos de filtragem e da introdução de novos índices radiométricos, são os principais agentes de análise na comparação dos resultados debitados pelo suporte singular das informações incluídas nos determinados dados de treino. Os níveis de filtragem dos dados são divididos por cinco grupos, e podem ser verificados e distinguidos na Tabela XX.

Tabela XX – Identificação dos grupos de dados de treino e respetivas regras de filtragem.

Dados de treino	
Legenda	**Regras de filtragem**
F0	Dados originais obtidos da conversão OSM2LULC
F1	Exclusão dos pixéis mistos e regras de filtragem do artigo Fonte, *et al.* (2020) para os índices radiométrico NDVI, NDWI e NDBI
F2	Novas regras de filtragem para os índices radiométricos do artigo Fonte, *et al.* (2020)
F3	Novas regras de filtragem para os índices radiométrico NDVI, NDWI, NDBI e os novos introduzidos MSAVI e NBR
F4	Novas regras de filtragem introduzidas pelo trabalho e perspetiva de outra pessoa

4.6.3 Seleção do Classificador

Também a seleção do classificador desempenha um papel crucial na classificação de mapas, no que concerne às técnicas de *machine learning*. Este aglomera potencialidades capazes de determinar, de forma decisiva, o tratamento de informações e posterior definição de padrões no processo de classificação. Se usado corretamente, o algoritmo permite uma análise de variáveis capazes de avaliar a separabilidade de determinadas classes, relativamente ao conjunto das restantes, superando dificuldades relacionadas com a confusão de informação entre classes semelhantes. No seguimento do que foi mencionado anteriormente, e tendo em consideração a análise das características feita a um vasto grupo de algoritmos de classificação, torna-se imperativo afirmar que será utilizado o classificador supervisionado RF para o trabalho.

Caracterizado pelo conceito das árvores de decisão aleatórias, este classificador recorre a um baixo esforço computacional ao longo da sua execução, com base em regressões e classificação multiclasse de rápida operação, capaz de ser aplicado a problemas ligados à definição de padrões em grandes quantidades de informações integradas no conjunto dos

dados de treino. Este classificador tem vindo a alcançar popularidade na área da investigação, com excelente desempenho na sua execução, e consequentes, resultados da classificação de imagens. No estudo de Waśniewski, *et al.* (2020), os autores recorreram ao RF para a avaliação da potencialidade das imagens Sentinel-2, e do respetivo classificador para a criação de mapas de ocupação florestal, atingindo resultados de exatidão temática muito altos com atribuição de grande sucesso ao modelo. Não obstante, o projeto investigativo de Fonte, *et al.* (2020), como referido anteriormente, utiliza também na sua prática, o treino do classificador RF para classificar a série temporal de imagens Sentinel-2.

Dado ao suporte bibliográfico do artigo, e uma vez que a presente investigação gira em torno da otimização da metodologia no processo de filtragem dos dados de treino, uma das melhores formas de reconhecer tal alcance, será a partir utilização do mesmo classificador.

Para o processo de classificação, explorou-se um método automatizado por via da programação em linguagem *python*, subdivido em dois códigos, em que um se direciona para o treino e outro para classificação. Os respetivas tarefas, são de fato semelhantes, todavia, são utilizadas com objetivos distintos. O primeiro *script* tem como objetivo beneficiar do classificador RF para realizar o treino, sendo dado como parâmetro de entrada um ficheiro excel constituído por diferentes diretrizes. O ficheiro compõe-se por vários campos com indicação de caminhos de entrada e de saída de ficheiros. Contudo, os campos mais ponderáveis associam-se para o ficheiro *raster* que contém as informações dos dados de treino resultantes do processo de filtragem. Em conjunto com o respetivo ficheiro matricial são também definidos o número de árvores (definiram-se *ntrees*= 1000) e as composições de bandas das imagens satélite e MDE definidas para o treino. Ou seja, para cada treino ou experiência são produzidos grupos de treino, cujos elementos se distinguem pela seleção das bandas das imagens de satélite, bem como pela integração ou ausência do MDE. Esta primeira fase serve inteiramente para a conceção de ficheiros associados ao treino do classificador.

Ao contrário do ponto anterior, o segundo código visa a realização da classificação mediante da reutilização do classificador RF, utilizando como parâmetro de entrada o mesmo ficheiro excel. No respetivo programa são devolvidos *rasters* que apresentam a classificação obtida para o mesmo número de experiências definidas no programa, direcionadas ao treino do classificador.

4.7 Validação da classificação

No contexto da produção de mapas, torna-se imperativo a incrementação de uma estratégia de avaliação e validação dos mapas, a fim de assegurar a exatidão temática dos mesmos face à real estratificação da superfície terrestre. Nenhum exercício deste tipo estaria completo sem o incremento de uma estratégia de validação e, portanto, no respetivo caso são utilizadas duas estratégias de análise:

i. O primeiro método ou estratégia 1 visa a criação de um conjunto de pontos amostra de forma aleatória, utilizando a distribuição de pontos pelo mapa classificado a partir de modo *random*. Considerando a distribuição de pontos amostra de forma aleatória, a validação dos mapas prossegue de forma completamente normal, utilizando como referências as informações do Google Earth Pro e da seguida produção de matrizes de confusão e cálculo de medidas de validação.

   A respetiva metodologia de validação, considera-se como exequível, no entanto é importante considerar que se caracteriza como um método de validação de mapas inadequado, pela forma de distribuição dos pontos amostra totalmente aleatória, revelando-se abaixo do ideal. A eficácia desta primeira hipótese de validação pode mostrar-se insuficiente, uma vez que a representatividade dos pontos amostra de forma *random*, pode conduzir à incidência de por exemplo, apenas 3 pontos numa determinada classe, produzindo validações vagas. Os resultados podem tornar-se incorretos, assim como leva a questionar a exatidão temática nesta primeira forma de validação.

ii. O segundo método ou estratégia 2 utilizada na validação dos mapas, foca-se na comparação dos resultados com um mapa de referência. Nesta linha de pensamento, pretende-se comparar a informação obtida dos mapas resultantes da classificação, tendo como referência o mapa dos produtos *raster* CLC+ BB, uma vez que esta informação compreende a área de estudo definida no contexto desta investigação.

   Para este segundo método de validação, deve realçar-se que não são produzidas matrizes de confusão, mas tabelas de contingência, assumindo a mesma realidade e representação de medidas das matrizes. Como se utiliza informação produzida por outras entidades, ao contrário de pontos amostra, não é admissível designar as tabelas como matrizes de confusão. Esta é uma importante forma de discussão entre a informação dos mapas classificados com a atual informação CLC+ BB, permitindo

discutir pontos de concordância entre mapas, mas também analisar as diferenças e perceber qual das duas se aproximou melhor daquilo que é a real ocupação do solo.

Assim como em qualquer outra situação, com apelo ao mapa de referência, são necessárias realizar fases de preparação prévia da informação, de forma a ajustar inteiramente com os mapas produzidos a partir do exercício de classificação. A adaptação da informação de referência envolve a reamostragem do tamanho das células do *raster* para a mesma dimensão dos mapas classificados, de maneira a haver uma sobreposição correta entre o mapa CLC+ BB e os mapas produzidos. O recorte pela camada da extensão da área de estudo, a projeção para o mesmo sistema de coordenadas e a harmonização da nomenclatura do produto *raster* para as classes definidas neste estudo são outros pontos a considerar para estabelecer a concordância entre as informações dos mapas.

Completando todos os pontos de preparação do mapa de referência, transita-se para a construção das tabelas de contingência e consequente cálculo de medidas de validação para a análise entre a concordância das informações.

Depois da construção de matrizes de confusão e tabelas de contingência, resultantes da aplicação das duas estratégias de validação, visa-se fortalecer a avaliação dos mapas classificados de forma eficaz, com a utilização de procedimentos associados ao cálculo de diversas medidas de validação. Neste sentido, medidas básicas como exatidão global, exatidão do utilizador e exatidão do produtor, são considerados como indicadores indispensáveis no processo de validação de mapas. Contudo, no intuito de tornar a validação da classificação mais credível, é ainda adicionada a medida de cálculo pontuação f1 (*f1-score*). Os respetivos indicadores têm como objetivo apresentar de forma clara e concisa a confiabilidades dos dados produzidos pelos mapas classificados, realçando a exatidão das informações atendendo aos objetivos impostos. Atendendo à Tabela XXI, é fornecido um exemplo de matriz de confusão para uma classificação.

Tabela XXI – Exemplo de matriz de confusão para uma classificação.

		Classificação						
		c_1	c_2	c_3	…	c_n	**Total Geral**	**Exatidão Produtor**
Pontos aleatórios	c_1	t_{11}	t_{12}	t_{13}	…	t_{1n}	t_{1x}	$\frac{t_{11}}{t_{1x}} \times 100$
	c_2	t_{21}	t_{22}	t_{23}	…	t_{2n}	t_{2x}	$\frac{t_{22}}{t_{2x}} \times 100$
	c_3	t_{31}	t_{23}	t_{33}	…	t_{3n}	t_{3x}	$\frac{t_{33}}{t_{3x}} \times 100$
	⋮	⋮	⋮	⋮	…	⋮	⋮	⋮
	c_n	t_{n1}	t_{n2}	t_{n3}	…	t_{nn}	t_{nx}	$\frac{t_{nn}}{t_{nx}} \times 100$
	Total Geral	t_{x1}	t_{x2}	t_{x3}	…	t_{xn}	T	
	Exatidão Utilizador	$\frac{t_{11}}{t_{x1}} \times 100$	$\frac{t_{22}}{t_{x2}} \times 100$	$\frac{t_{33}}{t_{x3}} \times 100$	…	$\frac{t_{nn}}{t_{xn}} \times 100$		

Em alusão ao exemplo da matriz de confusão, verifica-se uma associação entre a classificação do mapa produzido e os pontos de referência, podendo esta variar para o mapa de referência, no entanto passa a designar-se como tabela de contingência. Para além disso, a tabela desempenha um papel fundamental no cálculo de medidas e estatísticas mais especificas.

Cada elemento na diagonal, destacado a amarelo, é considerado como o caso de concordância entre as classes representadas na classificação da imagem e no conjunto dos pontos aleatórios. Quando isto acontece, tem-se a proporção de elementos classificados corretamente para a determinada classe. Todos os elementos fora da diagonal, são casos em que existe uma discordância entre classes, ou seja, uma determinada classe da classificação de imagem confundiu-se com a classe da classificação ponto aleatório.

As unidades da tabela denominadas como "Total geral" tanto na orientação das colunas como das linhas, correspondem às somas do número de classes para cada linha da classificação de imagem com as proporções de cada classe para as colunas dos pontos de referência. Isto representa a correspondência de células classificadas para cada classe e a dimensão de células que são classificadas para a mesma classe da classificação dos pontos.

Em referência ao cálculo de medidas básicas para avaliação da exatidão temática de mapas, reconhece-se a "Exatidão Produtor", complemento do erro de omissão, é calculado a partir do quociente entre o número de pixéis classificados corretamente e número total de pixéis assinalados para essa mesma classe. Este indicador representa a probabilidade do um pixel de referência se encontrar classificado corretamente, estando a respetiva fórmula apresentada na tabela, na coluna do exemplo da matriz de confusão, determinada para cada classe. O erro de omissão, medida suplementar a esta exatidão, ocorre quando uma classe presente no conjunto dos dados de treino não é identificada corretamente e, portanto, a classe correta acabou por ser omitida.

Em alternativa, a "Exatidão Utilizador", uma medida do erro de comissão associada a cada classe, é derivada do número pixéis classificados corretamente pelo utilizador, em relação ao número total de pixéis em que o utilizador afirmou que a classe estava presente no mapa. Este indicador é obtido através da divisão entre os elementos classificados corretamente e a soma total dos elementos presentes na mesma linha. O erro de comissão, derivado do respetivo indicador ocorre quando o pixel é classificado como pertencente à classe no mapa, no entanto, nos dados de referência pertence a outra classe diferente. Por outras palavras, o utilizador cometeu um erro ao atribuir uma classe incorreta.

Na produção de uma matriz de confusão, é comum executar a exatidão global. Neste caso, como foi mencionado anteriormente, utiliza-se a exatidão global, que detém importância na tomada de decisão de utilização determinados mapas. Esta medida tem a capacidade de transmitir a partir de uma classificação, a proporção corretamente classificada. A medida é habitualmente expressa em percentagem, considerando o valor de 100% da classificação, como um valor perfeito, onde todos as classes da classificação foram classificadas corretamente. É calculada a partir da soma do número de classes classificadas corretamente, ou seja, os elementos ao longo da diagonal, identificados a amarelo, e pela divisão do valor total "T".

Numa fórmula matemática (11), a *Overall Accuracy* é calculada da respetiva forma:

$$\textit{Overall Accuracy} = \frac{(t_{11}+t_{22}+t_{33}+\cdots+t_{\mathrm{nn}})}{T} * 100 \qquad (11)$$

Uma vez que esta exatidão global é restrita apenas à avaliação global de um mapa temático, este não permite a associação a uma classe exclusiva, acabando por vezes, por se considerar

inconsistente, dado o exemplo de agrupamento de classes de um conjunto de amostras mais significativo do que outro. Neste caso, a medida prevê a classe maioritária com maior frequência, resultando em valores de exatidão elevados, mas com baixo desempenho da classe minoritária.

Avançando para a medida pontuação *f1* (*f1-score*), conhecida como uma média ponderada entre a "Exatidão Produtor" e "Exatidão Utilizador", esta é utilizada para equilibrar a compensação entre os valores obtidos dos indicadores referidos. Graças ao desequilíbrio nos conjuntos de dados utilizados na classificação de imagens, com variações significativas no número de amostras por classe, a pontuação *f1* proporciona um balanço entre as exatidões do produtor e do utilizador para cada classe. Quando o *f1-score* atinge um valor próximo do valor máximo, significa que se está diante de um bom classificador. Tal situação é observada somente quando os valores das duas exatidões se encontram também elevados.

O *f1-Score* é calculado como se apresente na seguinte fórmula (12), isto é:

$$f1\text{-}Score = \frac{2 * Exatidão\ Produtor * Exatidão\ Utilizador}{Exatidão\ Produtor + Exatidão\ Utilizador} \quad (12)$$

5 Implementação da metodologia

Seguindo os objetivos propostos, de forma a colocar em prática a metodologia, é necessário ter em conta que esta etapa apresenta-se também como uma fase crucial de todo o processo executivo. Considera-se como uma fase de transição, de toda parte teórica descrita, para a concretização prática. Para este efeito, os processos e métodos implementados visam a aplicação da metodologia, tendo sido explorada através de um exercício de utilização de *software* SIG, por via da programação.

A utilização de programação como principal método de concretização da metodologia é deveras importante, na perspetiva de automatização de processos e manipulação eficiente de vários conjuntos de dados, com um olhar voltado para a otimização lógica. Deve mencionar-se que no desenvolvimento do exercício, privilegiou-se a conceção e utilização de procedimentos programados por meio da linguagem *python*.

A construção dos códigos dependeu sempre do conhecimento relativo às propriedades e potencialidades do *software*, uma vez que se assumiu um cuidado especial em construir programas, tendo em consideração as ferramentas disponíveis por um *software* SIG. Todavia, quando o desenvolvimento de exercícios era comprometido pela performance limitada de alguns métodos SIG, contornou-se a adversidade, através do acesso a pacotes de *software* com soluções capazes de sustentar este avanço metodológico, deixando de colocar o desenvolvimento prático em risco.

Ainda assim, certos pontos do trabalho impulsionam a aplicação manual, justificada pela necessidade que diz respeito a uma análise minuciosa e execução de resultados altamente rigorosos, sem recurso à programação. A respetiva aplicação direta e manual, deve-se ao facto das ferramentas e métodos de análise se desviarem completamente de qualquer processo programado. Fazendo referência ao exercício de definição de novos limiares de corte por determinado índice radiométrico, a seleção de pixéis puros impulsionou para uma análise rigorosa, de forma a assegurar uma seleção e execução com exatidão, garantido a qualidade e integridade dos resultados. A adaptação à abordagem manual é essencial para assegurar processos fidedignos, com benefício de dois recursos SIG.

Neste sentido, o capítulo que se inicia irá debruçar-se sob os temas das tecnologias utilizadas em ambiente computacional, bem como o seu papel na execução da metodologia e tomadas de decisão na adoção de um determinado programa ou sistema informático. Inicialmente, a

respetiva secção tem em consideração a analise singular dos pacotes de *softwares*, implementados ao longo o exercício, organizando informações vitais da sua utilização e estruturação de todas as suas características. Neste subcapítulo 5.1, esclarece-se a seleção dos dois *softwares* SIG e outros, explorados e testados nesta fase.

A segunda subsecção destina-se, principalmente, ao recurso a programação de tarefas, onde são definidos genericamente os conceitos estruturantes das diferentes bibliotecas implementadas nos códigos aproveitados e construídos. Neste contexto, é apresentada uma leitura acessível, de modo a facilitar a perceção das diferentes bibliotecas *python*, de maneira a explicar a necessidade da sua utilização, bem como a adaptação aos diferentes códigos. Estes dois subcapítulos destinam-se essencialmente à descrição das tecnologias utilizadas para implementação metodológica e a sua contribuição na conceção de códigos programados para as diversas fases. O subcapítulo final segue a mesma linha, descrevendo as tecnologias implementadas na metodologia, com referência para os *softwares* SIG e outros de relevância para integridade da metodologia, assim como as bibliotecas *python*. A secção 5.3 destina-se, assim, ao aproveito e construção de códigos, para a execução de tarefas que procuram respeitar os objetivos propostos.

A estruturação dos respetivos códigos consideraram uma leitura acessível, facilitando a sua reutilização, promovendo a integração de novos métodos preparados com tarefas simplificadas, com garantias de segurança e integridade da informação processada. Para os programas construídos, foi também assegurado o controlo da incorruptibilidade do sistema computacional, de maneira a evitar *crashes*, e períodos de execução de processos muito demorados.

5.1 *Software* SIG

Os SIG são considerados como parte integrante, no campo das TIG, atualmente utilizados para desenvolver um conjunto de instrumentos de gestão de informação geográfica, com um papel fundamental na administração da superfície terrestre e de todas as atividades que nela se desenvolvem. Nascem da necessidade de analisar a informação geográfica, podendo ser classificados, segundo a literatura, como um sistema informático de *hardware*, *software* e dados geográficos destinados para obter, armazenar, atualizar, manipular, analisar e exibir dados geográficos cuidadosamente georreferenciados (Cosme, 2012). Contudo, a definição de SIG pode ser problematizada, e segundo a maioria dos autores pode abordar outras noções, sendo

para além de um sistema informático, também um sistema manual, como é exemplo da seguinte definição de SIG: "*Qualquer sistema informático ou manual baseado num conjunto de procedimentos utilizados para armazenar e manipular informação georreferenciada*" (Aronoff, 1989).

Neste contexto, são reconhecidas várias definições e significados de SIG, e não é possível chegar a uma definição concreta e precisa. No entanto, no desenvolvimento das tecnologias, a perspetiva de SIG surge como uma estrutura que envolve a anatomia do *software* e *hardware*, como partes integrantes do processo de tratamento de dados geográficos. Paralelamente, o significado de SIG ligado à computação, surge do crescente desenvolvimento computacional para processamento de dados, funcionando como agente que garante eficiência à manipulação de dados. Assim, o SIG, pode desenvolver-se ao nível de processos em ambientes de programas informáticos, associado a uma máquina e a um vasto conjunto de ferramentas enunciados em linguagem de programação. Procedente do avanço computacional que se tem sentido, os *softwares* utilizados para o processamento de dados de geográficos são vários (Patriarca, 2016).

A ampla gama de *softwares*, surge na área de SIG como forma de alargar ferramentas e capacidade de processamento de dados geográficos em determinados programas. Ao deparar-se com um vasto conjunto de programas computacionais, é legítimo ter em consideração a eficiência e performance, bem como exatidão lógica e matemática dos *outputs*, e outros parâmetros de qualidade, na medida em que, todos os argumentos fundamentam a tomada de decisão no momento de escolha de um *software* (Patriarca, 2016).

Com um olhar voltado para o mencionado, o GRASS GIS foi o *software* eleito para atender aos critérios vinculados para o desenvolvimento de toda a metodologia, aliada a processos automatizados. No que diz respeito à disponibilidade de ferramentas, o GRASS, dispõe de um completo repositório, com um especial foco para a manipulação e tratamento de ficheiros de formato *raster*. Uma vez que esta investigação trabalha, na sua maior parte, com imagens de satélite e dados da respetiva extensão, o *software* tem à sua disposição uma grande variedade de ferramentas que respondem às necessidades. A seleção do GRASS GIS vai também ao encontro de se ter privilegiado do acesso a exemplos de utilização da sua API *python*, admitindo a adaptação de funções e módulos, integrando-os no respetivo desenvolvimento prático. Este ponto debruça-se no acesso ao repositório GLASS, disponibilizado pela equipa de orientação, com funções, módulos e ferramentas GRASS GIS programadas, de maneira a acelerar a execução de tarefas através da aplicação de programas pré-existentes, bem como a criação de novas cadeias lógicas. Como se detinha de um interesse para automatizar

processos e em referência a todos os pontos justificativos, a seleção do *software* GRASS GIS na sua versão 7.8, foi determinada para a execução metodológica.

Para além deste, de forma a colmatar os objetivos práticos automatizados, foi essencial recorrer a um segundo *software* SIG. Neste sentido a seleção passou também pela utilização do GDAL/OGR, uma biblioteca equivalente a um *software*, pelo que vai muito para lá de uma biblioteca como o numpy ou geopandas. Este distingue-se dos demais *softwares* SIG, uma vez que não detêm de interface gráfica, no entanto pode ser encontrado e executado em *tools* de outros *softwares*, como QGIS ou por via da programação. O GDAL/OGR foi lançado sob a licença de código aberto pela OSGeo para ler e manipular uma ampla variedade de ficheiros geospaciais nas suas extensões vetorial e *raster*. A sua seleção mostrou-se relevante, tendo em vista que ao longo de alguns processos automatizados a sua utilização foi obrigatória, como substituição ao GRASS GIS.

Contudo, tendo em conta que nem toda a metodologia permitiu uma execução a partir da programação de ferramentas por cadeia lógica, foi necessário recorrer a um terceiro *software* SIG. Este ponto relaciona-se com a investigação do método de reamostragem mais adequado ao pré-processamento do MDE, e ainda, com o exercício de análise para a definição das novas regras de filtragem para os diferentes índices radiométricos (secção 4.3.4.1 e secção 4.5 respetivamente). Em alusão aos respetivos subcapítulos, o *software* ArcGis Pro da Esri foi eleito para este desenvolvimento experimental, exclusivamente valorizado por motivos que remetem para a familiarização da sua interface gráfica e ferramentas, por experiências anteriores e pela fiabilidade da sua performance na maioria dos exercícios que envolvem tarefas complexas ou um volume de dados consideravelmente elevado.

Assim sendo, a implementação da presente metodologia, envolve fundamentalmente a utilização do *software* GRASS GIS na versão 7.8, GDAL/OGR e ainda o ArcGIS Pro para determinadas etapas processuais que não requisitaram de programação de ferramentas, mas sim de processos manuais. Mais adiante, ainda no seio do capítulo da implementação da metodologia, serão apresentados e caracterizados os vários programas que concretizam os pontos metodológicos conceptualizados, considerando apenas a execução de programas a partir de ferramentas GRASS GIS, GDAL/OGR e outras bibliotecas, de forma a explicar sucintamente a cadeia de processos lógicos e as suas principais propriedades.

5.2 Bibliotecas e módulos

A linguagem *python* é, atualmente, a mais comum no domínio das ciências e tecnologias espaciais. Conhecida como linguagem FOSS (*free open source software*) de acesso livre, como o nome indica, foi lançada em 1991 por Guido Van Rossum, permitindo a construção de programas e ferramentas de várias escalas. É compatível com muitos sistemas operativos e capacita a sua execução numa ampla variedade de sistemas, como o Linux, macOS e Windows. A linguagem permite uma sintaxe simples e um suporte de bibliotecas para as mais diversas áreas de investigação. É comum ouvir dizer que *python,* carrega em si "baterias incluídas", com alusão à vasta biblioteca padrão de módulos e pacotes. Os módulos são ficheiros fonte ou uma extensão que pode ser importada, no código, expandindo as capacidades e funcionalidades do *python.* Tipicamente, o módulo contém uma série de funções específicas que, uma vez importado, pode ser "chamado" e executado no programa (Borges, 2014).

Ao longo dos anos, o *python* tem vindo a evoluir de modo a aliar-se aos SIG, por meio da utilização de bibliotecas de módulos externos e ferramentas mais específicas, importantes para capacitar a manipulação de dados geográficos complexos, desde o formato vetorial passando por dados matriciais, e até mesmo dados do tipo pontos. Graças à sua versatilidade, e extenso conjunto de bibliotecas, possibilitou os profissionais em SIG, a executar, com eficácia, exercícios automatizados de manipulação de dados, análise espacial e visualização. Neste âmbito, com alusão para as várias bibliotecas *python*, considerando os módulos padrão da linguagem de programação e ainda as bibliotecas externas incorporadas, com objetivo de manipulação de dados geográficos, serão apresentadas nos pontos seguintes, todas aquelas que de alguma forma auxiliaram à implementação da metodologia, promovendo a sua boa execução.

O conjunto de bibliotecas padrão de *python* e os módulos externos associados a funções SIG utilizadas são as seguintes:

a. **OS**, fornece funções de interação com o sistema operacional, assim como uma forma portátil de interagir com ficheiros e pastas permitindo criar, eliminar, procurar ficheiros e identificar estes nas pastas.
b. **Random***,* utilizado para gerar números pseudoaleatórios, ou seja, não podem ser considerados completamente aleatórios. Este módulo é usado para ações como a conceção de números e conferir aleatoriamente um valor para uma lista ou *string.*

c. **Shutil**, oferece operações de alto nível para ficheiros ou conjuntos de ficheiros, para automatizar ações de cópia e eliminar ficheiros e pastas.
d. **Zipfile,** fornece ferramentas para criar, ler, gravar, anexar e listar um ficheiro *.zip*, padrão comum de pastas compactadas.
e. **Platform,** recolhe informações sobre a plataforma, na qual o programa está a ser executado, estando relacionadas com o sistema e a versão do sistema operativo, a versão do *python*, etc. Permite verificar a compatibilidade com a versão do *python* instalada num determinado sistema operativo, identificar se um programa pode ser executado, e perceber se algumas bibliotecas ou módulos externos possuem diferentes interfaces.
f. **Subprocess,** permite executar novos programas, com a criação de novos processos, e ainda conectar-se a caminhos de entrada, saída, erro e a códigos.
g. **PyGDAL,** conhecida como API *python* do GDAL/OGR, é uma biblioteca de código aberto *pyhton* para trabalhar e manipular dados geospaciais através da linguagem de programação *python*.
h. **PyGRASS,** considerada como uma API direcionada a objetos SIG do *software* GRASS GIS, é uma ferramenta de código aberto utilizada em áreas de especialização. A arquitetura de PyGRASS, compreende interfaces para os módulos GRASS, destinados à manipulação de dados vetoriais e *raster*, focando-se nos novos recursos que o *software* fornece aos utilizadores e desenvolvedores. Esta permite a ligação direta de entrada e saída de módulos de forma a criar cadeias lógicas de processamento de dados, incluindo a análise de compatibilidade, controlo de processos e correção de erros. A API de programação em *python* possui uma interoperabilidade que permite a utilização e o acesso de forma independente a todas as funções, classes e módulos do ambiente de geoprocessamento (Zambelli, *et al.* 2013).
i. **Shapely,** utilizado na análise e manipulação de características através de funções da biblioteca GEOS. Está inteiramente ligado às geometrias da extensão espacial, enraizadas nos sistemas de informação geográfica. Permite a manipulação e analise de objetos 2D, caracterizados como pontos, linhas e polígonos. Ademais, permite ainda, definir vários objetos ao mesmo tempo, com múltiplas projeções, uteis para definição de geometrias, sem haver esgotamento ao longo da execução do módulo (Gillies, 2013).

j. **Numpy,** fornece objetos poderosos, denominados por *arrays* de estrutura multidimensional, capazes de armazenar e manipular com eficiência grandes conjuntos de dados. Tem a capacidade de armazenamento e conferir mais rapidez às operações numéricas. Compreende ainda, funções matemáticas e estatísticas, que têm vindo a ser adotados em áreas da investigação e vias profissionais. O *array* descreve um ou mais blocos de memória do computador para que a manipulação de números possa ser facilitada, fornecendo um alto nível de computação numérica, exigindo menos esforço de programação. Oferece, também, recursos avançados para processos de indexação e divisão de *arrays*, permitindo extrair, modificar e manipular elementos das matrizes (Walt, *et al.* 2011).

k. **Rasterio,** baseada em bibliotecas como o GDAL e o Numpy, é importante para aceder e ler diferentes tipos de ficheiros de dados *raster* utilizados em SIG. Desenvolveu-se devido a *bugs*, apresentando melhorias no seu desempenho, menos carga cognitiva, e códigos mais simples e limpos (Gillies, 2019).

l. **Pandas,** apresenta dados flexíveis e rápidos, tornando a manipulação de dados relacionais mais fácil e intuitiva. O objetivo principal enfoca-se na construção de um bloco fundamental de alto nível para análise de dados em *python*, e outras linguagens de programação. É adequada para dados em tabelas, por exemplo SQL ou excel, dados de séries temporais e matriciais com linhas e colunas identificados por legendas (McKinney, 2015).

m. **Simpledbf,** converte ficheiros DBF em ficheiros Pandas *DataFrames*, tabelas SQL ou tabelas HDF5. Projetado no âmbito da simplicidade e eficiência, relativamente à memória utilizada no processamento de ficheiros.

n. **Pyexcel_ods,** assume o objetivo de ler, preparar e gravar dados no formato *.ods* através de *python*. O *.ods* pertence ao grupo de ficheiros *Open Document Formats*.

o. **Joblib,** executa tarefas em múltiplas operações, possuindo ferramentas que auxiliam na execução de códigos complexos, otimizando-os de forma a tornarem-se mais rápidos. É utilizado para trablhar dados de grandes dimensões, com otimizações particulares para *arrays* Numpy. Para otimização dos processos, concentra-se no *disk cache*, memória usada para acelerar o armazenamento e acesso de dados do disco rígido e processamento paralelo, com a execução de várias operações ao mesmo tempo (Faouzi & Janati, 2020).

p. **Scikit-Learn,** também conhecida como Sklearn, oferece um conjunto de ferramentas *machine learning* e de modelação estatística, incluindo algoritmos de

classificação, regressão, *clustering*, abarcando classificadores como SVM, RF, etc. Foi projetada tendo em consideração a simplicidade e fácil utilização, tornando-se indispensável a investigadores que procurem facilitar fluxos de trabalho.

q. **GeoPandas,** facilita o processamento com dados geográficos em linguagem *python*, com extensão dos tipos de dados utilizados pelo Pandas. Além disso recorre ao Shapely para permitir operações espaciais para vários tipos de geometrias (Jordahl, 2016).

r. **Multiprocessing,** permite ao computador executar várias tarefas ou processos em simultâneo repartidos pelos vários núcleos do CPU. É capaz de utilizar todos os recursos do *hardware*, repartindo cada processo por cada núcleo ou processador. O desempenho do programa é ampliado por uma execução mais rápida e fluída, uma vez que existe uma redução de carga por cada núcleo individual.

5.3 Implementação da metodologia e automatização de procedimentos com *Python*

Com o objetivo de elucidar o leitor, e tendo em consideração os métodos e procedimentos técnicos, que merecem um esclarecimento fundamentado e profundo no avanço do presente subcapítulo, será abordado o que se concretizou no âmbito da programação da metodologia. Deve dizer-se, que no desenvolvimento deste exercício, se privilegiou a conceção e utilização de procedimentos programados, através da linguagem de programação *python*.

É pertinente explicar, que a construção dos códigos utilizados pode não ter seguido o caminho mais eficiente e simplista, uma vez que, não existe um contato experiente com a linguagem de programação, porém o recente conhecimento da área permitiu, a concretização da metodologia na sua totalidade. Houve o cuidado de programar numa lógica direcionada ao aproveitamento das ferramentas disponíveis pelo *software* GRASS GIS. Para a construção de códigos programados, a interface *web* utilizada foi o Jupyter Notebook, *software* de código aberto que permite criar e partilhar ficheiros que contêm *scripts* ativos, equações, visualizações e textos. Este foi usado para a transformação e eliminação de impurezas nos dados, para a sua visualização e aplicação de técnicas de *machine learning*, através da linguagem de programação *python*. A possível partilha de *notebooks*, concebeu o acesso ao repositório de códigos e funções pré-programadas do GLASS (*The Geoscientific Library for Analysis of Spatial Systems)*.

"*The Geoscientific Library for Analysis of Spatial Systems (*GLASS*) is a free and open source library for geospatial data science. It is a set of Python Methods to support the automatization of spatial data science activities based on Geographic Information Systems Software. These Python Methods could be included in any high-level geospatial analysis application*" (https://github.com/jasp382/glass), graças à respetiva base de *scripts* considerou-se como o ponto de partida, para a construção de qualquer código programado, nos processos conceptuais do exercício. Ainda assim, deve salientar-se que antes de tomar proveito do repositório, foi pertinente estudar as características das várias funções, de maneira a entender o objetivo e a sua utilidade para a construção de novos *scripts* programados.

Nas secções seguintes serão apresentados todos os pontos que de alguma forma, necessitaram de fazer recorrer à programação de códigos, com a devida identificação, objetivo e análise detalhada das tomadas de decisões.

5.3.1 *Software* utilizado no pré-processamento de dados de imagem

Nas situações em que se trabalha com imagens recolhidas de diferentes plataformas *web*, com particularidades únicas, o pré-processamento de dados é sempre recomendável, como forma de harmonizar e igualar as componentes das diferentes informações. Esta fase da metodologia tem como objetivo garantir a igualdade perante as características das informações de imagens adquiridas, alcançando uniformidade no que diz respeito à resolução espacial, sistema de coordenadas e projeção, extensão geográfica e mitigações de erros, tal como os efeitos atmosféricos encontrados nas imagens.

Para cada padrão de dados de imagens, a fase de pré-processamento, como foi demonstrado anteriormente, seguiu diferentes modelos processuais. Esta secção destina-se à explicação aprofundada e particular das decisões tomadas e ferramentas aplicadas para os determinados dados de imagens.

5.3.1.1 *Integração de software desenvolvido por terceiros para o pré-processamento de imagens Sentinel-2*

Face ao que foi exposto, e em alusão aos métodos de programação para o pré-processamento dos dados de imagens Sentinel-2 implementados, os programas disponibilizados pela orientação, foram particularmente concebidos para auxiliar na fase de iniciação da metodologia programada. As etapas metodológicas procuraram cumprir as seguintes tarefas:

i. Descompactar as diferentes bandas que compõem as imagens Sentinel-2, visto que aquando do *download* das suas plataformas, estas se encontram, geralmente, em pastas compactadas no formato .*zip*;
ii. Reamostrar o tamanho do pixel das bandas das diferentes imagens para a resolução espacial de referência da banda 2 (10 metros), e recortar o conjunto de imagens pela extensão da área de estudo;
iii. Conceção de máscaras de nuvens, com produtos originados do *software* Fmask para a exclusão de todas as células com valores de nuvens, sombras e outras intervenções, que colocassem em causa as informações espetrais, depois de se ajustar a meta em função dos resultados adquiridos pelo primeiro código programado aliados aos produtos SCL.

Para tal utilizou-se o Programa 1[5], disponibilizado pela orientação, como meio de garantir o cumprimento dos dois primeiros pontos. Este tem a capacidade de receber os diferentes dados das imagens Sentinel-2, após o descarregamento da interface *web* em pastas compactadas .*zip,* para descompactar os diferentes ficheiros, referentes ao conjunto das bandas. Depois da descompactação auxiliada do módulo Zipfile, é realizada a reamostragem do pixel das imagens e o recorte pela extensão da área de estudo, através das dependências do PyGRASS e GDAL/OGR. Este ponto é apenas conseguido pelo dado de entrada de uma camada *raster* ou vetorial de referência, com a extensão da área limite e uma resolução espacial de 10 metros para a reamostragem das restantes bandas, para a mesma dimensão das células.

O terceiro ponto, foi proposto num primeiro momento pelo Programa 2[6], aliado à criação de produtos SCL, com a principal função de remover as informações espetrais perturbadas pelos efeitos atmosféricos. Este segundo programa encontrava-se também disponível e foi

[5] Ver Anexo D1, Programa 1: *Resampling Sentinel-2 images to 10 meter spatial resolution* (código existente).

[6] Ver Anexo D2, Programa 2: *Remove clouds from Sentinel-2 bands* (código existente).

aproveitado para a conceção de máscaras de nuvens, das quais criaram-se máscaras binárias (valores de *Null* e 1) para cada data das imagens, recorrendo a regras de reclassificação dos valores de células. A partir dos produtos SCL, as várias máscaras binárias atribuem o valor de 1 às classes de pixéis que pretendem ser mantidos, enquanto o valor de *Null* é atribuído a valores dos produtos que identificam 2. CAST_SHADOWS, 3. CLOUD_SHADOWS, 8. CLOUD_MEDIUM_PROBABILITY, 9. CLOUD_HIGH_PROBABILITY e 10. THIN_CIRRUS[7]. Posteriormente, é gerado um ficheiro *clouds* de extensão *.tif*, onde se encontram todas as máscaras das diferentes datas das imagens conjugadas, com a atribuição do valor *Null* para as células perturbadas, enquanto as informações restantes são mantidas por representar dados sem impacte das condições atmosféricas. Bibliotecas como a Numpy e PyGRASS foram imprescindíveis para a execução do programa. É importante notar que neste código, quanto mais datas de imagens fossem introduzidas, maior seria o número de células com informações irrelevantes, condicionando a representação do ficheiro de nuvens de saída, com um número superior de "áreas buraco", partes da imagem sem informação espetral.

Este Programa 2 foi, inicialmente, construído para cumprimento do respetivo objetivo com os produtos SCL, todavia, como foi mencionado na conceção teórica da metodologia, este acabou por manifestar algumas lacunas devido a confusões entre classes de ocupação, nuvens e sombras, acabando por originar a perda de informações relevantes. Devido a tal adversidade, e tendo em consideração a experiência pessoal, tornou-se relativamente acessível contornar esta questão através do algoritmo Fmask. Deste modo, encontrou-se a solução adequada para permitir o avanço metodológico, com resultados satisfatórios, mas infelizmente, o respetivo rumo tomado desperdiça uma parte do trabalho investigativo desrespeitando o conceito de uma metodologia programada.

5.3.1.2 *Integração de software desenvolvido por terceiros para o pré-processamento de imagens Landsat-8*

Na segunda fase de pré-processamento, com o objetivo principal de efetuar tratamento às imagens Landsat-8, procuraram-se etapas semelhantes aos programas executados para as

[7] Guião técnico do algoritmo de classificação dos produtos *Scene Classification Maps* (SCL), para a criação de máscaras classificadas em função dos valores das classes e da sua representatividade(https://Sentinels.copernicus.eu/web/Sentinel/technical-guides/Sentinel-2-msi/level-2a/algorithm-overview).

imagens Sentinel-2. Tomando proveito dos códigos divulgados pela orientação, os blocos de código criados para o pré-processamento das imagens Landsat-8 procuraram uma construção adaptada às características especificas dos dados. Uma vez que os Programas 1 e 2 permitem a execução apenas para os dados de imagens Sentinel-2, foi necessário realizar adaptações para que estes respeitassem as características dos respetivos dados. É preciso notar que as imagens Landsat-8, exigem um tratamento mais rigoroso e para tal, foi importante, para além da adaptação de códigos programados anteriormente, criar blocos de código programados no âmbito deste estudo, para um processamento de imagens detalhado. Assim, as principais fases de procedimentos debruçam-se sobre os seguintes pontos:

i. Aplicação de correções atmosféricas, isto é, a conversão dos DN para radiância TOA, e correções topográficas da refletância nas imagens, com recurso à utilização do modelo de iluminação, produto do MDE. Como as imagens Landsat-8 foram descarregadas no seu nível L1TP de processamento, necessitaram das correções evidenciadas;
ii. Reamostragem do tamanho do pixel do conjunto das bandas para 10 metros, de forma a uniformizar resolução espacial relativamente às imagens Sentinel-2 e recortar as várias bandas pela extensão da área de estudo;
iii. Produção de máscaras de nuvens, tal como se concessionaram para as imagens Sentinel-2, mais uma vez, recorrendo ao Fmask disponível para uso público.

O primeiro ponto de intervenção aplicado às imagens Landsat-8, foi conseguido pela cadeia lógica do Programa 3[8], que incrementa geoprocessos simples do pacote de ferramentas do *software* GRASS GIS. O respetivo código, reverencia uma sequência lógica de início de uma sessão GRASS, utilizando um ficheiro *raster,* com o objetivo de aludir à extensão da área de trabalho, obrigatório na criação de qualquer sessão por este *sofware*. Na preparação das várias bandas após a sua importação, o programa incrementa a ferramenta *i.Landsat.toar* para correção atmosférica. Deve ser evidenciado que para a correta execução da ferramenta, é necessário identificar o tipo de sensor do satélite e o caminho para o ficheiro dos metadados. Em seguida, o *i.topo.corr* é implementado e corrido duas vezes para a correção topográfica. A primeira execução da *tool* destina-se à criação do modelo de iluminação, dando como entrada o MDE e os respetivos valores da distância zenital e azimutal, obtidos a partir do ficheiro de metadados. A distância do zénite é dada através da subtração de 90º pelo valor da elevação

[8] Ver Anexo D3, Programa 3: *Atmosferic and topographic Landsat-8 images correction* (código programado no âmbito deste trabalho).

do sol, já o azimute está corretamente definido no ficheiro. Efetuando a ferramenta pela segunda vez, esta conduz à correção topográfica das bandas, auxiliada do modelo de iluminação produzido. Depois das correções, o programa encaminha os dados para uma calculadora de *rasters* (*r.mapcalc*), para realizar a multiplicação singular de cada banda por 10 000. Esta abordagem serve para igualar a informação Landsat-8 às bandas Sentinel-2 e converter os valores para números inteiros. É importante reconhecer que para o programa executar devidamente, as diferentes imagens Landsat-8 têm de respeitar a extensão original do pós-descarregamento, assim como o MDE utilizado tem de igualar esses limites e obedecer à mesma resolução espacial das imagens.

Quanto à reamostragem do tamanho das células das várias bandas para uma resolução de 10 metros e recorte pelas dimensões da área de estudo, o Programa 4[9] mostrou-se à altura para a concretização do objetivo. O respetivo código remete para uma adaptação do programa utilizado na reamostragem das bandas Sentinel-2, respeitando um comportamento conceptual e lógico idêntico. As simples adaptações de funções implementadas no *script* original, como a atualização dos números que identificam as bandas Landsat-8 manifestaram um desempenho eficaz na conformidade das metas definidas.

Tendo por base o produto de saída do Programa 4, as bandas Landsat-8 apresentam o tamanho do pixel reamostrado de 10 metros, como pretendido, no entanto, tornou-se imprescindível recorrer, à exclusão de nuvens e sombras encontradas nas imagens. Esta fase é um exemplo de como os conhecimentos sólidos em programação e tratamento de imagens podem ser cruciais ao encontrar limitações e condicionalismos. Neste sentido, pela falta de conhecimento neste nível, julgou-se que a solução mais adequada seria recorrer novamente ao algoritmo Fmask, dada a inexistência de produtos SCL para as imagens Landsat-8, assim como por ter revelado resultados satisfatórios nas imagens Sentinel-2. Recorda-se que as imagens Landsat-8 utilizadas na presente investigação foram descarregadas no nível de processamento L1TP e, portanto, detinham características adequadas para serem submetidas ao *software* Fmask. Por infelicidade, esta abordagem não utiliza métodos programados, por falta de tempo, todavia, isso não impossibilitou o sucesso da tarefa, e concessão de máscaras de nuvens e do ficheiro com as respetivas informações.

[9] Ver Anexo D4, Programa 4: *Resampling Landsat-8 images to a 10 meter spatial resolution* (código existente e adaptado no âmbito deste trabalho).

5.3.1.3 *Software desenvolvido para o pré-processamento do MDE*

No que diz respeito ao último ponto de pré-processamento de dados, com alusão ao MDE, as etapas de tratamento assemelharam-se às fases de processamentos aplicadas nas imagens de satélite. Contudo, é preciso notar que as características do modelo diferem dos restantes dados, com alusão para o sistema de referência definido pela agência divulgadora da respetiva informação altimétrica. Este ponto foi tido em consideração, bem como todo o exercício investigativo para o reconhecimento do método de reamostragem, para transformação das dimensões dos pixéis que melhor preservasse os valores de elevação relativos à área de estudo. Neste contexto, a "ordem de trabalhos" para o tratamento do MDE, obedece aos seguintes pontos:

i. Reamostrar as dimensões das células do modelo para a resolução espacial de referência (10 metros) e recortar pela extensão da área de estudo;
ii. Projetar o MDE para ajustar o sistema de referência à localização da área de estudo e igualar o sistema com os outros dados de imagens;
iii. Processar a correção da superfície do modelo para eliminar imperfeições nos dados de elevação, relativamente às diferenças entre topos e bases de vertente.

O pré-processamento do MDE foi gerado a partir do Programa 5[10], com a implementação de funções do repositório GLASS e das ferramentas análogas do *software* GRASS GIS, executadas numa ordem lógica tendo em vista o resultado pretendido. Para o início de sessão GRASS dentro do código, fez-se uso de um *raster* de referência para área de trabalho com características que o diferem dos utilizados nos *scripts* descritos anteriormente. Antes do ficheiro de referência *raster* se ter dado como entrada, sofreu um procedimento *buffer*, de forma a aumentar a sua extensão. O procedimento considerou-se necessário, uma vez que, métodos de interpolação, geralmente, reduzem o número de pixéis de um *raster*, e para limitar erros inerentes aos bordos pensou-se trabalhar com uma extensão de área superior. Com o MDE importado na sessão, a sequência lógica do programa encaminha o ficheiro para a ferramenta *r.proj*, com o objetivo de projetar o sistema de coordenadas pré-definido para o mesmo sistema das imagens de satélite (ETRS 1989 LAEA para WGS 1984 UTM zona 26N).

Em seguida, o modelo passa pela reamostragem do tamanho do pixel, recorrendo à ferramenta *r.resamp.interp,* definida pelo método *nearest neighbor*, o mais indicado na

[10] Ver Anexo D5, Programa 5: *Projection and resampling MDE to a 10 meter spatial resolution* (código programado no âmbito deste trabalho).

preservação dos valores de elevação. Segue-se a correção de imperfeições dos dados de elevação, para a qual se considerou o *r.hydrodem*, uma ferramenta *addon*[11] com grande utilidade na ponderação de critérios ligados à suavização de topos e bases de vertente. Por fim, a extração do modelo pelos limites da área de estudo, é feita a partir da definição da região e criação de uma máscara para limitar operações *raster* pela ferramenta *r.mask*, com recurso a um segundo *raster* de referência, seguido da exportação do MDE para fora da sessão GRASS.

5.3.2 *Software* desenvolvido para a filtragem dos dados de treino

No âmbito na produção de dados de treino por meio de aplicação de técnicas de filtragem, o objetivo principal centra-se na aquisição de informação relativa aos dados essenciais para a classificação de imagens, e a exclusão de toda a restante que se possa tornar enganosa.

À semelhança dos programas construídos nos exercícios anteriores, as rotinas produzidas respeitam essencialmente a utilização de ferramentas do pacote GRASS, auxiliada pelas funções do repositório GLASS. Quanto às bibliotecas introduzidas, o Multiprocessing foi o módulo de utilização mais comum com forte peso para o seguimento metodológico, no entanto, os módulos Numpy, GDAL/OGR, GeoPandas e OSR também se consideraram imprescindíveis. Esta fase é a mais complexa de todo o desenvolvimento conceptual, e igualmente a mais importante e decisora para o cumprimento dos objetivos impostos para a presente dissertação.

Para a conceção das diferentes etapas construíram-se rotinas através de conjuntos de programas, que permitem a divisão em 2 grupos constituídos por diferentes programas e um código final:

1. Grupo 1: 8 códigos programados, aliados à construção de grelhas vetoriais para uma filtragem auxiliada dos dados extraídas do OSM;
2. Grupo 2: 5 códigos programados, que remetem para o cálculo de índices radiométricos;
3. Código programado que conjuga os resultados dos processos de filtragem do Grupo 1 e 2 e aplica regras de filtragem para aquisição do produto final dos dados filtrados.

[11] Funções que necessitam de ser instaladas, uma vez que não fazem parte do pacote principal de ferramentas, expandido e adicionando recursos a um *software*.

5.3.2.1 *Grupo 1: Programas aliados às grelhas vetoriais*

A criação de grelhas consiste num conjunto de polígonos quadrangulares que delimitam individualmente cada célula de uma imagem de satélite, onde cada polígono tem em si um atributo associado. Para a conceção de grelhas vetoriais foi necessária a utilização de uma imagem de satélite para delimitar os vários pixéis, de forma a dar continuidade às fases de filtragem auxiliadas das informações recolhidas do OSM. O grupo de 8 programas executados recorre a um diverso conjunto de módulos, sendo que a principal a biblioteca implementada foi o Multiprocessing para execução de processos, tornando o desenvolvimento das diferentes cadeias lógicas menos dispendioso relativamente a períodos de execução e esforço computacional.

Numa primeira fase, a sequência inicial de códigos remete para o Programa 6[12], relativamente simples, mas exigente, dada a forte influência na performance de todos os programas seguintes. O Programa 6, constitui um bloco de código que estabelece a divisão das dimensões da área de estudo, em diferentes partes correspondentes de 500x500 pixéis. Para o desenvolvimento do programa foi dada como entrada, uma banda das imagens Sentinel-2 com resolução espacial de 10 metros, resultando num total de 24 ficheiros *rasters* distribuídos pela área de estudo definida. O código auxilia das bibliotecas Shapely, GDAL/OGR, OSR, Numpy e GeoPandas para aceder a informações pertinentes do ficheiro, como o sistema de coordenadas, o número de colunas e linhas, a extensão geográfica (canto superior esquerdo e canto inferior direito) e o número de partes menores definidas para a divisão da imagem. Neste ponto, obtêm-se os ficheiros *rasters* que compõem a extensão da área de estudo permitindo o avanço para o programa seguinte da sequência de códigos.

O Programa 7[13], segundo código criado para a execução ordenada aliada à filtragem dos dados a partir das grelhas vetoriais, beneficia dos *outputs* dos *rasters* obtidos no *script* anterior e procura realizar a conversão das respetivas informações matriciais para dados vetoriais. Graças à divisão da extensão da área de estudo por vários ficheiros o Programa 7, debruça-se na introdução da biblioteca Multiprocessing e ferramentas GRASS GIS programadas para efetivar os desígnios dos códigos construídos. Respeitando a evidência elucidada anteriormente, o Multiprocessing, revelou-se como a oportunidade de aplicar diferentes ferramentas em vários dados simultaneamente, sem correr o risco de problemas ligados a

[12] Ver Anexo D6, Programa 6: *Split area* (código programado no âmbito deste trabalho).
[13] Ver Anexo D7, Programa 7: *Make grid* (código programado no âmbito deste trabalho).

crashes, tornando o desempenho mais rápido e fluente, através da distribuição da carga de trabalho pelos vários núcleos, sem exigir muito da máquina. Portanto, o Programa 7 lista os 24 *rasters* obtidos previamente, em 4 processos diferentes, para a aplicação da ferramenta *v.mkgrid* a fim de produzir grelhas vetoriais para cada uma das partes representantes das dimensões da área de trabalho.

Depois de criadas todas as várias grelhas vetoriais, sem qualquer informação associada, o Programa 8[14], é executado no sentido de relacionar cada polígono da grelha com informação de ocupação do solo. O exercício atende à necessidade de intersetar cada quadrícula da grelha, com classes informativas obtidas da camada vetorial OSM após a execução do *software* OSM2LULC. Para isso, o código criado respeita a mesma divisão dos vários ficheiros vetoriais por 4 processos e realiza a interseção das informações das classes de ocupação do solo para cada grelha, com recurso da ferramenta *v.overlay* da biblioteca GRASS GIS. Esta é uma forma de compreender a área ocupada por determinada classe no interior dos polígonos das grelhas. Ainda assim, aproveitou-se para o mesmo *script*, a implementação de mais duas ferramentas GRASS, que servem de complemento à posterior análise estatística. Para tal, as *tools v.db.asscolumn* e *v.to.db* foram aplicadas para a criar uma coluna nas várias tabelas de atributos de tipo *double precision*, seguidas do cálculo de área ocupada por cada classe no interior das células da grelha. É importante referir que após a execução do Programa 8 uma grelha vetorial do grupo de 24 ficheiros acabou por ser excluído, uma vez que não houve qualquer interseção desta com a informação das classes de ocupação do solo, do produto do *software* OSM2LULC.

Os ficheiros resultantes da interseção, seguem para filtragem sendo submetidos ao Programa 9[15], tendo este código o propósito de agregar a informação comum das tabelas de atributos, e para esta abordagem, recorreu-se à biblioteca GDAL/OGR. A inexistência, no GRASS GIS, de uma ferramenta *dissolve* que faça a contagem de polígonos (análoga ao *dissolve* do ArcGis Pro) levou a incrementar uma estratégia que contorna este problema. Para este propôs-se uma resolução que passou pela utilização da biblioteca GDAL/OGR para agregar informação e realizar a contagem de modo a receber o número de classes que se encontravam no interior das células das grelhas intersetadas.

[14] Ver Anexo D8, Programa 8: *Intersection of grids with* OSM *classes* (código programado no âmbito deste trabalho).
[15] Ver Anexo D9, Programa 9: *Dissolve with* GDAL/OGR (código programado no âmbito deste trabalho).

Esta tomada de decisão foi importante para o próximo ponto, onde o Programa 10[16] concretiza a seleção, nas várias tabelas de atributos, dos campos com apenas uma classe de ocupação e uma área de ocupação total da célula associada ao atributo, para extração dessa informação. Foram fundamentais para o cumprimento deste exercício as ferramentas *v.db.join* e *v.extract* do módulo GRASS. Nesta etapa é descartada toda a informação restante, que remete para células mistas, com duas ou mais classes associadas a uma célula da grelha e atributos com ocupação da quadrícula inferior à totalidade.

O Programa 11[17], concebido a partir do resultado do código anterior, envolve uma atualização da informação relativa aos dados das classes de ocupação do solo nos ficheiros. A partir deste *script* pretende-se enriquecer os campos das tabelas de atributos através da associação do valor da legenda e da descrição das classes nos diversos atributos. Por este motivo, pegando nos *outputs* resultantes do Programa 10, realizou-se uma nova interseção com a camada vetorial das informações obtidas do OSM, após a conversão e harmonização de nomenclaturas. Sequencialmente, este código tira proveito da ferramenta *v.overlay* para a interseção entre os dados, de maneira a reter as referências do código da classe e respetiva descrição. Acede à *tool v.db.dropcolumn* para eliminar campos desinteressantes do resultado e adiciona um novo campo do tipo *double precision* às tabelas de atributos por meio do *v.db.addcolumn*. Nesta nova coluna são atualizadas as informação relativas às legendas das classes, através da ferramenta *v.db.update*, uma vez que o campo original que descreve o valor da legenda vinha por definição como tipo texto. Foi necessário criar e transferir a informação entre as colunas, para o avanço metodológico da próxima etapa, dado que, caso contrário, esta não seria possível.

De acordo com os processos de filtragem implementados a partir dos programas descritos até ao momento, os ficheiros na presente fase encontram-se em perfeita condição de sofrer a segunda conversão, para passar de informação vetorial para matricial. O Programa 12[18] tem em vista a integração e execução final dos ficheiros auxiliado da biblioteca Multiprocessing, introduzindo a ferramenta *v.to.rast* (GRASS GIS), que habilita a conversão do conjunto dados vetoriais para dados *raster*. Como parâmetros da conversão, esta exige a transferência dos valores da legenda das classes para o conjunto de ficheiros matriciais. Foi por este motivo,

[16] Ver Anexo D10, Programa 10: *Selection and extraction of attributes with 1 class and total occupied area* (código programado no âmbito deste trabalho).

[17] Ver Anexo D11, Programa 11: *Updating class legend values* (código programado no âmbito deste trabalho).

[18] Ver Anexo D12, Programa 12: *Vector grids to raster* (código programado no âmbito deste trabalho).

que se revelou fundamental converter valores das legendas das classes para um campo quantitativo no *script* anterior.

Numa fase final, tendo em consideração a implementação das etapas de filtragem de dados condicionada pela divisão da extensão da área de estudo, considerou-se essencial realizar a correção da respetiva repartição metodológica. Este é um fator importante, pois para dar continuidade ao exercício de filtragem, a agregação dos diferentes dados *raster*, limita dificuldades ao nível da exploração dos métodos subsequentes, assim como torna o trabalho mais simples, comparativamente a um exercício que utiliza constantemente um grande volume de ficheiros. A presente etapa, relaciona-se com o Programa 13[19], um código que visa aplicar operações de agregação de ficheiros matriciais a partir do módulo *r.patch* do *software* GRASS GIS. Este código tem como objetivo receber a lista do conjunto de *rasters*, independentemente do seu volume, e unir as informações num único ficheiro final. É a partir do respetivo ficheiro *raster* que se encontram condições de avançar para os outros métodos programados de filtragem de dados de treino.

5.3.2.2 *Grupo 2: Programas direcionados para o cálculo de índices radiométricos*

No âmbito do avanço metodológico direcionado à filtragem dos dados de treino, o cálculo de índices radiométricos partilha de práticas indispensáveis a exercícios desta natureza, pois a partir das suas fórmulas de cálculo, é possível reter informações que individualizam determinadas classes relativamente ao intervalo de valores compreendidos num determinado índice. Nessa mesma medida, foram implementadas boas práticas programadas para o cálculo individual dos índices radiométricos. Estes cálculos acarretam impactos positivos, na medida em que as suas informações, individualizam as classes em questão, e de certo modo, segmentam os dados em função do intervalo de valores ideias, como forma de seleção única dos dados mais importantes para o posterior treino.

Para a presente abordagem, referente aos índices radiométricos, as séries temporais das imagens Sentinel-2 foram submetidas a vários códigos programados, desenvolvidos a partir de um exemplo disponibilizado pela orientação, onde foi apenas necessário ajustar alguns parâmetros aos carecimentos das respetivas fórmulas dos índices. Neste sentido, como foram calculados 5 índices radiométricos para além dos selecionados com referência ao artigo

[19] Ver Anexo D13, Programa 13: *Aggregation of rasters* (código programado no âmbito deste trabalho).

Fonte, *et al.* (2020), os vários Programas 14, 15, 16 ,17 e 18[20] conduzem ao cálculo individual dos índices radiométricos a partir das diferentes bandas. A vantagem de utilizar um exemplo de código programado disponível, é que permite a replicação do mesmo em função do índice calculado, onde a adaptação das bandas facultadas como entrada, assim como a fórmula aplicada são os únicos fatores de distinção dos diferentes programas.

A estrutura dos códigos programados determina a integração das bibliotecas Numpy, GDAL/OGR e OSR, que, revela resultados muitos satisfatórios, pois utiliza uma função inicial para determinar o formato do ficheiro de entrada, associa *.tif* à extensão *GTiff*, para a posterior definição do formato mais apropriado para guardar os ficheiros. A função seguinte recebe um *array* Numpy, para criar um ficheiro *raster* através da biblioteca GDAL/OGR, define o sistema de coordenadas e realiza a transferência das informações dos *arrays* para *raster*. Isto foi importante para a transferência de dados entre o *array* e o *raster* final do índice radiométrico calculado. Por último, são chamadas as diferentes bandas a partir dos caminhos facultados no início do *script* para a análise das informações e conversão das bandas para *arrays* Numpy. Segue-se o cálculo da fórmula do índice radiométrico sendo os resultados guardados num *array*, que posteriormente é exportado para um ficheiro *raster* e desta forma, obtêm-se o resultado dos valores do índice radiométrico.

5.3.2.3 *Programa final de filtragem dos dados que conjuga Grupo 1 e 2*

Na mesma linha de pensamento, para o desenvolvimento de metodologias para a filtragem dos dados, a etapa final conjuga a fase oriunda dos programas do ciclo de trabalho aliado à criação das grelhas vetoriais, com as informações obtidas dos códigos programados para o cálculo de índices radiométricos. Nenhum exercício deste tipo, estaria completo sem uma estratégia que implemente as classes dos dados OSM filtrados através das grelhas vetoriais, bem como as informações espetrais conseguidas com o cálculo de índices radiométricos para realizar uma filtragem e seleção rigorosa, assistida da introdução de regras dos limiares de corte para as classes relativamente ao índice radiométrico associado.

A técnica de execução, segue uma cadeia lógica de instrumentos que visam, para além da introdução do ficheiro *raster* final obtido do Grupo 1 e os ficheiros dos índices radiométricos

[20] Ver Anexo D14, Programas 14, 15, 16, 17 e 18: *Example code for calculating the radiometric index* (código existente e adaptado no âmbito deste trabalho).

calculados através dos programas do Grupo 2, a adição das regras de filtragem. Estas regras associam as classes em relação aos valores (-1 e 1) que devem compreender para cada índice radiométrico, e o número de imagens (datas) que devem respeitar os respetivos limiares (*all* ou *at least one*), como o exemplo constatado na Tabela XIV, da secção 4.5. O Programa 19[21] foi desenhado para a seleção minuciosa das informações, alcançando apenas dados autênticos e eliminando aqueles que possam transmitir informações imprecisas das classes. A cadeia lógica do código programado, auxilia das bibliotecas GDAL/OGR, Numpy, OS e módulos GRASS. Para um primeiro momento os processos consideram-se simples, dado ao início de uma sessão GRASS e a introdução de todos os ficheiros *rasters*, respetivamente, o ficheiro final das classes resultantes da filtragem das grelhas vetoriais e os *rasters* de índices radiométricos calculados.

Em segundo lugar, com base nas regras definidas no início do código, são criados vários processos de máscaras binárias que indicam se um pixel deve ser considerado (valor de 1) ou não (valor 0) para cada combinação entre classe, índice radiométrico e data da imagem. Estas informações são guardadas num dicionário, para posterior acesso, na terceira fase do processo, onde são novamente produzidas máscaras binárias, considerando para cada classe e índice, a criação de ficheiros que indicam se o pixel deve ser ou não incluído, tendo em vista as máscaras obtidas em cada uma das datas consideradas.

Entre as várias máscaras obtidas até ao ponto anterior, seguindo o processo de filtragem, o *script* cria mais uma sucessão de máscaras binárias finais para cada classe, que posteriormente, são combinadas num último ficheiro com informações das classes conjugadas, a serem consideradas para a fase de treino. Para este exercício, e considerando a biblioteca de ferramentas GRASS, este código programado, foi apenas possível ser executado com recurso à ferramenta *r.mapcalc*, permitindo realizar o construção de fórmulas nos diferentes ficheiros *rasters*, com o auxilio de todas as regras de filtragem definidas.

Do presente programa resulta o ficheiro que retém todas as informações necessárias, relativas às diferentes classes resultantes dos vários processos de filtragem indispensáveis nas aplicações futuras de fases de treino do classificador, e posterior classificação de imagens, influenciando de forma decisiva no seu desempenho.

[21] Ver Anexo D15, Programa 19: *Final code for data filtering* (código programado no âmbito deste trabalho).

5.3.3 Integração de *software* desenvolvido por terceiros para o treino do classificador

O universo de algoritmos de *machine learning*, tem vindo a crescer e ganhar alguma popularidade, dado aos meios disponíveis para transmitir conhecimentos, ferramentas de código aberto para implementação e disponibilidade de acesso ao poder computacional.

Qualquer processo de classificação de imagem deve recorrer a um momento na fase de treino, que permita ao modelo a analise das respostas, bem como das informações relativas às classes de ocupação do solo, para que este possa aprender com os dados oferecidos. Para um classificador, espera-se sempre que exista uma boa capacidade de conhecer e aprender com as informações ao longo da fase de treino, com o intuito de na fase de classificação, este detenha uma forma de prever e classificar imagens com exatidão.

Em *python*, quando se ambiciona a utilização de algoritmos *machine learning*, para classificações supervisionadas de imagens, a biblioteca Scikit-Learn é considerada de eleição para qualquer exercício, dado o conjunto de algoritmos de classificação. É a partir desta que se recorre ao algoritmo RF para executar as tarefas que compõem este exercício.

Uma vez definidas as amostras de treino e o classificador a ser implementado, a operacionalização desta metodologia avança com recurso ao Programa 20[22], capaz de receber um ficheiro excel, com todas as informações referentes ao ficheiro dos dados de treino, bandas das imagens de satélite e MDE, e ainda, o número de árvores aleatórias (i.e. *ntrees*= 1000), para a posterior execução na fase de treino, auxiliada do algoritmo de classificação definido.

O respetivo programa desenvolveu-se, para o cumprimento do processo de treino do classificador por intermédio das bibliotecas OS, Scikit-Learn, Pandas, Numpy, Joblib e GDAL/OGR. Este respeita uma lógica que consiste em aceitar um ficheiro excel, e através de uma primeira função, realizar a sua leitura e converter a informação do ficheiro para Pandas *DataFrame*, onde são guardadas todas as características. Em seguida, é realizado o treino do classificador RF, atendendo aos ficheiros dos dados de treino introduzidos, à lista de caminhos construídos para os ficheiros *rasters* das várias bandas das imagens e MDE, bem como o caminho de saída dos diferentes modelos e o número de árvores aleatórias definidas

[22] Ver Anexo D16, Programa 20: *Random forest – Training model file* (código existente).

ao algoritmo. Após o treino do mesmo, são produzidos os ficheiros de modelo do treino, guardados no caminho saída, também indicado no ficheiro excel.

A partir dos resultados obtidos, com referência para os modelos treinados pelo classificador RF seguindo os diferentes conjuntos de dados e os parâmetros definidos, apresentam-se condições para levar a cabo a última etapa da metodologia de código programado, que se destina à classificação final das imagens de satélite.

5.3.4 Integração de *software* desenvolvido por terceiros para a classificação

Resta esclarecer a última sequência lógica explorada, que materializa a classificação de imagens e a produção de mapas de ocupação do solo. Na sequência do treino do classificador, e consequente produção de ficheiros de modelos treinados, tendo como base os dados de treino filtrados, as várias bandas que compõem as imagens satélite e informação relativa às elevações, aproveitou-se o Programa 21[23]. Este conduz à instância descrita acima, tomando por princípio aceder às informações treinadas pelo classificador e, através do mesmo, proceder ao processo de classificação.

O presente código programado foi o último construído e disponibilizado pelos orientadores, e respeita um conceito, com uma ótica idêntica ao *script* concebido para a etapa de treino do classificador (Programa 20). O respetivo código necessita mais uma vez do ficheiro excel, com os mesmos parâmetros definidos no programa anterior, para a correta execução da classificação. Este auxilia de bibliotecas OS, Joblib, GDAL/OGR Numpy e especificamente do módulo Pandas, para realizar a leitura dos padrões inseridos no ficheiro excel e converter a informação recolhida em tabelas Pandas *DataFrame*, iterando ao longo de um ciclo todas as linhas constituintes da folha do ficheiro *.xls*. A execução ordenada do Programa 21, procede de seguida à classificação de imagens, auxiliada do modelo RF para o número de 1000 árvores, com base nas informações disponibilizadas pelo ficheiro dado como entrada.

A classificação é realizada através da visualização das bandas *raster*, e do módulo GDAL/OGR, bem como do registo do número total das mesmas, para a posterior leitura dos valores dos pixéis e realização das previsões das classes por intermédio dos modelos treinados. Relativamente ao programa informático implementado para o treino do

[23] Ver Anexo D17, Programas 21: *Execute classification* (código existente).

classificador, a grande diferença, remete para a necessidade de leitura dos ficheiros dos modelos treinados. Estes são sustentados pelas características das diferentes classes dos dados de treino, as imagens submetidas à classificação, adotando as mesmas utilizadas no exercício de treino do classificador e ainda uma alusão aos caminhos e nomes para saída dos *rasters* classificados.

6 Resultados e Discussão

No âmbito da produção de mapas de ocupação do solo, a presente secção apresenta todos os resultados, aliados da discussão do desempenho dos métodos de classificação, que são mencionados pela ordem enunciada, separados por dois grupos. O primeiro grupo estabelece os produtos resultantes das diferentes classificações geradas a partir das combinações de dados de imagens de satélite e MDE, uma vez que se pretende analisar a confiança das classificações com maior destaque. O segundo, dependente das performances das classificações mencionadas anteriormente, determina as avaliações dos produtos obtidos das diversas etapas da metodologia para a geração dos conjuntos de dados de treino. Em síntese, os resultados para a presente dissertação são distinguidos como:

1. Resultados: Análise de classificações obtidas a partir das combinações de dados de imagens e MDE para os conjuntos de dados de treino F1 e F2;
2. Resultados: Em virtude do melhor desempenho dos 1. Resultados, realiza-se a análise das classificações adquiridas através dos diferentes conjuntos de dados treino produzidos F0, F1, F2, F3 e F4.

6.1 Resultados: Classificação das combinações de dados de imagens e MDE auxiliado dos dados de treino F1 e F2

Neste primeiro momento de análise das classificações de mapas de ocupação do solo, faz-se a comparação da classificação, a partir do algoritmo RF no âmbito da produção de mapas, com recurso a diferentes combinações de imagens de satélite e dados de elevação. Os vários resultados envolvem a utilização combinada das bandas Sentinel-2, Landsat-8 e informações relativas ao MDE, como forma de procurar combinações com os melhores resultados em relação à interpretação das características da superfície terrestre. As diferentes experiências, são repartidas em dezasseis classificações e podem ser identificadas com as respetivas denominações na Tabela XXII.

Tabela XXII – Classificação a partir da combinação de dados de imagens de satélite e MDE.

Nome	Características
classi_D1_F1	4 bandas Sentinel-2 de 10 m e dados de treino F1
classi_D1_F2	4 bandas Sentinel-2 de 10 m e dados de treino F2
classi_D2_F1	10 bandas Sentinel-2 e dados de treino F1
classi_D2_F2	10 bandas Sentinel-2 e dados de treino F2
classi_D3_F1	4 bandas Sentinel-2 de 10 m e Landsat-8 e dados de treino F1
classi_D3_F2	4 bandas Sentinel-2 de 10 m e Landsat-8 e dados de treino F2
classi_D4_F1	10 bandas Sentinel-2 e 11 bandas Landsat-8 e dados de treino F1
classi_D4_F2	10 bandas Sentinel-2 e 11 bandas Landsat-8 e dados de treino F2
classi_D5_F1	4 bandas Sentinel-2 de 10 m e MDE e dados de treino F1
classi_D5_F2	4 bandas Sentinel-2 de 10 m e MDE e dados de treino F2
classi_D6_F1	10 bandas Sentinel-2 e MDE e dados de treino F1
classi_D6_F2	10 bandas Sentinel-2 e MDE e dados de treino F2
classi_D7_F1	4 bandas Sentinel-2 de 10 m, 4 Landsat-8 e MDE e dados de treino F1
classi_D7_F2	4 bandas Sentinel-2 de 10 m, 4 Landsat-8 e MDE e dados de treino F2
classi_D8_F1	10 bandas Sentinel-2, 11 bandas Landsat-8 e MDE e dados de treino F1
classi_D8_F2	10 bandas Sentinel-2, 11 bandas Landsat-8 e MDE e dados de treino F2

É importante reconhecer que, ao longo do desenvolvimento do exercício da classificação para os primeiros resultados, que tem em vista a procura dos produtos combinados com melhor exatidão temática, foram inicialmente excluídas de qualquer combinação a banda 01, banda 09 e banda 10 de 60 metros, do Sentinel-2, devido ao insignificante contributo de informação proporcionado por estas. Como forma de preservar o máximo de informações possíveis nas imagens de satélite para classificação, como foram produzidas duas máscaras de nuvens, uma para apenas imagens Sentinel e uma segunda máscara para imagens Sentinel e Landsat, foi considerada uma forma de reduzir a exclusão de pixéis em determinadas classificações.

Para as combinações de bandas que envolvem dados Landsat-8 e Sentinel-2, aplicou-se uma máscara de nuvens mediante da distribuição apresentada em todas as datas das várias imagens, resultando num número superior de áreas "buraco", enquanto para a classificações, onde são usados apenas os dados Sentinel, a máscara de nuvens restringiu-se unicamente para as datas de imagens do respetivo satélite, mitigando o número de células excluídas. Ademais, no decorrer da etapa de validação dos mapas através da segunda estratégia, direcionada para a comparação entre mapas, a informação referente à classe temática *wetland* encontrava-se em falta. Tendo este ponto em consideração, esta categoria foi excluída de todos mapas de ocupação do solo produzidos, tendo em vista as condições restritas de

associação entre os mapas e por comprometer os valores nas medidas de validação selecionadas.

6.1.1 Resultados obtidos: explicação da distribuição geográfica das classes nas diferentes classificações

O presente subcapítulo é alusivo à interpretação visual (Figura 16), dos mapas gerados pelas diferentes combinações de *features*. Este assume a principal meta de encontrar uma explicação para as diferentes classificações dos mapas de ocupação do solo produzidos, no que diz respeito ao respetivo contributo da utilização de um maior ou menor número de bandas de imagens de satélite e MDE. Neste sentido, é igualmente importante, aliar ao mencionado uma explicação para a distribuição geográfica das classes temáticas nas diferentes classificações.

Na figura abaixo (Figura 16), ilustram-se os resultados das classificações apenas para as combinações de *features* classi_D1, classi_D4, classi_D6 e classi_D8, uma vez que decorreu um número de experiências muito elevado, tirando-se assim partido de representar os mapas de ocupação do dolo que se consideraram mais indicados, e não a totalidade dos mesmos. De modo geral, os diferentes mapas são classificados por seis classes de ocupação distintas, e numa primeira impressão facilmente se distingue a diferença entre os produtos originários dos dados de treino F1 para F2.

Os mapas gerados a partir dos dados F1 revelam semelhanças em vários parâmetros, em conformidade com individualização espacial das classes temáticas, contudo para as classificações de F2 as diferenças entre a combinação de *features* acabam por ser reconhecidas. Para os dados de treino F1, a variação visualmente interpretada, remete para a classe *sealed surface*, concretamente para a região urbana, na zona mais a sul da área de estudo, onde a concentração de edificado aumenta de classi_D1 para classi_D8. Além disso, classi_D1 aparenta uma distribuição geográfica da classe *partly vegetated or non-vegetated*, mais sentida do que nas restantes classificações, tendo em conta que os espaços artificializados são acompanhados de imensa informação sobre solo descoberto. Em referência para a distribuição geográfica das restantes classes informativas, os mapas produzidos a partir do *dataset* F1, remetem para uma individualização temática idêntica, sem reconhecimento de alguma outra diferença. Este fator parece estar associado ao particular grau de rigor das classificações produzidas pelo dado de treino em questão, para as diferentes combinações de *features*, estabelecendo uma distribuição temática indistinguível.

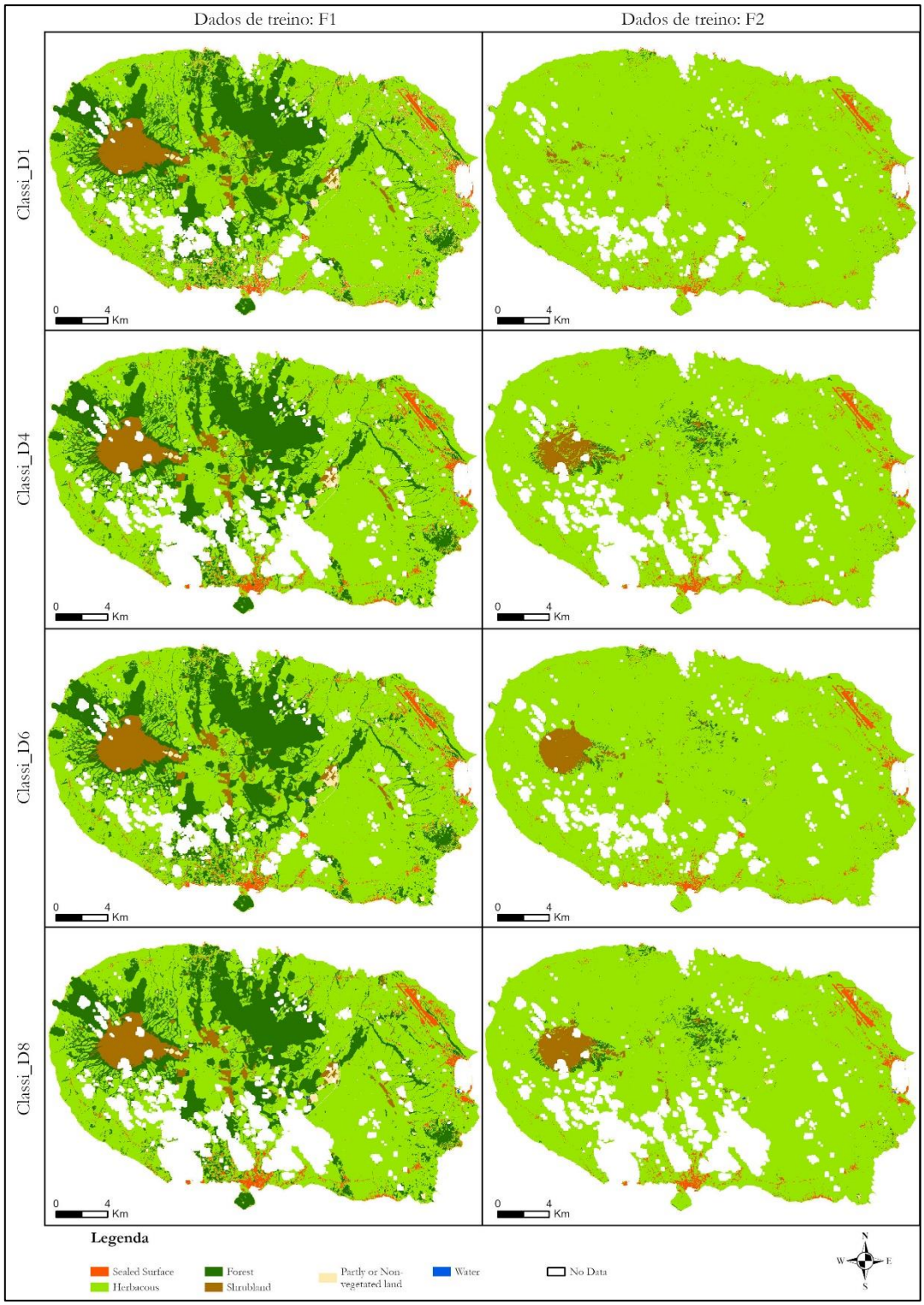


Figura 16 – Resultados das classificações de combinações de *features*.

Ainda no que concerne à Figura 16, as classificações geradas pelos dados de treino F2, espelham um contraste eminente na distribuição espacial das classes de ocupação do solo, relativamente às diferentes combinações de *features*. De um modo genérico, comparando com os resultados inerentes aos dados F1, as informações incluídas no *dataset* F2 geraram algumas incertezas na distribuição da ocupação do solo em relação à realidade, no sentido que promoveu a criação de mapas com um nível de detalhe comparativamente inferior. Todavia, independentemente do mencionado, estes mapas são os únicos que permitem realizar a distinção visual, da distribuição da ocupação do solo e, por isso, revelam-se fundamentais para a discriminação do peso atribuído à combinação de diferentes *features*.

Neste contexto, a organização das classes temáticas entre classificações produzidas a partir de F2, remete para uma distribuição espacial semelhante das informações referentes a *herbaceous* e *sealed surface*. Essa semelhança foi devidamente identificada a partir das combinações de *features*, classi_D4, classi_D6 e class_D8, exceto pela classificação classi_D1, que ostenta uma classificação completamente distinta das restantes. Este cenário pode estar inteiramente relacionado com o número inferior de *features* determinado para esta classificação, tendo em conta que recorre somente às quatro bandas de 10 metros do Sentinel.

Para a organização espacial das classes relativas às informações de *forest* e *shrubland* as divergências entre as classificações produzidas por F2, são consideradas como as mais sentidas, uma vez que o nível de detalhe da sua distribuição é complexo. A devida individualização das classes mencionadas, remete para um conjunto de *features* com contributos de informações capacitadas para a distribuição correta das ocupações do solo. Neste sentido, a partir da interpretação visual dos mapas, as combinações de *features* de classi_D4 e de classi_D8, aparentam apresentar um rigor superior da distribuição espacial das classes *forest* e *shrubland*, comparativamente às restantes classificações. A classificação classi_D6, apresentou da mesma forma as respetivas classes de ocupação, contudo o seu nível de detalhe, na distribuição geográfica das classes, foi subjacente.

Através da análise das classificações dos dados F2, pode determinar-se que na distribuição geográfica das classes, os melhores desempenhos vão ao encontro de classi_D4 e classi_D8, uma vez que se caracterizam com uma combinação de *features*, superior às restantes. O único fator que as distingue é a inclusão do MDE, ainda assim, ambos tiram proveito de informações Sentinel e Landsat. Para classi_D6, a grande diferença passa por não usufruir de informações Landsat, determinando de forma decisiva a sua classificação.

6.1.2 Resultados da validação com amostra aleatória

Na resolução da tarefa, tendo em vista a validação pela amostra aleatória e as medidas de validação complementares, num primeiro momento, de acordo com a Tabela XXIII, os desempenhos das classificações, estabeleceram exatidões discordantes, aquando da análise estipulada entre os dois dados de treino. Contudo, aliado a uma observação individual pelos *datasets*, para o treino F1, os valores de exatidão obtidos, não aparentam assimetrias significativas, assim como as percentagens de exatidão de F2. No entanto, enquanto o primeiro dado de treino apresenta uma diferença de 1%, os dados F2 foi de 3%.

O mencionado vai ao encontro daquilo que foram as distribuições geográficas das classes nas diferentes classificações, uma vez que para os dados F1 os mapas produzidos mostraram-se idênticos, impossibilitando a respetiva deteção visual da variação de ocupação de classes. No que concerne às classificações de F2, como foi possível realizar uma identificação na alteração do solo, as exatidões temáticas registas são o reflexo disso.

Considerando a elevada exatidão temática registada pelos dados de treino F1, comparativamente ao *dataset* F2, é possível demonstrar que a base dos resultados está na forte qualidade das regras de filtragem estabelecidas no artigo de Fonte, *et al.* (2020). Todavia, dado que esta secção se destina à análise comparativa da combinação de *features,* a referência aos dados de treino será apenas esclarecida posteriormente no documento.

Tabela XXIII – Exatidão temática da combinação de *features* validada com amostra aleatória.

Estratégia de validação: Amostra de pontos aleatórios		
1.Resultados	Exatidão temática	
Combinação de *features*	F1	F2
classi_D1	84.83%	61.67%
classi_D2	85.50%	62,83%
classi_D3	85.67%	62.33%
classi_D4	85.17%	64.67%
classi_D5	85.33%	64.17%
classi_D6	85.67%	64.00%
classi_D7	85.67%	64.17%
classi_D8	85.17%	64.67%

Na generalidade, a utilização de apenas 4 bandas de 10 metros do Sentinel, está associada a valores de exatidão dos produtos menos vantajosos, tanto para os dados de treino F1, com 84.8%, como para os dados de F2 de 61.6% de exatidão temática. A nível de interpretação visual, as percentagens revelam o esperado, dada a distribuição geográfica das classes por parte da combinação de *features*, mostrando-se limitada em relação às restantes. A utilização de um número superior de *features* para combinação de dados, possibilita a integração e correlação de informações de várias bandas, bem como informação altimétrica, acarretando de uma subida hierárquica dos resultados em termos de exatidão temática. Posto isto, assiste-se a uma soberania das classificações classi_D8 e classi_D4, nos dados de treino F2, com a mesma exatidão de 64.6%. Para os dados F1, são as combinações de classi_D7, class_D6 e classi_D3 que, igualmente, estabelecem as percentagens mais altas de 85.6%.

Em referência para o *dataset* F1, é importante apontar o crescimento desacelerado da medida de validação, em comparação aos dados F2, onde se assiste a uma diferença progressiva. Todavia, de modo geral, a exatidão temática das combinações de *features*, acabam por rondar valores muito próximos, distinguindo as percentagens, apenas, por pequenas décimas. Das classificações analisadas até ao momento, é pertinente destacar que as combinações de *features* com melhor desempenho da medida de validação, remetem para os grupos de bandas Sentinel e MDE ou bandas Sentinel, bandas Landsat e MDE.

Com o intuito de reafirmar ideias e analisar o suporte estatístico da combinação de *features*, foi construído um gráfico de linhas síntese, em alusão à relação entre os valores de exatidão temática e o número de *features* introduzidas em conjunto nas determinadas classificações. Para tal, a abaixo (Figura 17) tenciona resumir a informação apresentada.

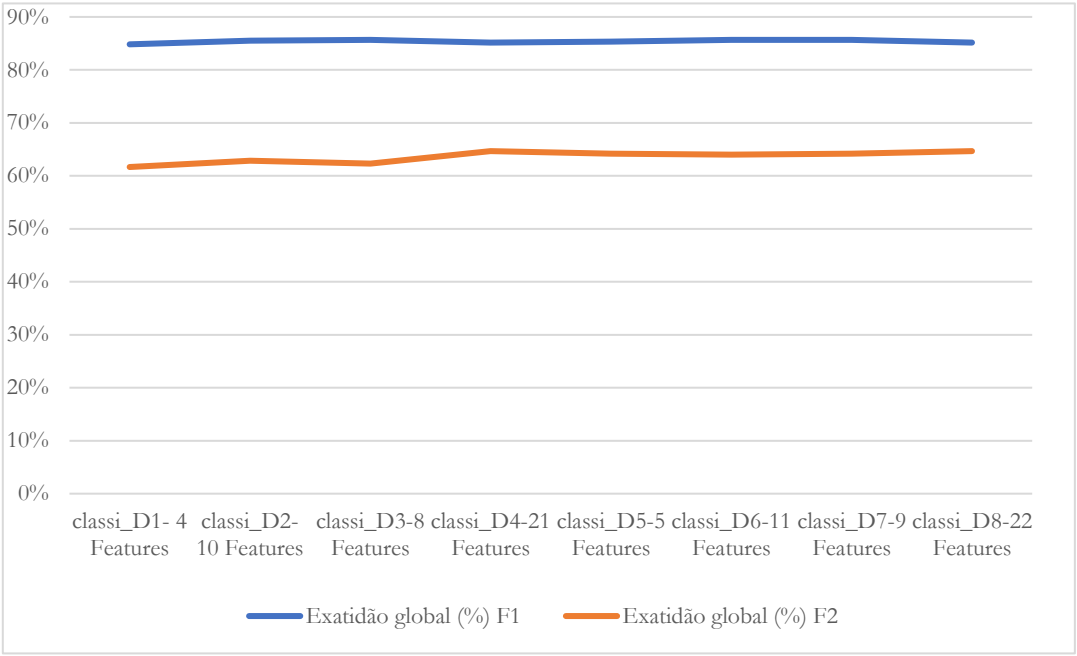


Figura 17 – Relação entre exatidão temática e número de bandas utilizadas na estratégia de validação: Amostra de pontos aleatórios

Revela-se uma tendência de variação de valores de exatidão obtidos dos dados F1 e F2, em detrimento do número de informações utilizadas nas classificações, sendo possível afirmar dois pontos essenciais. Relativamente aos dados F1, a linha apresenta variações regulares e constantes, uma vez que não se assinala nenhuma oscilação. Este cenário, afirma que, independentemente da utilização combinada de um número maior ou menor de *features*, a exatidão temática, é assegurada através de uma variação regular de percentagens análogas. Considera-se como um comportamento expectável, tendo em conta que a diferença entre a exatidão da combinação de informações, foi de apenas 1% para o treino F1.

Ainda que o comportamento da linha de F1 seja considerado regular, é possível assinalar alguns pontos de oscilação para os dados de treino F2. A partir do gráfico de linhas, foi possível determinar, para as respetivas classificações, algumas variações ainda que pouco evidentes. O percurso da linha assumiu-se normal nas duas primeiras classificações, sendo apenas sentida, no ponto de transição entre classi_D3, e a classificação de classi_D4, uma ligeira subida. Neste caso concreto, está-se perante a transição de uma classificação com apenas 8 *features* e outra de 21 informações combinadas, promovendo a ideia de uma possível relação positiva entre exatidão temática mais alta associada a um número elevado de *features* combinadas.

Como complemento ao estudo do primeiro grupo de resultados, e de forma sustentar as ideias propostas até ao momento, no decorrer da análise das classificações, optou-se por uma análise rigorosa, auxiliada do estudo de outra medida de validação. Neste seguimento, realizou-se a comparação das diferentes classificações, para avaliar o seu peso decisivo na individualização das classes temáticas de ocupação do solo. Para tal, as próximas Figuras 18 e 19, têm como objetivo apresentar de forma sucinta, os diferentes padrões da medida *f1-score*, individualmente para cada classe temática, em detrimento da classificação de informações combinadas.

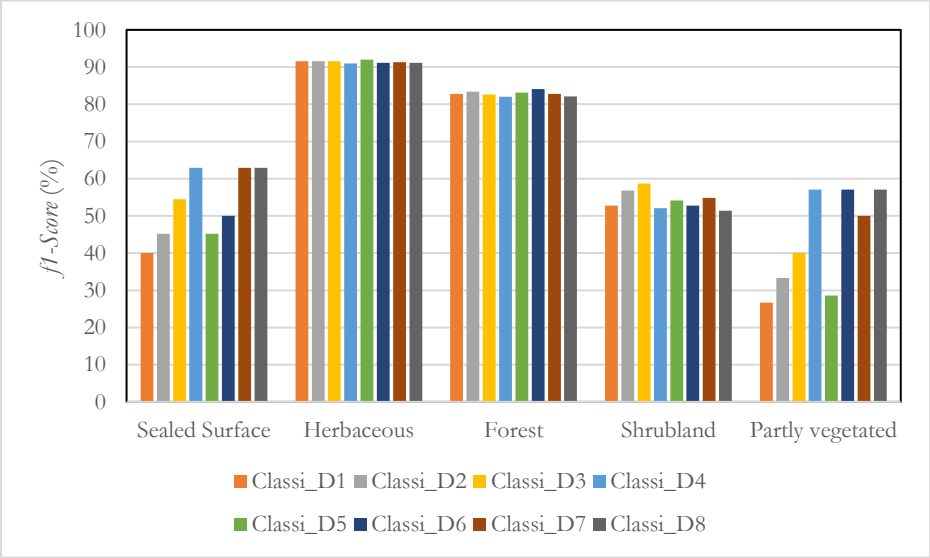


Figura 18 – Valor f1-score para cada classe F1. Estratégia de validação: Amostra com pontos aleatórios.

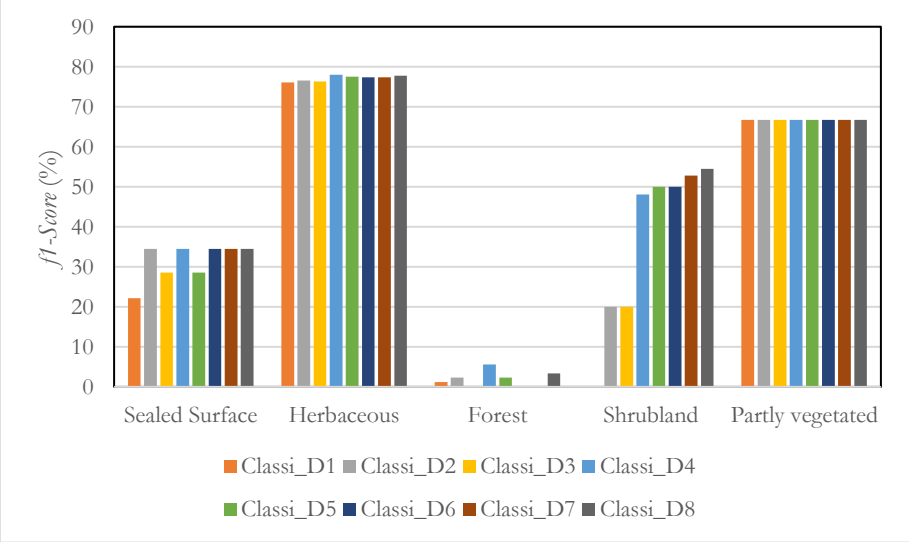


Figura 19 – Valor *f1-score* para cada classe F2. Estratégia de validação: Amostra com pontos aleatórios.

De um modo genérico, a partir dos gráficos de barras, é possível retirar a primeira conclusão, que se dedica à falta de uma sexta classe de ocupação do solo. A partir da secção que se destina à explicação da distribuição geográfica das classes, pensava-se que a referida classe em falta, com informações sobre *water*, iria ser contabilizada, contudo isso não aconteceu. Esta situação parece debruçar-se na problemática aliada à utilização de uma validação com uma amostra de pontos distribuídos de forma aleatória, onde nenhum ponto incidiu na respetiva classe.

Numa perspetiva de análise individual dos valores obtidos pela medida *f1-score*, pode determinar-se, para a classe de *sealed surface*, uma melhor individualização da classe por parte dos dados combinados referentes a classi_D4, classi_D7 e classi_D8. Nas classificações destas combinações de *features*, com recurso aos dados de treino F1, os respetivos grupos referidos, apresentam uma superioridade considerável em relação às restantes, igualados com uma percentagem de 63% de *f1-score*. Para os dados F2, estas classificações registam, do mesmo modo, os valores mais elevados, no entanto classi_D6 e classi_D2 apresentam-se com o mesmo valor da medida de 35%.

A partir da análise dos resultados sintetizados nos dois gráficos, a próxima classe, direcionada para as informações sobre *herbaceous*, espelha a ocupação do solo mais comum na área de estudo, devido à sua evidência como a classe com percentagens de *f1-score* mais altos, registados pelas diferentes classificações. Possibilitando a ligação das informações apresentadas pelos dados de treino F1 e F2, mesmo que estes se distinguem com uma certa variação, é exequível assinalar que todas as classificações apresentaram uma excelente capacidade de distinção da respetiva classe. Esta ideia é alimentada, pelo fato, das classificações terem apresentado valores de *f1-score* quase idênticos e elevados, distinguindo-se apenas por pequenas décimas, as combinações de *features* classi_D8 e classi_D4, com 77.8% e 78% respetivamente, para os dados F4. Para os *datasets* F1, classi_D5 merece destaque por se ter verificado um valor de 92%.

Para o caso específico dos valores obtidos para a classe *forest* nas classificações que tiram proveito dos dados F2, a informação da respetiva figura comprova a visão relativa à representação da classe informativa nos mapas apresentados na secção da distribuição geográfica das classes. Nestes mapas foram apenas distinguidas e identificadas pequenas áreas da classe informativa e, portanto, o valor de *f1-score* não poderia ser elevado. Pensa-se mais uma vez, que este fator em muito se deve às regras de filtragem aplicadas nos dados F2, com uma ampla exclusão informativa da classe temática de floresta dos dados de treino. Contudo,

na análise dos valores de *f1-score* assinalados para os dados F1, determinou-se classi_D7, como a classificação com a percentagem mais elevada, de 84%, caracterizando-se no entanto, as restantes classificações com valores também altos e semelhantes. Na situação particular de F2 que reflete a ideia sobre aquilo que foi mencionado, classi_D4 e classi_D8 permitiram uma melhor individualização da classe *forest*, superando inconsistências que as restantes classificações não alcançaram.

No que concerne às duas últimas classes de ocupação do solo, os resultados evidenciados pela medida *f1-score*, remete para um quadro atípico do que foi analisado até ao momento. Para estas classes, de *shrubland* e *partly vegetated or non-vegetated*, verifica-se uma forte discordância do que foi registado para os dados de treino F1, comparativamente com o que F2 apresentou nas respetivas informações de ocupação do solo. Para a classe *shrubland*, no treino F2, classi_D4 revelou-se com uma forte superioridade das restantes classificações, devido ao seu valor de 55% de *f1-score*, enquanto classi_D3 e classi_D2, com o valor da medida mais baixo. Quando se centra a análise de *f1-score*, para os resultados obtidos a partir do treino F1, o valor mais alto registado é direcionado para a classificação classi_D3, que assinalou uma das piores percentagens em F2. Para além disso, em referência à classe *partly vegetated or non-vegetated* existe uma avaliação da medida *f1-score*, exatamente igual para as diferentes classificações, quando se utilizam os dados de treino F2. Este cenário torna-se contestável, uma vez que se está perante uma análise particular de validação de uma amostra totalmente aleatória. Considerando o mencionado, espera-se que na próxima secção que se dedica à validação de resultados com comparação entre mapas, se encontrem respostas para a análise da medida *f1-score* para as respetivas classes.

6.1.3 Resultados da validação com comparação entre mapas

Para a dar continuidade à análise de resultados, a presente secção tem como objetivo realizar a validação das classificações produzidas a partir de diferentes combinações de *features*, aliada a uma estratégia de validação que recorre à comparação dos mapas produzidos com informações CLC+ BB. Esta etapa procura uma validação mais coerente, procurando explicações relativamente a temas aos quais a validação com uma amostra aleatória, não permitiu encontrar resposta. Neste sentido, toda a análise realizada anteriormente, será executa da mesma forma através da comparação com a referência.

Apontando para a Tabela XXIV, referente às exatidões temáticas das classificações registadas para um mapa de referência, é inicialmente reconhecida uma superioridade dos valores de exatidão para F2, comparativamente aos resultados obtidos com a estratégia de validação com uma amostra aleatória. Para os dados de treino F1, as exatidões temáticas das classificações também se distinguem entre duas estratégias de validação, no entanto de uma forma menos notória. Conferindo um certo grau de confiança para a presente validação, os resultados das exatidões temáticas assinalados por esta estratégia remetem, de forma mais clara, para a ideia que tem vindo a ser desenvolvida ao longo da discussão de resultados.

Tabela XXIV – Exatidão temática da combinação de *features* validada com comparação entre mapas.

Estratégia de validação: Comparação entre mapas		
1.Resultados	Exatidão temática	
Combinação de *features*	F1	F2
classi_D1	84.31%	69.51%
classi_D2	84.91%	69.97%
classi_D3	85.06%	69.65%
classi_D4	85.21%	71.41%
classi_D5	84.63%	70.65%
classi_D6	84.96%	70.85%
classi_D7	85.18%	70.56%
classi_D8	85.16%	71.52%

Como se pode verificar, na estratégia anterior, as combinações de *features* que utilizam um número reduzido de bandas dos diferentes satélites, ou não usufrua de informação altimétrica, caracterizam-se como classificações com valores de exatidão mais baixa. Esta discussão surgiu na secção anterior, aliada às exatidões temáticas e, entretanto, a presente validação veio a comprovar o respetivo o ponto. Nesta perspetiva, reconhece-se classi_D1, sendo a que combina bandas de um único satélite (Sentinel), é também a que regista a percentagem mais baixa das classificações para ambos os dados de treino F1 e F2.

A respeito da análise singular das exatidões registadas para as diferentes classificações por intermédio dos dois *datasets*, F1 apresenta exatidões temáticas muito próximas, registando apenas pequenas variações. Para os dados de treino F2, a variação das exatidões temáticas acaba por se considerar mais ampla, evidenciando com maior coerência, as classificações com o melhor desempenho.

Tendo por base a segunda ideia que tem vindo a ser reformulada, no sentido da relação entre o número de *features* utilizadas e a sua determinação nas exatidões temáticas das classificações, as informações apresentadas na tabela acima aludem para esse sentido. A partir das exatidões temáticas das classificações, produzidas através do treino F2, as combinações de um número superior de *features*, remetem devidamente para um valor superior de exatidão. Estas duas classificações são classi_D4 e classi_D8, com percentagens quase idênticas, de 71.4% para a primeira e de 71.5% para a segunda classificação. Fazendo alusão aos dados de treino F1, repete-se o mesmo cenário, com classi_D4 a apresentar o valor mais alto de exatidão, com 85.2%, seguido de classi_D7 e classi_D8 de 85.1%.

De forma a permitir confirmar a realidade em torno da relação entre o número de *features* utilizadas, correlacionando-as com informações que podem determinar a exatidão temática mais exata das classificações, construiu-se um gráfico de linhas que interliga as duas medidas.

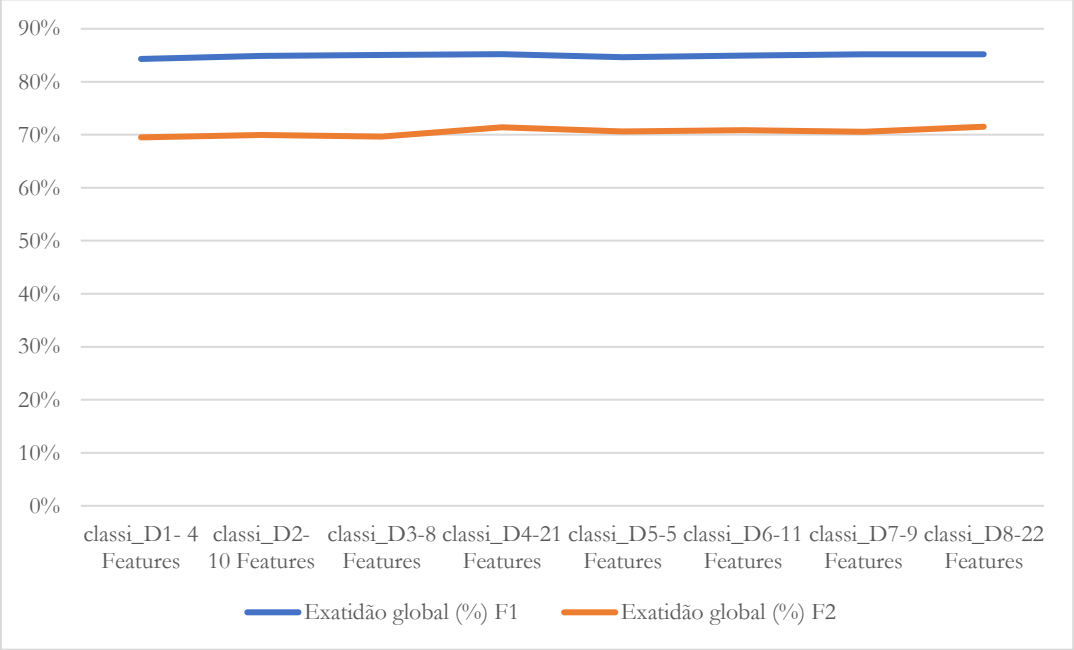


Figura 20 – Relação entre exatidão temática da classificação e número de bandas para a estratégia de validação: Comparação entre mapas.

Considerando as medidas expressas, pela Figura 20, que diz respeito ao gráfico de correlação entre o número de *features* usadas na combinação das classificações e as devidas exatidões temáticas atribuídas, assumiu-se uma particular relação entre ambos. O comportamento da linha relativo a este *dataset*, assemelha-se de forma determinante ao que se assinalou na validação com a amostra aleatória. Assim como na estratégia anterior, verifica-se uma ligeira

subida da exatidão temática, no momento de transição da classificação classi_D3 para classi_D4, assim como na evolução de classi_D7 para classi_D8, cenário que não foi detetado na validação anterior. Relativamente às exatidões temáticas proporcionadas pelos dados de treino F1, mantêm-se as semelhanças nos valores, derivado de resultados extremamente próximos.

Com o intuito de completar a análise, de forma a assegurar a integridade valorativa para as classificações com um número superior de combinações de *features*, a medida estatística de *f1-score* foi também calculada para a presente validação. Através da construção de gráficos de barras representados pelas Figuras 21 e 22, objetivou-se alcançar uma conclusão consistente e estruturada, bem como responder a pontos da medida de *f1-score*, que a estratégia de validação com amostra aleatória não possibilitou.

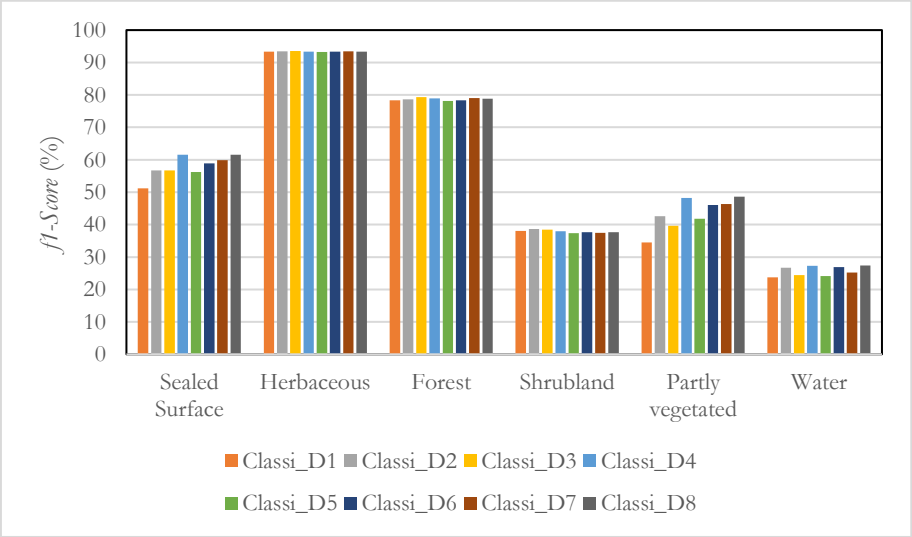


Figura 21 – Valor *f1-score* para cada classe F1. Estratégia de validação: Comparação entre mapas.

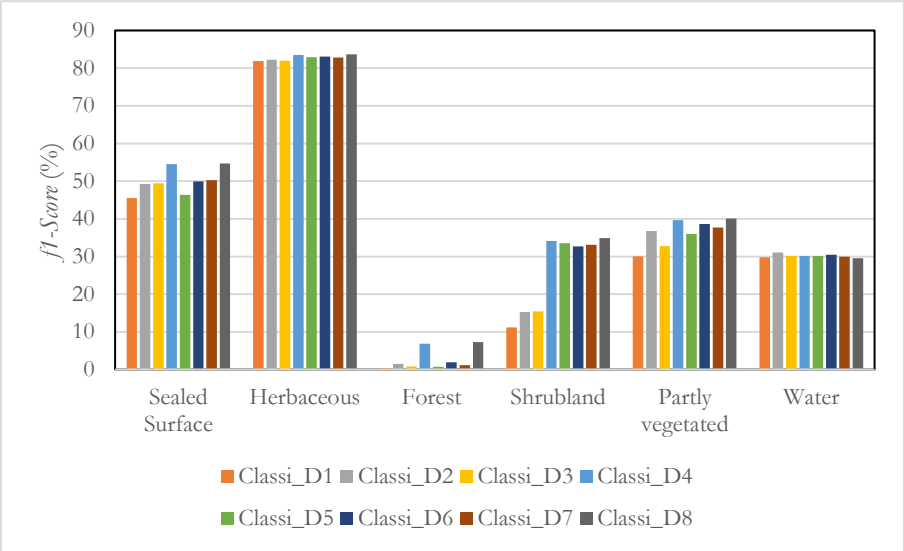


Figura 22 – Valor *f1-score* para cada classe F2. Estratégia de validação: Comparação ente mapas.

Tal como verificado na secção anterior, os valores da combinação de *features* obtidos para a medida *f1-score,* revelaram uma distribuição idêntica nas diferentes classes temáticas. Para além disso, na introdução dos dados de treino F1, os valores revelaram-se superiores na análise dos resultados da medida, em comparação com as percentagens de *f1-score* inerentes ao treino de F2.

Numa interpretação inicial, a respeito dos valores *f1-score* registados para a classe *sealed surface*, as diferentes combinações de *features*, revelaram uma boa capacidade para a classificação desta informação, apresentando valores intermédios, através dos dados de treino F1, bem como F2. Todavia, destacaram-se duas classificações, dadas as percentagens mais elevadas registadas pela medida *f1-score*, sendo estas as de classi_D4 e classi_D8, para os dois dados de treino utilizados. A partir dos dados de treino F2, classi_D4 assinalou uma percentagem de 46% e classi_D8 uma medida de 58%. Para o *dataset* F1, classi_D4 e classi_D8 igualaram-se com um valor de 62%.

A partir de uma análise individual, para a classe *herbaceous*, assim como relativamente ao ponto de validação precedente, as combinações assinalaram a respetiva classe com os melhores valores de *f1-score*. Como referido, este fator advém da ampla cobertura da área de estudo por esta tipologia de ocupação do solo, tornando-se numa classe de classificação mais simples. É relevante realçar, que os valores de *f1-score* verificados para o treino F1, mostraram percentagens idênticas para todas as classificações executadas. Para a maioria das combinações de *features* foi atribuído um valor de 93.4%, sendo considerada uma "percentagem quase perfeita". Em relação à utilização dos dados de treino F2, não é possível

verificar o mesmo quadro e, portanto, os valores de *f1-score* mais elevados registam-se em classi_D4 e classi_D8, com uma percentagem correspondente de 84%.

As informações dos valores de *f1-score* para a classe *forest* assumem, tal como se verificou na validação anterior, um desempenho insuficiente das classificações auxiliadas dos dados de treino F2, compensado posteriormente pelas classificações adequadas aliadas aos dados F1. Para a respetiva amostra de dados F1, a grande maioria das classificações proporcionadas desencadearam uma identificação assertiva das informações sobre esta classe de ocupação, com destaque para o valor de 79% assinalado pela maior parte das combinações de *features*. Para o caso concreto determinado pelos dados de treino F2, devido ao tamanho da amostra, considerado insuficiente para individualização informações sobre florestas, grande parte das classificações aplicadas remeteram para percentagens baixas na medida de validação *f1-score*. Porém, a combinação de um número superior de *features*, permitiu esquivar-se à generalização verificada. Neste sentido, destacam-se as classificações de classi_D4 e classi_D8, que devido ao número superior de dados combinados, permitiram registar valores de *f1-score* de 7%.

Apesar das classes de ocupação do solo, de certo modo revelarem uma classificação acessível por parte das diferentes combinações de dados, justificada pelas percentagens satisfatórias de *f1-score*, seguem-se as próximas classes temáticas que apontam para uma individualização mais exigente. Estas classes fazem referência à ocupação *shrubland* e *partly vegetated or non-vegetated* e para além de se caracterizarem pela ampla variedade espetral, o que significa que as suas assinaturas podem cruzar-se com outras classes, são importantes determinantes de boas práticas de classificação aliadas à combinação de *features*. Ainda assim, uma vez que a estratégia de validação precedente se apresenta inconsistente, nesta secção apresenta-se aquela que permitiu maior confiabilidade aos resultados.

Prosseguindo, os valores encontrados nos gráficos de barras, demonstraram que as combinações de dados superiores remetem, de um modo geral, para percentagens mais altas na medida de validação *f1-score*. No que diz respeito à classe *shrubland*, é possível determinar para os dados de treino F1, resultados semelhantes para todas as classificações, porém para o treino F2, foi assinada uma superioridade respetiva. A partir das percentagens de *f1-score* reveladas pelo *dataset* F2, concluiu-se que o uso de um número superior de *features* conduz a valores mais elevados, em comparação a padrões das combinações de dados que utilizem bandas de um único satélite, caso das classificações classi_D1 e classi_D2. É neste parâmetro que classi_D4 e classi_D8 se destacam das outras classificações, devido aos respetivos padrões de 34% e 35% de *f1-score*.

Na mesma linha de pensamento, a classe *partly vegetated or non-vegetated* apresenta-se com uma distribuição de valores *f1-score* bastante idênticos, tanto na utilização dos dados de treino F1, como F2. Para os dois *datasets*, os melhores valores de medida remetem para as mesmas combinações de *features,* apontando novamente para classificações que utilizam maior número de dados combinados. Na respetiva classe de ocupação do solo, faz-se destaque às classificações classi_D4 e classi_D8 com hegemonia das restantes. Para os dados de treino F1, o valor de *f1-score* verificado, foi de 48% para a primeira classificação e 49% para a segunda. Já o *dataset* F2, classi_D4 assumiu um valor da medida de 34% e classi_D8 destaca-se com um *f1-score* de 35%. Em conformidade com a classe *shrubland*, os resultados obtidos para a ocupação de solo descoberto, promovem a correlação do número de *features* combinadas e o desempenho de resultados positivos da classificação.

Relativamente à última classe, que remete para informações de tipologia *water*, o padrão de distribuição dos valores da medida de validação para as diferentes classificações, caracteriza-se por pontuações intermédias baixas para os dados de treino F1 e F2. De um modo geral, as diferentes combinações de *features* registaram valores de *f1-score* próximos, sem qualquer destaque para uma combinação de dados concreta. Julga-se que este cenário esteja inteiramente relacionado com a pequena dimensão da classe, uma vez que são poucos, os corpos de água que se podem encontrar na área de estudo.

6.1.4 Síntese de resultados

Sintetizando, dos resultados encontrados ao longo das discussões desenvolvidas nas secções anteriores, é possível determinar um conjunto de considerações aliadas à utilização conjunta de *features*, e como estas influenciam nas classificações de imagens. Uma vez que para o exercício foram utilizados dois conjuntos de dados de treino (F1 e F2), numa primeira fase foi exequível realizar uma comparação entre estas informações, relativamente à distribuição geográfica das classes nas diferentes classificações. A partir dos mapas produzidos, entendeu-se que o maior nível de detalhe da representatividade das classes de ocupação do solo recaiu sobre os dados de treino F1, enquanto algumas inconsistências foram associadas às informações produzidas através dos dados de treino F2. Dado que o rigor da distribuição de classes informativas com base nos dados combinados, foi apenas reconhecida para as classificações auxiliada dos dados de treino F2, atendendo para um primeiro momento, em que o desempenho das classificações melhorava com um número superior de *features* combinadas.

A análise exploratória foi ampliada, e para esta segunda fase do exercício, os mapas produzidos passaram por duas estratégias de validação, avaliados por uma amostra de pontos aleatórios e pela comparação entre mapas. Medidas de validação, como exatidão temática e *f1-score* foram calculados, e assim se procedeu à análise de resultados. A primeira impressão conduzida pelas exatidões dos mapas, remeteu para as grandes variações da medida entre os dados de treino, conferindo exatidões superiores para o dado de treino F1. Para além disso, quando se comparavam as percentagens de exatidão das classificações, individualmente pelo determinado treino, as percentagens da exatidão temática remetiam para valores extremamente próximos para as diferentes classificações. Uma das razões apontadas, em literatura científica, para os comportamentos erráticos destes parâmetros de análise, direciona-se para a possível influencia da igualdade estabelecida na resolução espacial das bandas das diferentes imagens e MDE, consequência das etapas de pré-processamento aplicadas aos diferentes dados.

Contudo, a baixa variação entre a exatidão temática permitiu, da mesma forma, atingir o objetivo da análise. De modo geral, para as classificações discutidas, manifestou-se uma relação direta entre o número de *features* combinadas e os resultados. Conclui-se que o desempenho de combinações de bandas de apenas um satélite ou das principais bandas de dois satélites, conduziam a classificações de qualidades inferiores, assegurados pelos baixos valores registados tanto na exatidão temática, como na medida de *f1-score*. Todavia, com o

desenvolvimento da discussão de resultados, apercebeu-se de que classificações que logravam do maior número possível de bandas dos dois satélites ou destas informações em combinação com o MDE, promoveram para avaliações mais positivas. Neste sentido, as combinações de *features* de classi_D4 e classi_D8 afirmaram-se com as classificações de exatidão temática mais detalhadas, bem como um excelente desempenho na medida de *f1-score* para a grande maioria das classes de ocupação do solo. Estes promoveram uma exata individualização das informações, bem como acabaram por contornar generalizações das restantes classificações. Visto que as referidas combinações tiram maior proveito de um amplo conjunto de informações, estas destacaram-se pela sua preeminência e capacidade de produzir mapas de ocupação do solo, bem como na correta representação de classes temáticas com características espetrais mais complexas.

6.2 Resultados: Classificação dos grupos de dados de treino F0, F1, F2, F3 e F4

Relativamente à segunda parte de análise de resultados da classificação de mapas de ocupação do solo, perante a utilização concreta de diferentes dados de treino proporcionados das várias regras de filtragem introduzidas, serão estudados os produtos de mapas mais satisfatórios, mediante da utilização dos diferentes conjuntos de dados de treino. É importante salientar que, a utilização combinada das bandas Sentinel-2, Landsat-8 e informações relativas ao MDE das classificações executadas no primeiro exercício, vão ao encontro dos dados combinados selecionados para este segundo momento que diz respeito ao desempenho individual dos dados de treino F0, F1, F2, F3 e F4. Dadas as conclusões retiradas na secção anterior, classi_D4 e classi_D8 são os conjuntos de dados de imagens definidos, uma vez que mostraram ser superiores às demais classificações. Dada à complexidade de análise dos resultados do presente ponto, comparativamente aos primeiros resultados discutidos, de forma estabelecer uma estruturação clara dos conhecimentos que abordados, a presente secção divide-se em:

6.2.1 Cobertura dos dados de treino e representatividade das classes;
6.2.2 Comparação do treino produzido com a referência;
6.2.3 Resultados obtidos: explicação da distribuição geográfica das classes nas diferentes classificações;
6.2.4 Resultados da validação com amostra aleatória;
6.2.5 Resultados da validação com comparação entre mapas;
6.2.6 Síntese de resultados.

6.2.1 Cobertura dos dados de treino e representatividade das classes

Anteriormente, foram apresentadas as características e determinantes que definiram os grupos de dados de treino. Nesta sequência de resultados e discussão, a análise que se inicia aborda a cobertura total dos dados de treino, desde o descarregamento das informações OSM na sua constituição original, até à sua estrutura e transformação final após a implementação das diferentes regras de filtragem produzidas. Deve-se relembrar, de forma sucinta que os cinco *datasets* de treino utilizados, distinguem-se por F0 correspondente aos dados originais descarregados do OSM e posterior conversão para classes LULC, auxiliada da execução do *software* OSM2LULC. Os dados F1 que surgem da exclusão das células mistas

e das regras de filtragem introduzidas no artigo de referência Fonte, *et al.* (2020) e F2 dos mesmos índices radiométricos utilizados nas regras do estudo, mas da introdução das novas regras investigadas ao longo do presente trabalho. O treino F3 desenvolve-se das novas regras procuradas e da introdução de novos índices radiométricos e F4 de regras de uma perspetiva de trabalho investigativo proporcionado por outro autor no momento de desenvolvimento da presente investigação.

Numa lógica de discussão dos produtos dos dados de treino, observa-se a área total coberta pela representação das áreas do *dataset* F0. A Figura 23 expõe os elementos de cobertura da informação associada ao respetivo treino, com inventariação para todas as classes de ocupação do solo da nomenclatura definida. Tendo em mente que são considerados os dados brutos recolhidos do OSM, é comum verificar que as informações delimitam toda a extensão da área de estudo, concedendo a sua forma original, com uma cobertura de informação de quase todas as áreas internas.

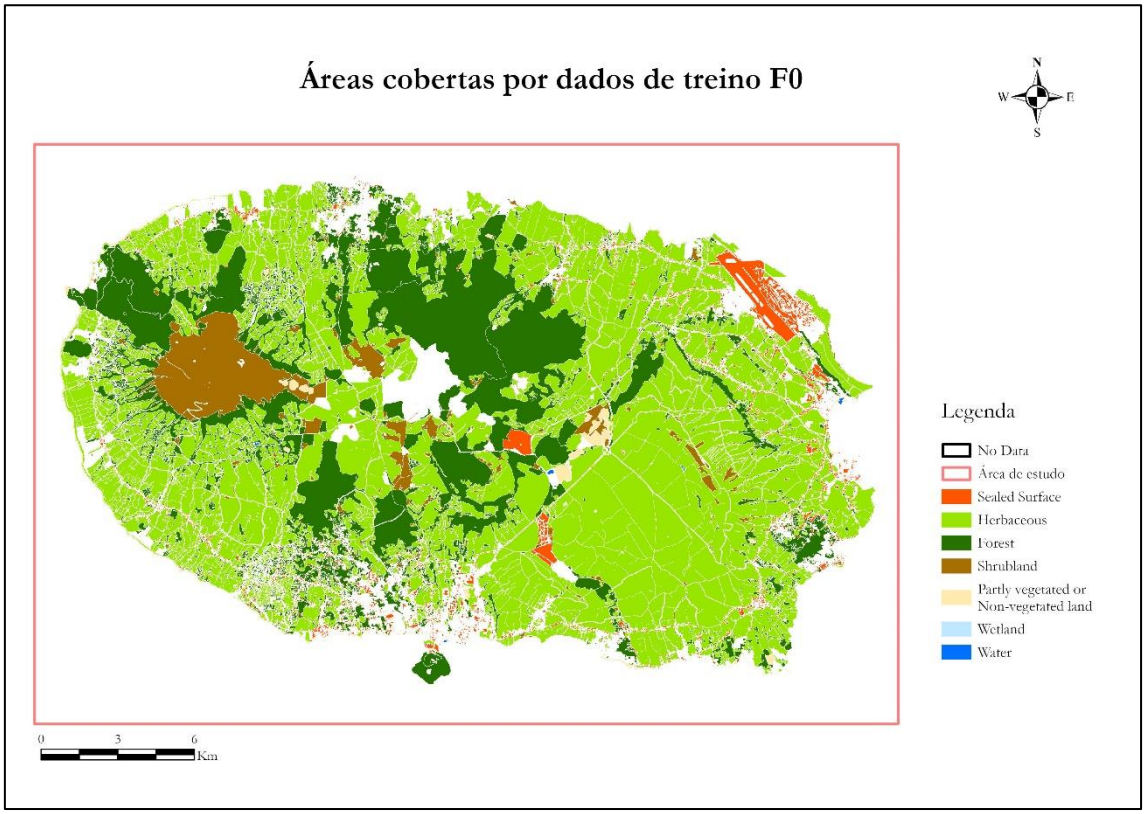


Figura 23 – Mapa de áreas cobertas pelos dados de treino F0.

A partir da figura observa-se, que algumas regiões interiores, nomeadamente de proximidade a grandes centros urbanos, detêm de insuficiência de dados, no entanto, prevê-se que o respetivo conjunto concentre maior complexidade de informação. A partir deste conjunto de dados de treino, as demais regras de filtragem serão aplicadas, originando os diferentes *datasets*. O estudo dos presentes resultados centra-se nestas etapas, uma vez que a grande maioria da informação coberta pelos dados F0 apresenta inconsistências de determinadas classes temáticas. Por isso, é essencial aplicar processos e formas de filtrar os dados de maneira a obter apenas as informações corretas e tentar melhorar resultados.

Na sequência da distinção da área total coberta pelos diferentes dados de treino, as informações F1, resultado da aplicação de processos de filtragem que incluem a exclusão dos pixéis mistos dos dados OSM e a aplicação das regras introduzidas por Fonte, *et al.* (2020), a partir dos determinados índices radiométricos, dão início ao primeiro processo de seleção. Este processo, tal como os seguintes, visa a seleção das informações unicamente corretas e eliminação de inconsistências, causando a estratificação dos dados e consequente diminuição da representatividade. Com um olhar atento para a análise da Figura 24, cujo objetivo se foca na apresentação da cobertura das informações OSM após a aplicação dos métodos de filtragem descritos, é evidente dar conta de um decréscimo percentual da informação do conjunto dos dados de treino.

É interessante notar que, embora a diminuição não seja muito significativa, este primeiro processo de filtragem dos dados de treino eliminou uma parte considerável de polígonos relativos à informação urbana. Isto acontece porque muitas das regiões deste caráter deixaram de ser consideradas, com apenas uma permanência evidente, mas também com uma redução percentual do aeroporto. De acordo com informações particulares das classes de vegetação, *hebaceous*, *forest* e *shrubland*, o decréscimo percentual é também reconhecido, com uma forte evidência para a classe relativa às herbáceas. Este fator advém da aplicação das regras de filtragem, através do Programa 19, visto que utiliza uma banda de referência do Sentinel-2 após a aplicação de correções de pré-processamento, proporcionando a exclusão de informações nos dados de treino onde os efeitos atmosféricos coincidissem.

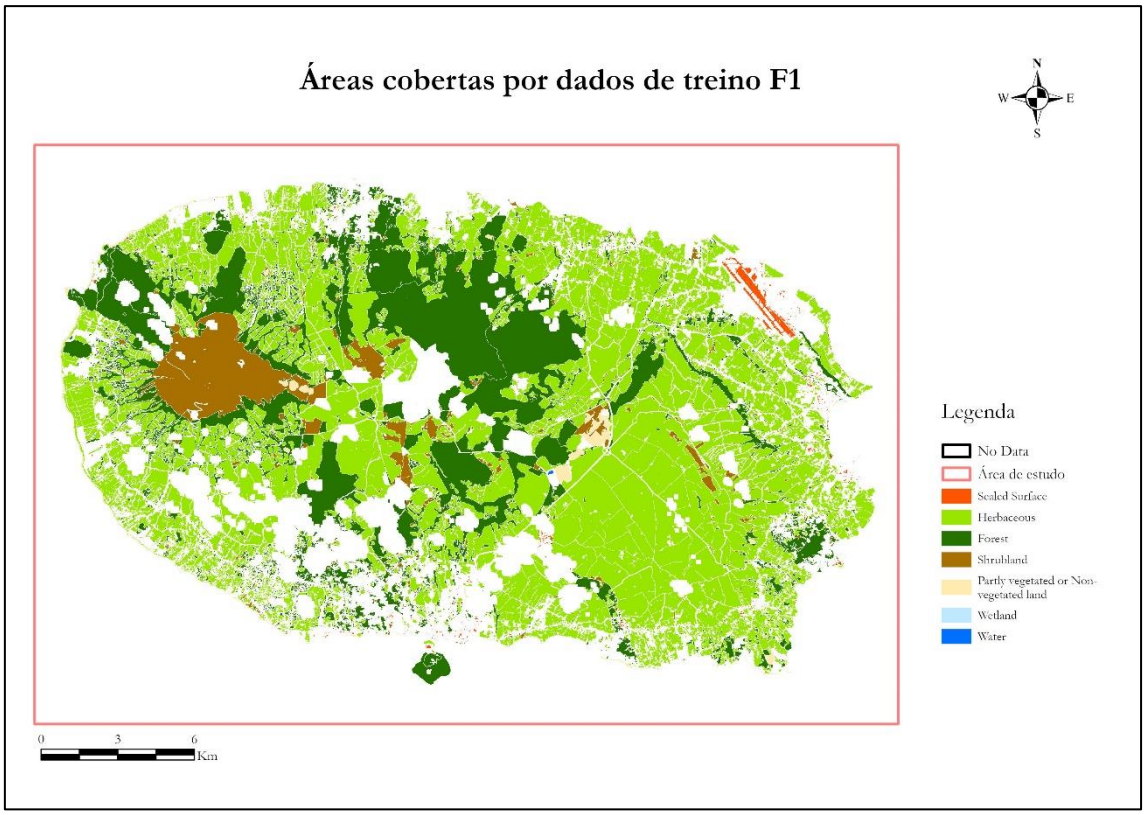


Figura 24 – Mapa de áreas cobertas pelos dados de treino F1.

Para o *dataset* F1, é possível reconhecer que as condições de filtragem conservaram um conjunto de informação eminente, podendo ser encarado como um modelo de regras abrangente e não discriminatório. Esta conjuntura faz entender que alguma informação incorreta permanece nos dados de treino F1, demonstrando que é necessário alterar e aprofundar o conhecimento sobre regras de filtragem.

Neste sentido, a metodologia implementada, teve em consideração, a criação de regras discriminatórias da informação imprecisa evidente nos dados de treino. No âmbito deste trabalho, os dados de treino F2 são inicialmente considerados. Este *dataset* incorpora, como anteriormente referido, os índices radiométricos aplicados em F1, mas com novas condições de filtros. Esta abordagem, levou ao resultado da Figura 25 que discrimina as amostras de treino para F2, onde é sentida uma forte redução de informação relativa às diferentes classes temáticas. De um modo geral, as classes de extensões superiores, nomeadamente *forest*, *herbaceous*, *shrubland* e *sealded surface* padeceram de uma redução significativa, sem considerar as áreas de incidência dos efeitos atmosféricos, eliminados nas correções de imagens. Este cenário é também sentido para as informações relativas à classe *partly vegetated or non-vegetated.*

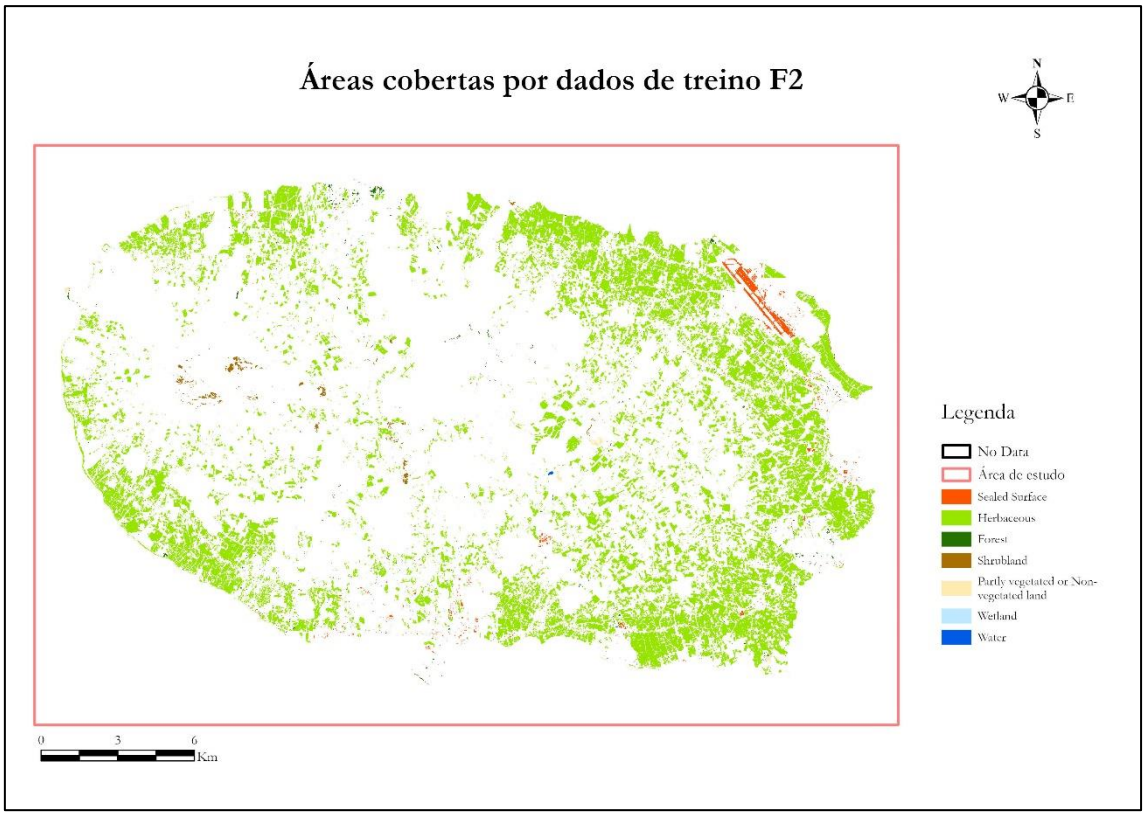


Figura 25 – Mapa de áreas cobertas pelos dados de treino F2.

O treino F2 apresenta uma filtragem mais rigorosa, comparativamente aos dados F1, percebendo que a informação das superfícies artificiais e de áreas herbáceas mantém uma representatividade percentual, contudo as classes de florestas e de arbustos detêm de uma diminuição significativa, proporcionando a sua difícil distinção na figura. Os resultados dos respetivos dados revelam que as regras de filtragem realizam uma seleção informativa dos dados de treino mais exigente, acabando por considerar muita informação como imprecisa.

Na mesma lógica, as informações de F3 revelam uma discriminação informativa que acompanha as imprecisões encontradas nos conjuntos de treino. Para além de introduzir as condições de filtragem de F2, destaca-se com o cálculo de novos índices radiométricos com regras associadas. De acordo com a Figura 26, num primeiro momento, pode assimilar a percentagem de cobertura dos dados como idêntica ao *dataset* F2. As regras de filtragem implementadas seguiram as mesmas condições, para a supressão rigorosa da informação ligada à ocupação por florestas e arbustos, bem como de espaços urbanos e solo descoberto. Embora as semelhanças entre os dados de treino sejam evidentes, as percentagens de área coberta das classes mais marcantes sofreram, do mesmo modo, de um decrescimento da sua representatividade.

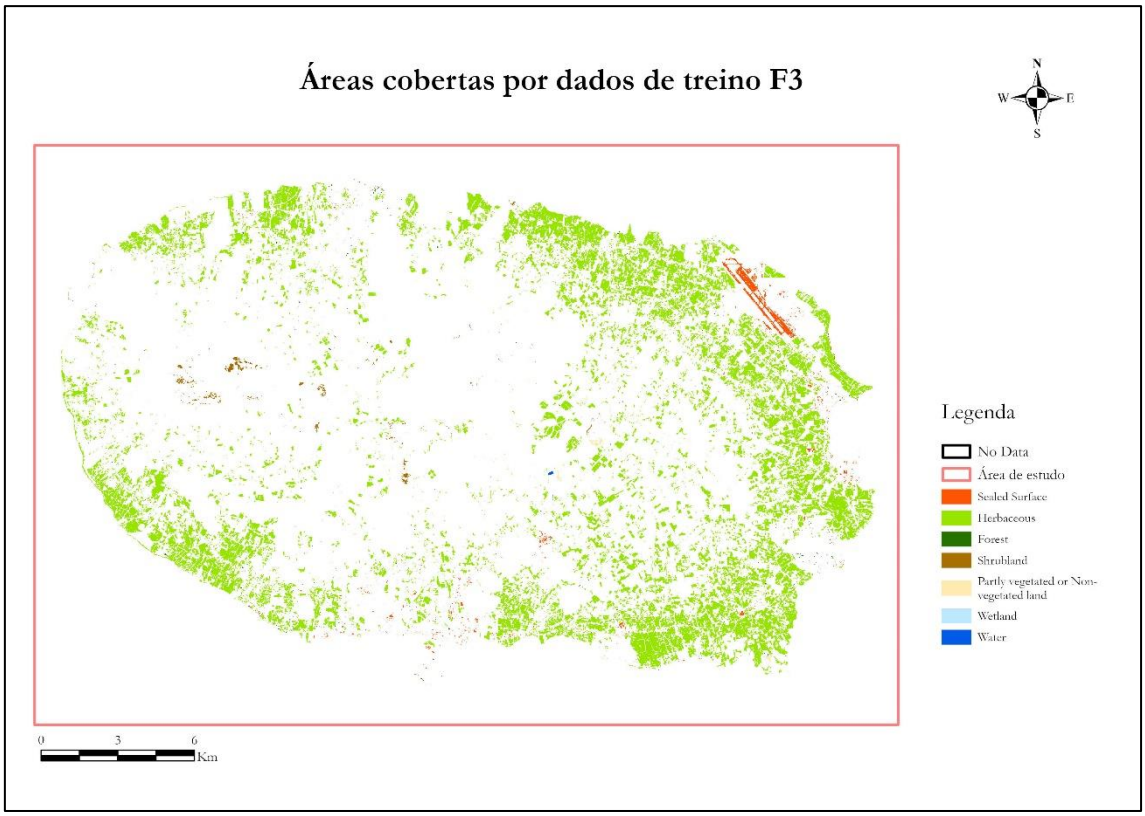


Figura 26 – Mapa de áreas cobertas pelos dados de treino F3.

Deste modo, as diferenças entre os treinos F2 e F3 resultam, da introdução de novos índices, comprovando um peso associado, mas no fundo para as duas condições de filtragem percebe-se que a metodologia de identificação de pixéis puros para definição de novas regras, permite uma distinção e discriminação de classes de pormenor superior. Vale a pena mencionar que a estratégia metodológica escolhida, para a definição de novas regras, proporcionou uma filtragem superior aos dados F1, no entanto, é importante interpretar os resultados obtidos na primeira discussão.

Finalmente, menciona-se o último conjunto de dados de treino, em que se apura a área coberta por dados originários de um estudo realizado em simultâneo com o desenvolvimento da presente investigação. O *dataset* F4 é fruto da investigação proporcionada por um segundo autor, com a metodologia, também, direcionada para identificação de células puras por classe, porém para uma área geográfica mais ampla e representativa. O processo de filtragem do treino envolve os mesmos índices radiométricos do estudo de Fonte, *et al.* (2020) e outros que o autor achou por bem empregar. Ademais, as regras estabelecidas pelo estudo acabam por se distinguir das definidas para os dados F2 e F3, fundamentando uma análise e discussão de resultados mais complexa e vigorosa.

A Figura 27 alude para a área coberta pelos dados treino F4 que respeita as condições estabelecidas pelas regras de filtragem implementadas por esta segunda pessoa, entre as quais predominam as informações relativas à ocupação por florestas. Por sua vez, verifica-se uma grande extração de informação dos dados, estabelecida pela redução de percentagem de cobertura de classes temáticas relacionadas às herbáceas, espaços urbanos, solo descoberto e arbustos.

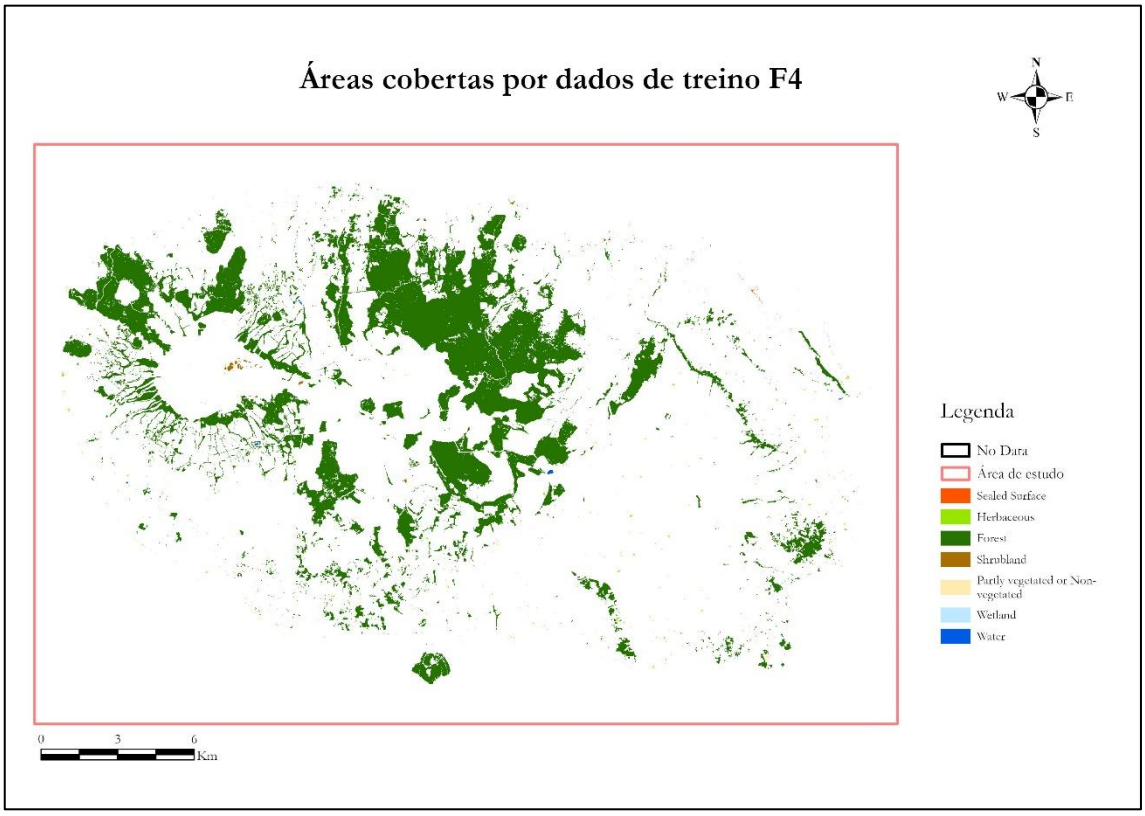


Figura 27 – Mapa de áreas cobertas pelos dados de treino F4.

Segundo a análise da figura, pode assumir-se que a estratégia de filtragem estabelece pontos rigorosos, conferindo uma estratificação das demais classes para uma representatividade mais restrita, como se verificou com os *datasets* F2 e F3. Não deixa de ser curioso, que mediante as duas abordagens para uma metodologia idêntica, produzem-se resultados consideravelmente distintos. É importante reconhecer que para os treinos F2 e F3, a classe *herbaceous* foi a mais preservada pelos processos de filtragem, enquanto a informação quase inalterada foi *forest*. De certo modo, tirando partido do estudo introduzido, deve ser considerado que as informações estratificadas regem dados precisos e de confiança, dado que a cobertura de informação eliminada, diz respeito a polígonos imprecisos, proporcionando inconsistência para o treino do algoritmo e classificação de imagens.

6.2.2 Comparação do treino produzido com a referência

Com o intuito de demonstrar que regras de filtragem conduziram à melhoria dos conjuntos de dados de treino, e examinar a veracidade da informação selecionada, foi realizada uma análise relativa à exatidão temática entre os *datasets* resultantes dos processos de filtragem e o mapa de referência. Este ponto é apresentado como uma primeira análise na compreensão da utilidade da metodologia. Assim sendo, para o objetivo pretendido, a partir dos dados produzidos foi elaborada a Figura 28, com finalidade de apresentar os resultados das exatidões temáticas do treino em referência às informações CLC+ BB.

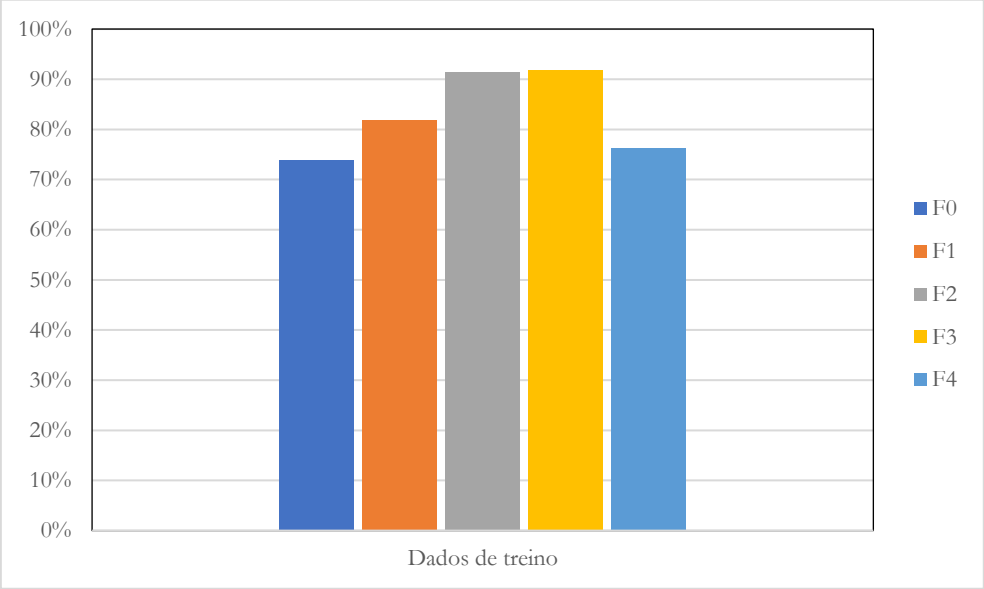


Figura 28 – Resultados de exatidão temática do treino dos dados de treino, com referência para o mapa CLC+ BB.

Os dados brutos do OSM (F0) registados e apresentados na figura, assinalam a medida mais baixa de 74%, para os dados filtrados a partir das regras do artigo Fonte, *et al.* (2020), verificando-se uma subida para 82% que se perpetua até aos treinos de F2 e F3 que assinalam 91% e 92% respetivamente. Em referência para os dados F4, o mesmo cenário não se verifica, dada à redução do valor de exatidão para 76%, situação que pode atender a um possível desajuste associado às regras de filtragem.

No sentido de aprimorar a análise da medida estatística dos padrões das classes, a discussão dos resultados alargou-se para a análise dos valores de *f1-score*. Para tal, a Figura 29 apresentada abaixo, demonstra de forma sintetizada as estatísticas de *f1-score* obtidas na fase de treino, individualmente para cada classe e o respetivo conjunto de dado de treino a que pertence.

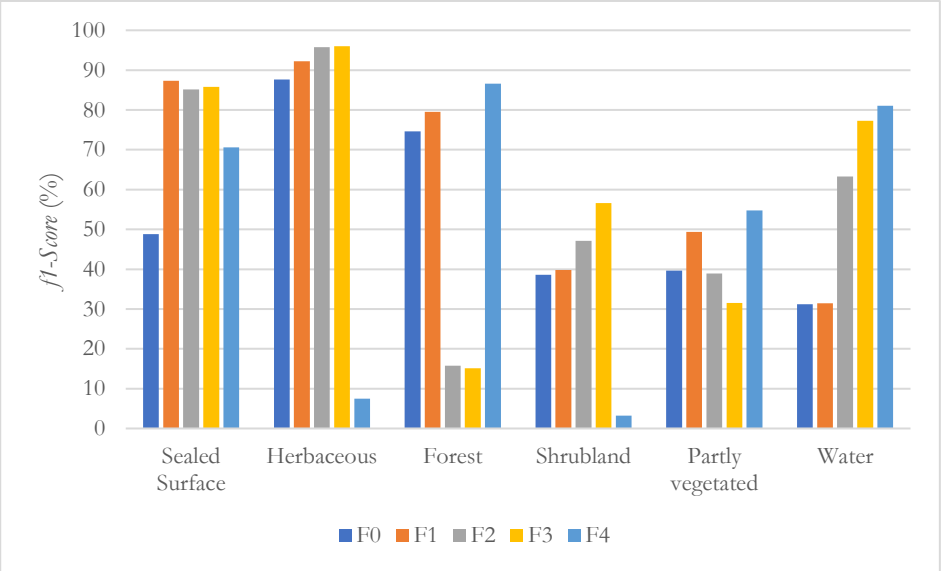


Figura 29 – Valor *f1-score* de cada classe a partir dos vários conjuntos de dados de treino com referência para o mapa CLC+BB.

Tirando partido do complemento estatístico da figura acima, entende-se que para as duas primeiras classes de ocupação do solo, *sealed surface* e *herbaceous,* assinala-se uma superioridade da medida *f1-score* para o treino dos *datasets* produzidos a partir dos diferentes processos de filtragem em comparação com os dados F0, porém para a classe de herbáceas, o treino F4 registou a estatística mais baixa. Nesta fase do treino, os padrões da medida favorecem as informações dos dados filtrados, com referência para os espaços artificializados, onde se destacam as amostras de F1, como o valor mais alto, de 87%, seguido de F3 e F2 com percentagens muito próximas, de 86% e 85% respetivamente.

Nas informações de *herbaceous*, é assinalada uma subida em cadeia lógica de F0, para F1, de F1 para F2 e F3, cenário que revela a soberania das novas regras introduzidas. Relativamente às percentagens, F0 regista 87%, seguido de F1 com 92% e F2 e F3 assinalam o valor de 96% para esta classe *herbaceous*. Para o *dataset* F4, a situação foi muito diferente, uma vez que deteve de um *f1-score* de apenas 7%. Este contexto, foi igualmente verificado na classe de ocupação do solo *shrubland*, com representação da mesma sequência lógica na medida de *f1-score* e o valor mais elevado a corresponder ao treino F3, com 57%. A classe *forest* aponta para um quadro do treino com percentagens elevadas dos dados F0 com 75% e F1 de 80%, no entanto a filtragem providenciada por F4 aparenta uma melhoria comparativa para o valor de 87%, enquanto as amostras F2 e F3 sublinham um *f1-score* baixo, de apenas 15%.

No que diz respeito, à classe de *partly vegetated or non-vegetated*, os valores de *f1-score* para os *datasets* F0, F2 e F3 são reconhecidos com percentagens idênticas e as mais baixas registadas

para a respetiva classe. Os valores destes três dados de treino encontram-se próximos de 40%, no entanto é pertinente destacar o treino para as amostras de F4, identificadas como o valor mais alto de 55%, seguido das informações de F1. Para a última informação de ocupação do solo, a classe *water* acarreta uma distribuição dos valores de *f1-score* nos dados de treino que indicam o aumento significativo dos valores de *f1-score*, dos dados de treino F2, F3 e F4 com uma sequência lógica crescente. Ainda nesta classe, os treinos de F0 e F1 com valores de 31%, assinalam a medida mais baixa.

Dado ao contributo dos gráficos, verificou-se que os valores mais altos de exatidão temática registados vão ao encontro dos dados de treino F2 e F3, em comparação com o mapa de referência. Além disso, de um modo geral, o registo das medidas *f1-score*, dos dados produzidos a partir das novas regras de filtragem revelaram-se superiores, em relação aos dados F1 e F0. Ao longo desta análise sustentada nos princípios mencionados, percebeu-se que a seleção rigorosa e a eliminação de informações imprecisas nos dados de treino, a partir de regras de filtragem exigentes, conduze a uma amostra por classe mais reduzida. Contudo essas informações detêm de um grau de assertividade e de certeza elevada para as informações encontradas nos dados de treino. Esta visão reforça a ideia de que as iniciativas para metodologias de filtragem de informações OSM, como foi o caso de estudo do artigo de Fonte, *et al.* (2020) e a metodologia desenvolvida ao longo da presente dissertação, se enquadram num grau de extrema relevância.

6.2.3 Resultados obtidos: explicação da distribuição geográfica das classes nas diferentes classificações

A presente secção refere a interpretação de sensibilidade visual, das figuras posteriormente apresentadas (Figuras 30 e 31), correspondentes aos *outputs* finais dos mapas classificados a partir dos diferentes dados de treino para as combinações de *features* classi_D4 e classi_D8, respetivamente. Através de uma observação crítica das imagens é possível detetar variações da distribuição de classes temáticas, registadas em detrimento dos *datasets* introduzidos. A partir de uma visão comparativa, percebe-se que os mapas dos dois conjuntos de *features* combinados, detêm pequenas diferenças na distribuição da ocupação do solo entre os mesmos dados de treino introduzidos.

A comparação dos mapas de ocupação do solo produzidos pelos diferentes dados de treino, em relação ao mapa de referência permite destacar algumas diferenças. Numa primeira

análise visual, verifica-se uma forte semelhança entre o mapa de referência e a distribuição geográfica das classes das classificações com dados de treino F0 e F1. No que diz respeito aos mapas gerados através das informações dos dados F2, F3 e F4, a representação da superfície difere da realidade, quando as informações são comparadas com o mapa CLC+ BB. Sem dúvida que este é primeiro ponto assente na análise dos resultados das classificações obtidas, sendo contudo importante, ir ao fundo da questão no que toca à distribuição das classes de ocupação do solo várias classificações.

A qualidade da classificação dos mapas de ocupação do solo, tendo por base a utilização individual dos dados F0, leva a uma apreciação de pequenos fenómenos de erros de distribuição geográfica para as classes temáticas de *sealed surface*, *shurbland* e *forest*. Para as categorias identificadas verifica-se uma certa inexatidão na classificação, uma vez que se assinala um exagero das áreas ocupadas por superfície artificial, como por exemplo da área do aeroporto, bem como de todas as regiões de aglomerados urbanos, onde são assinaladas áreas de ocupação superiores à de referência.

A determinação de áreas ocupadas por matos originou, igualmente, um certo erro associado a confusão com a classe de florestas, tendo em vista que em duas áreas particulares no interior da área de estudo, estas são facilmente identificadas. Verifica-se que existe uma certa limitação na distinção entre as duas classes, percebendo que nestas áreas específicas o algoritmo considerou tudo como floresta ou tudo como arbustos, acabando por comprometer o detalhe informativo da distribuição e distinção das duas ocupações do solo.

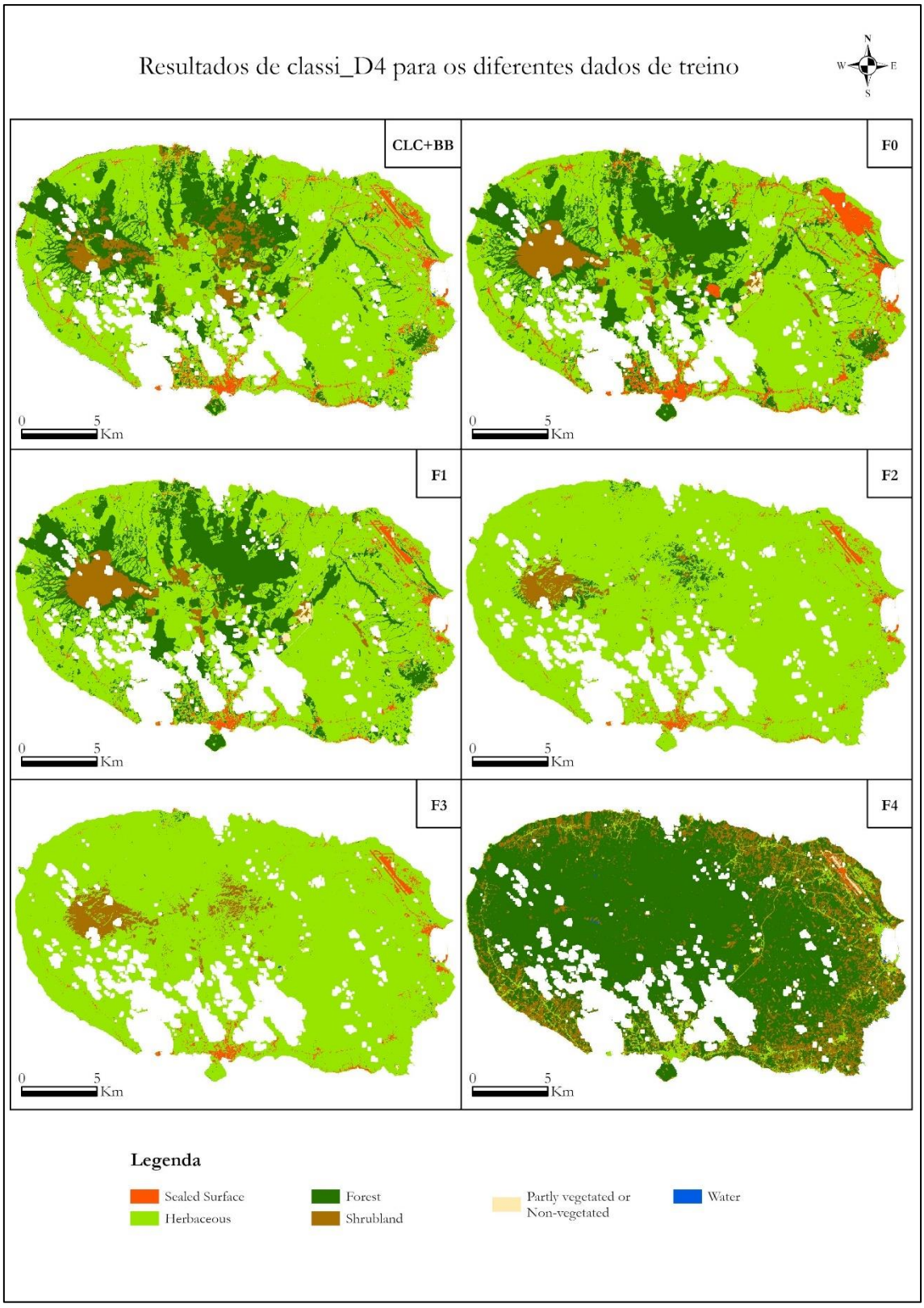


Figura 30 – Resultados das classificações de classi_D4 para os diferentes dados de treino.

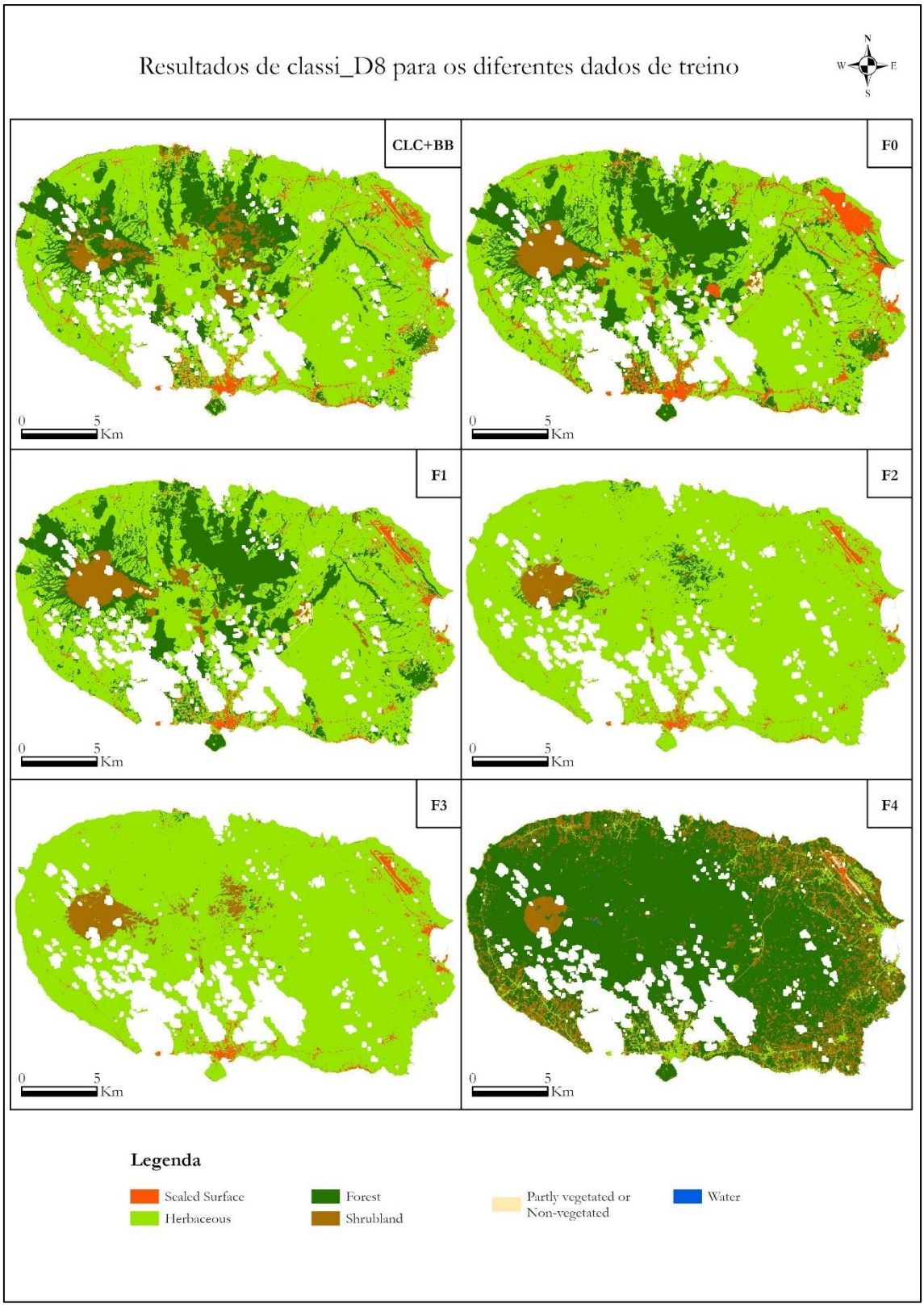


Figura 31 – Resultados das classificações de classi_D8 para os diferentes dados de treino.

No fenómeno de classificação através do *dataset* F1, assinala-se uma melhoria relativa à exatidão temática dos espaços artificializados, com uma correta identificação das áreas urbanas, porém com uma perda de detalhe associado às vias de comunicação. As confusões entre as regiões ocupadas por florestas e matos prevalece, como erros únicos visualmente assinalados na classificação, tal como se particularizou com os dados de treino F0. De forma a apresentar as alterações de ocupação do solo sofridas entre as classificações produzidas a partir das informações F0 e F1, e o mapa de referência, a Figura 32 representada abaixo, alude para as respetivas variações. Para a classificação produzida através de F0, a área total sofreu uma alteração de 15.7%, e no caso da classificação de F1, foi de apenas 14.8%.

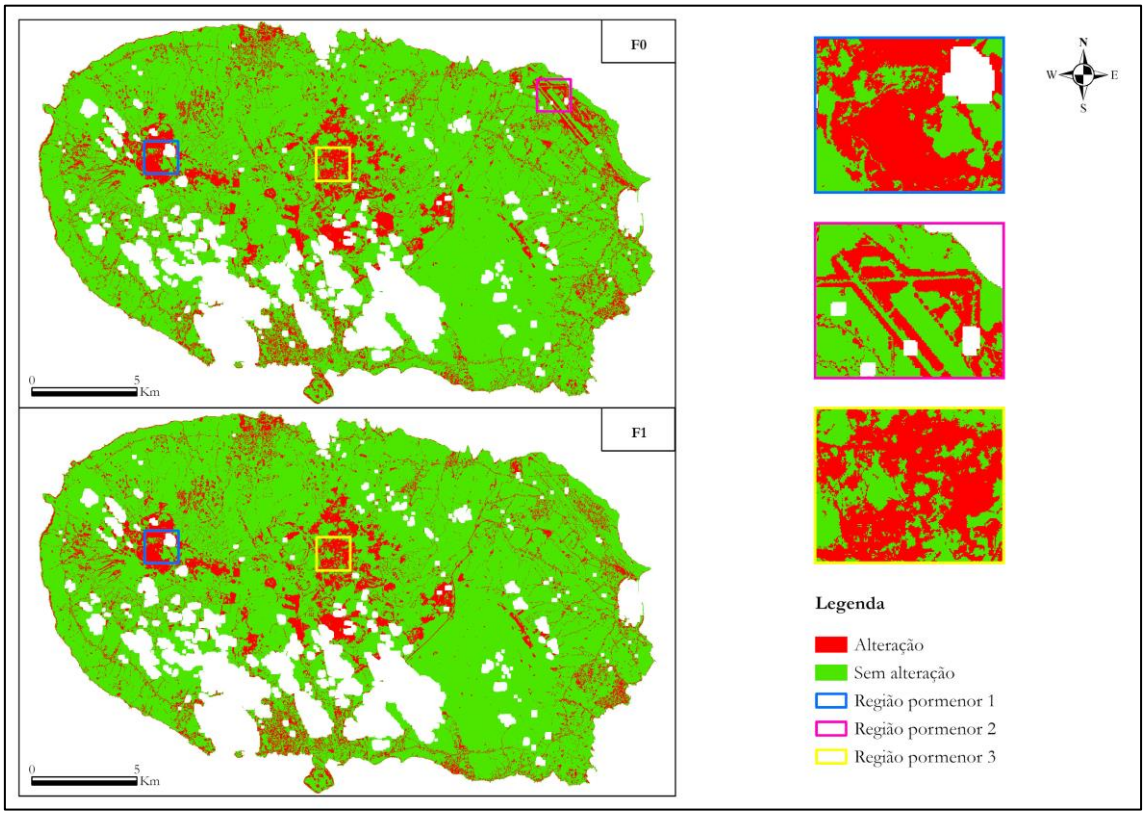


Figura 32 – Evolução da ocupação do solo entre classificações de dados F0 e F1 e o mapa de referência. Região pormenor 1 e 3: confusões entre classes *forest* e *shrubland*. Região pormenor 2: área particular do aeroporto.

Os padrões de qualidade dos mapas de ocupação do solo, produzidos por intermédio dos dados de treino F2, F3 e F4, em ótica individual apresentam fenómenos de erros de classificação mais assentes do que os analisados previamente. Com base nos dados F2 e F3, salienta-se uma distribuição exponencial de áreas ocupadas pela classe *herbaceous*, considerando a informação deste domínio como prevalente em quase toda a área de estudo. O fato dos respetivos mapas resultarem numa ocupação de vegetação do tipo herbáceas

ampla, demonstra que existe um problema relacionado com a restante informação, evidenciando a distribuição geográfica das classes *forest*, *shrubland* e *partly vegetated or non-vegetated*.

Em comparação com a referência, percebe-se que as classes referidas perderam alguma representatividade, notando uma redução da percentagem ocupada. Assume-se a possível justificação, como referente à perda de heterogeneidade e representatividade informativa destas classes nos *datasets* F2 e F3, mediante das novas regras de filtragem, como verificado na secção da cobertura dos dados de treino. Embora aparentem inconsistências na distribuição das informações aliadas à vegetação, assinala-se ainda, uma boa classificação de espaços artificializados, com uma correta individualização do aeroporto e dos maiores centros urbanos. Todavia, o nível de detalhe para a classe de espaços artificializados, foi comparativamente inferior ao de referência.

No que diz respeito à distribuição geográfica das classes por parte dos dados de treino F4, os resultados mostram que grande parte da extensão geográfica foi considerada como florestas, situação particular e distinta das análises feitas às classificações anteriores. É possível assinalar este cenário como um claro erro de classificação, dado que as ocupações predominantes são herbáceas, no entanto é importante ter em atenção para outros aspetos ligados às classes temáticas de *shrubland*, *herbaceous* e *sealed surface*.

Observam-se confusões evidentes entre a distribuição da ocupação de herbáceas e de espaços artificializados, com classificações de vias de comunicação e de espaços artificializados como vegetação e a identificação de matos em locais de presença de herbáceas. Além disso, é também evidente um desajuste na área do aeroporto, onde uma parte desta informação do tipo *sealed surface*, acabou por ser classificada como solo descoberto. Esta situação em específico, pode estar relacionada com as regras de filtragem dos dados de treino F4, que contribuíram para a elevada mitigação de informações da maioria das classes, prevalecendo apenas dados relativos à classe de florestas. Estas informações fizeram com o que o algoritmo considerasse a maior parte da área ocupada como *forest*.

Em comparação com a referência, as diferenças são amplamente evidentes, contudo, é possível assinalar classes temáticas corretamente distribuídas pela área de estudo, nomeadamente para as informações relativas a *forest*, parte do aeroporto considerada como *sealed surface* e ainda uma região interior classificada como *shrubland*. Na figura seguinte são apresentadas as áreas que sofreram alterações de ocupação do solo. Para a classificação de F2 e F3, a área total alterada foi de 28.5% e para F4 75.9%.

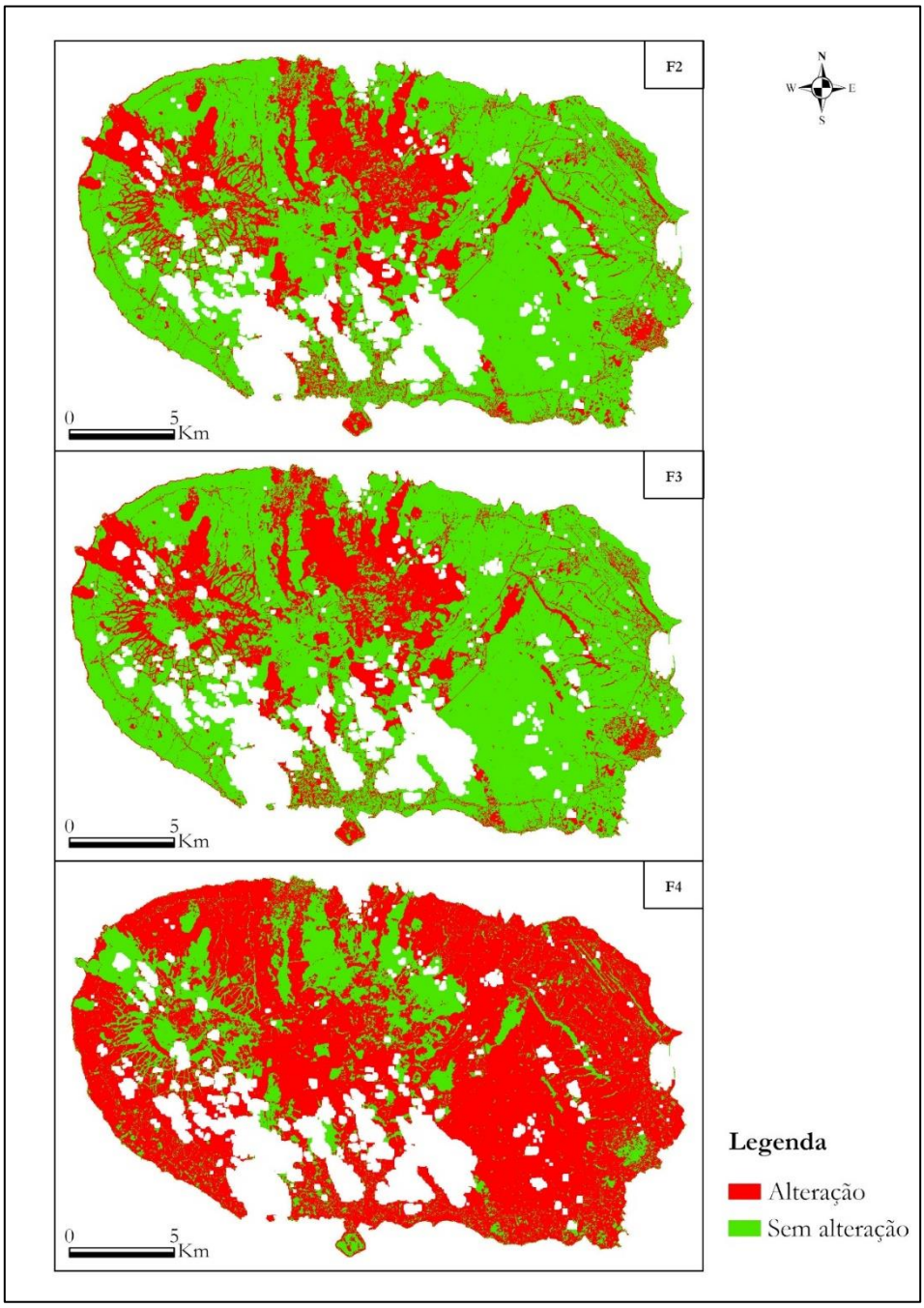


Figura 33 – Evolução da ocupação do solo entre classificações de dados F2, F3 e F4 e o mapa de referência.

6.2.4 Resultados da validação com amostra aleatória

No âmbito do desempenho eficiente das classificações de ocupação do solo a partir dos diferentes grupos de dados de treino, foram avaliadas matrizes de confusão por intermédio da primeira estratégia de validação, para alcançar informações sobre a exatidão temática, para os diferentes conjuntos de dados de treino e conduzir à discussão de resultados. Na Figura 34, sintetizam-se os valores de exatidão temática dos mapas de ocupação do solo, em função dos dados de treino introduzidos classificados com recurso ao algoritmo RF.

Os valores registados no respetivo gráfico, mas também no apresentado na seguinte secção são o produto da execução dos Programas 20 e 21, descritos no subcapítulo dedicado à implementação da metodologia. Todavia, é importante evidenciar que no decorrer da criação de mapas, foi realizado um teste de treino e classificação dos dados em ambiente ArcGis. Estes revelaram avaliações de exatidão distintas dos resultados obtidos a partir da automatização de processos. Através da experiência, percebeu-se que a parametrização de métodos influencia nos resultados, e na respetiva situação, os mapas produzidos através de ferramentas projetadas pelo *software* SIG desfavoreceu as exatidões temáticas dos mapas. Considera-se, a este nível, que a implementação de uma metodologia, com recurso a métodos programados, surge como uma abordagem com capacidades transcendentes para exercer as etapas processuais de treino e classificação de imagens. Neste sentido, tomou-se como princípio utilizar as melhores avaliações e, portanto, as informações evidenciadas nos gráficos desta secção e da seguinte, aludem para os resultados obtidos de processos programados.

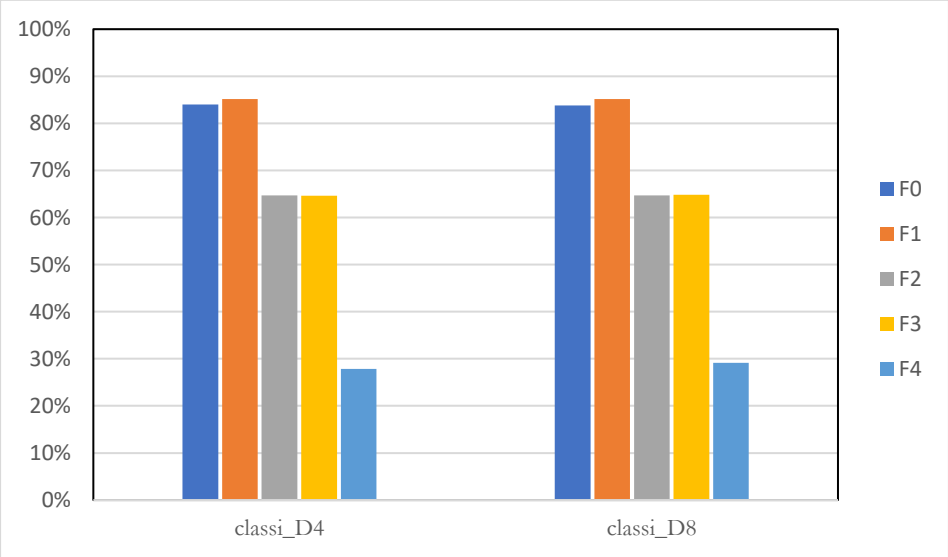


Figura 34 – Síntese dos resultados de exatidão temática obtidos para as classificações pela validação de amostra de pontos aleatórios.

Avançando para a discussão de resultados, segundo as informações da figura, verifica-se um destaque da percentagem de exatidão temática para os dados de treino F1, independentemente do conjunto de *features* combinadas. O valor de exatidão registado é de 85% e considera-se como um valor ótimo, conferindo um certo grau de certeza às informações de classes representadas por estes mapas de ocupação do solo. Quanto aos mapas produzidos pelos dados originais do OSM (F0), estes estabeleceram a segunda medida mais alta das classificações, com uma percentagem de 84%, com uma pequena diferença para os dados de treino F1.

A exatidão temática registada pelos restantes dados de treino, sendo estes produto das regras de filtragem introduzidas, assinalam uma sequência decrescente de F2 para F3 e de F3 para F4. Os *datasets* F2 e F3 são avaliados com um desempenho de classificação semelhante, de 65% e a diferença significativa para os dados de treino F4, alude para um valor de exatidão temática de apenas 28% na combinação de classi_D4 e 29% em classi_D8. Tendo em consideração a distribuição geográfica das classes nas diferentes classificações, estes valores são esperados e espelham a realidade nas classificações obtidas

Contudo, devido aos desajustes das classificações geradas pelos diferentes dados de treino, ao nível da distribuição geográfica das diferentes classes temáticas, procedeu-se à análise estatística do cálculo da medida de *f1-score*. Esta permita manifestar de forma clara as propriedades das diferentes classes nos mapas classificados, e perceber a sua correta distribuição no espaço geográfico. Face ao exposto, as Figuras 35 e 36 aludem, para a representação gráfica dos padrões de *f1-score*, para as determinadas classes informativas, relativamente aos dados de treino introduzidos, na fase de classificação. No Anexo E1 estão organizadas as diferentes matrizes de confusão calculadas, que para além de englobar a informação sintetizada abaixo e as percentagens de exatidão temática, são também encontradas informações relativas a outras medidas de validação de mapas, de modo a completar a sequente discussão de resultados.

Entre os diferentes conjuntos de dados de treino, utilizados como entrada nas classificações de imagens, os valores de *f1-score* apontam para a superioridade dos dados de treino F0 e F1 na grande maioria das classes temáticas, comparados às restantes amostras de treino. Devido à ampla representatividade da classe temática *herbaceous*, a percentagem da medida de *f1-score*, regista os valores mais altos, com destaque para os *datasets* F0 e F1, que determinam resultados acima de 90%, e sublinham hegemonia para as informações de ocupação do solo relativas a *sealed surface* e *forest,* para a respetiva estratégia de validação com amostras de pontos

aleatórios. Além disso, as diferenças de percentagem que separam os respetivos conjuntos de amostras de treino F0 e F1 para a classe de floresta, é insignificante, pois rondam percentagens muito próximas de 81%. A única marca de diferença encontra-se na classe de espaços artificializados, onde os dados de F1 são comparativamente superiores aos dados F0, com uma diferença de 54% para 63%.

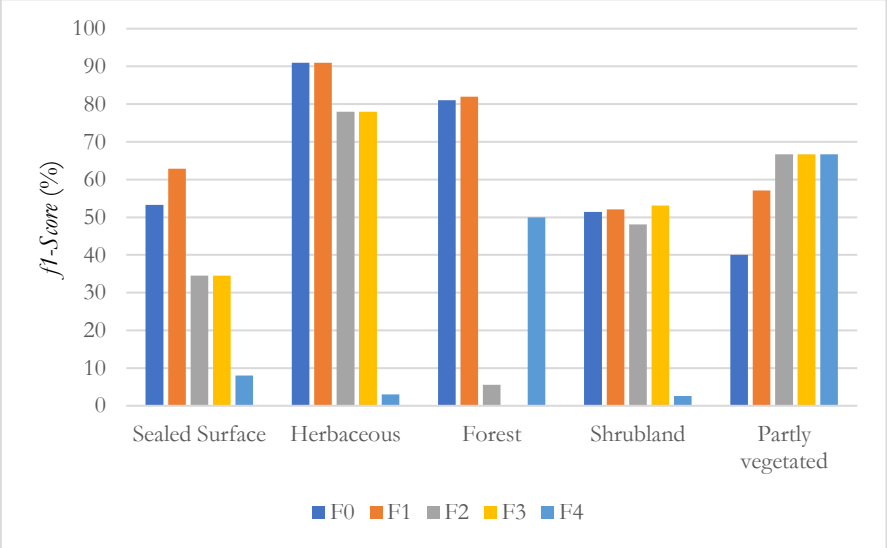


Figura 35 – Valor *f1-score* de cada classe classi_D4. Estratégia de validação: Amostra de pontos aleatórios.

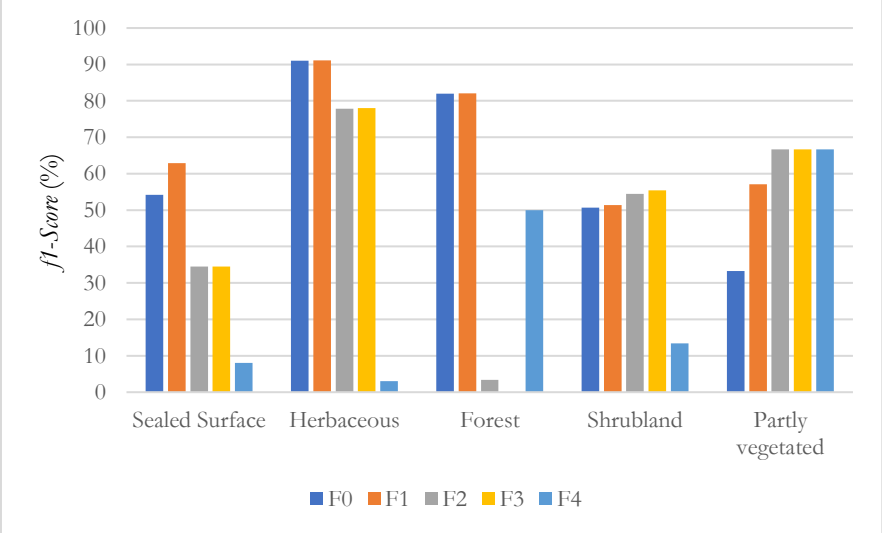


Figura 36 – Valor *f1-score* de cada classe classi_D8. Estratégia de validação: Amostra de pontos aleatórios.

O comportamento dos valores de *f1-score* dos dados de treino F2, F3 e F4 para as mesmas classes de ocupação do solo discutidas previamente, demonstram um nível da medida inferiores. Nos casos específicos dos *datasets* F2 e F3 para a informação referente a herbáceas apontam para resultados também altos e semelhantes de 78%, ponto que reforça a justificação aliada à ampla cobertura da área de estudo pela classe, mas também pela correta classificação e individualização da informação por parte dos respetivos dados de treino. Para as amostras F4, os resultados de *f1-score* da classe *herbaceous* remete para percentagens pobres de 3%, levando a possíveis erros associados à baixa representatividade de informações no conjunto dos dados de treino, determinada pelas regras de filtragens. Isto sugere que o respetivo dado de treino não têm capacidade de classificar corretamente a informação *herbaceous*, dado ao seu alto erro de omissão, próximo de 77% em classi_D4 e à sua confusão com a classe de espaços artificializados.

As informações da classe de *sealed surface* remetem para um quadro idêntico à distribuição dos resultados do critério *f1-score* da classe *herbaceous*, dado ao reconhecimento dos *datasets* F0 e F1 com os valores mais altos, seguido de uma descida para as amostras de treino F2 e F3 e de F2 e F3 para F4. Este cenário admite uma melhor capacidade dos treinos F0 e F1 para classificar espaços artificializados. O valor de *f1-score* para os dados de treino F2 e F3, remetem para 35%, no entanto o erro de omissão de F2 para esta classe foi muito alto, de 79%. Já o dado de treino F4, assumiu para esta classe um valor da medida *f1-score* muito baixo, uma vez que apresentou uma exatidão do produtor de 100%, e em descompensação uma exatidão do utilizador de apenas 4.2%.

Com um olhar voltado para a classe *forest*, nesta é também representada superioridade dos treinos F0 e F1 com valores da medida superiores a 80%. Para além disso, foi assinalado um realce da qualidade do *f1-score* para F4, comparativamente com as informações de F2 e F3 para a respetiva ocupação do solo. Os valores de *f1-score* mais baixos dos *datasets* F2 e F3, remetem para um erro de omissão de 98% nos dados de treino F2, aparentando uma insuficiência das informações deste treino para a classificação da classe, com um erro associado à classe *herbaceous*. No caso concreto de F3, este remete para uma situação única, tendo em vista que o *f1-score* foi de 0%. Este problema está inteiramente ligado à utilização da estratégia com uma amostra de pontos distribuídos de forma *random*, dado que nenhum ponto da amostra incidiu na classe *forest*.

Para além dos padrões das classes discriminados até ao momento, a análise das duas restantes classes, faz referência a valores de *f1-score* intermédios baixos para todos os *datasets*, comparativamente ao discutido inicialmente. Este fator parece estar relacionado com a dimensão das classes na área de estudo, assim como a complexidade espetral associada, que exige grande capacidade de individualização das suas características por parte dos diferentes dados de treino. Isto aponta para a aptidão de boas regras de filtragem determinarem uma representatividade significativa de amostras nos dados de treino e possibilitarem uma classificação exata.

Segundo a interpretação estatística de *f1-score* para a classe *shrubland*, os diferentes *datasets* apontaram para uma discriminação sensivelmente correta, com exceção dos dados de treino F4 que registaram um valor de apenas 13% na melhor classificação e um erro de omissão de 91%. Para os dados de treino F0, F1, F2 e F3 assistiu-se uma certa semelhança nos valores de *f1-score*, contudo foram as informações de F3 com o resultado mais alto de 55%, na melhor classificação. Este foi o primeiro exemplo das boas práticas das novas regras de filtragem, uma vez que o erro de omissão mais baixo, foi para este respetivo dado de treino.

Para as informações *partly vegetated or non-vegetated*, última classe de ocupação do solo, os comportamentos dos diferentes dados de treino determinam uma melhoria sequencial nos valores de *f1-score*, de F0 até F4. As percentagens mais baixas foram assinaladas pelos dados originais do OSM, registando um valor de 40% na melhor classificação. O treino F1 registou, para as duas classificações um *f1-score* de 57%, assim como para os dados F2, F3 e F4 as percentagens de *f1-score* foram de 67% para os três *datasets*, tanto na classificação classi_D4 como em classi_D8. A validação estatística para esta classe parece ser insuficiente, uma vez que os valores de *f1-score* foram idênticos para os dados de treino procedentes das novas regras de filtragem. A partir da leitura das diferentes matrizes de confusão percebeu-se que a amostra de pontos para esta classe foi demasiado reduzida, compreendendo numa amostra insuficiente para a validação. Neste sentido, espera-se uma maior veracidade e confiança nas percentagens de *f1-score* da validação com comparação entre mapas.

6.2.5 Resultados da validação com comparação entre mapas

Na segunda validação das classificações produzidas, foram avaliadas tabelas de contingência, por intermédio da estratégia aliada ao mapa de referência, de maneira a realizar a análise relativa às percentagens de exatidão temática providenciados pela introdução dos diferentes dados de treino. A Figura 37, abaixo apresentada, sintetiza esta realidade, permitindo uma visão sobre alguns resultados. Numa primeira comparação com as exatidões temáticas discutidas no ponto anterior, é possível determinar que a presente estratégia de validação contribuiu com percentagens que se distinguem da estratégia de validação com amostra aleatória. Em alguns casos particulares as classificações produzidas apresentaram exatidão superior, bem como o inverso, pela utilização de determinados dados de treino.

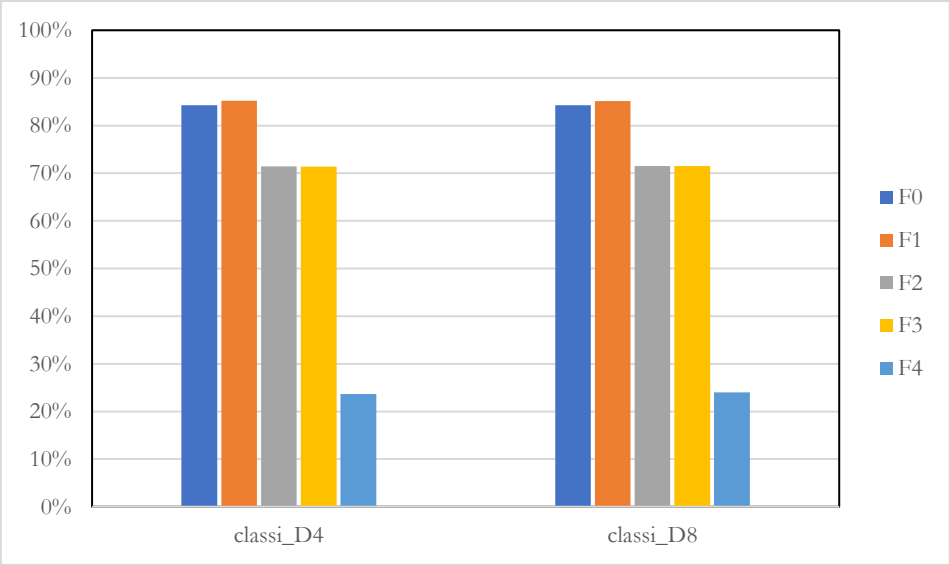


Figura 37 – Síntese dos resultados de exatidão temática obtidos para as classificações pela estratégia de validação com comparação entre mapas.

A partir da representação gráfica, relativa às estatísticas registadas para a exatidão temática das classificações produzidas pelos diferentes *datasets*, confere-se uma semelhança visual no gráfico construído a partir da estratégia de validação com os pontos de amostra aleatória. Assim como no ponto anterior, os dados de treino com exatidão mais elevada foram registados pelas informações de F0 e F1. Os dados F1, distinguem-se de F0, uma vez que a percentagem de exatidão atingiu 85%, e os dados originais OSM um valor de 84%, assim como na primeira estratégia. Depois destes, reconhece-se a sequência decrescente dos dados de treino, aliados às novas regras de filtragem introduzidas. Para os dois *datasets* F2 e F3, as percentagens analisadas foram idênticas, contando com uma percentagem de 71% para os

dois conjuntos de classificações. Ao contrário do que foi discutido na primeira estratégia de validação, as exatidões dos respetivos dados de treino apresentaram uma variação positiva, tendo em vista que na secção anterior caracterizavam-se com 65% de exatidão temática. Para o último dado de treino, o contributo de F4 no processo de classificação revelou-se, mais uma vez, como a percentagem de exatidão mais baixa, de 24% nas duas classificações. Em comparação com a primeira estratégia, o valor de exatidão temática para os dados F4, apresentou-se mais baixo, com uma diferença de 4%.

Assim como na primeira estratégia, a análise de resultados não estaria completa sem realizar a mesma análise da medida *f1-score* para as diferentes classes na presente estratégia de validação. Para além disso, como na primeira estratégia, foram encontradas algumas incertezas na variação dos valores de *f1-score* em determinadas classes, foi importante fazer a análise e discussão da medida neste ponto. Esta decisão foi deveras importante, com intuito de aumentar o grau de confiança das propriedades encontradas nas classes, responder a dúvidas procedentes, e de certa forma, complementar a discussão de resultados.

Neste sentido, as Figuras 38 e 39 que se seguem, fazem referência para a construção gráfica dos padrões de *f1-score*, para as determinadas classes informativas, em relação aos dados de treino utilizados. A análise desta medida, foi acompanhada do mesmo modo das tabelas de contingência encontradas no Anexo E1, como forma de completar discussão de resultados.

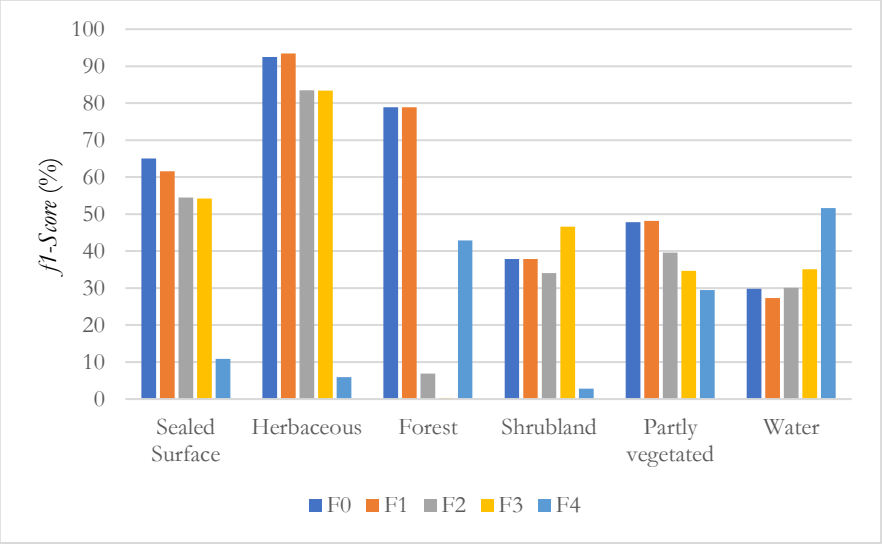


Figura 38 – Valor *f1-score* de cada classe classi_D4. Estratégia de validação: Comparação entre mapas.

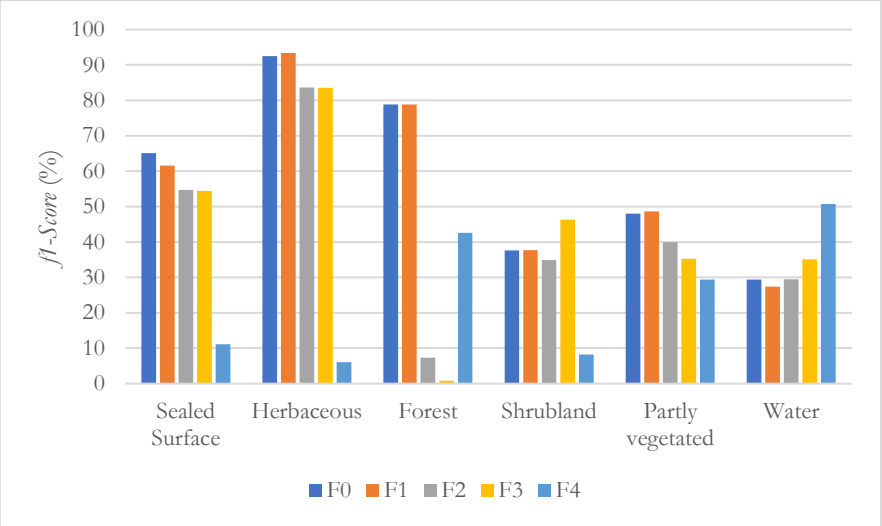


Figura 39 – Valor *f1-score* de cada classe classi_D8. Estratégia de validação: Comparação entre mapas.

Quanto ao número representativo das classes encontradas nos diferentes gráficos das figuras anteriores, é pertinente referir a identificação da classe *water*, dado que na discussão dos valores de *f1-score* da estratégia anterior, não foi apresentada. Esta situação associa-se inteiramente à falta de incidência de pontos na respetiva classe nos resultados da validação da primeira estratégia. Dadas as inconsistências providenciadas pela amostra aleatória, a estratégia de comparação entre mapas mostrou-se mais uma vez, fonte de auxilio para completar os resultados e conferir um grau de fiabilidade da medida de validação.

Todavia, para o estudo do comportamento das diferentes classes na medida *f1-score*, reconhece-se uma superioridade dos valores encontrados pelos dados de treino F0 e F1 para as ocupações do solo *sealed surface*, *herbaceous*, *forest* e *partly vegetated or non-vegetated*. Para a classe de espaços artificializados, assim como na estratégia anterior, os dados F0 e F1 apresentaram as percentagens de *f1-score* mais altas, no entanto, no respetivo paradigma as informações originais OSM mostraram ser superiores ao *dataset* F1. Enquanto F1 detêm de uma percentagem de 62%, os dados F0 assentam em 65% de *f1-score*. Logo a seguir, encontram-se as informações de F2 e F3, com valores idênticos de 55% e mais baixo o *f1-score* de F4. Este quadro admite uma maior capacidade dos treinos F0 e F1 para classificar *sealed surface*, e uma total inabilidade dos dados F4, uma vez que representa um *f1-score* de 8% e um erro de omissão, de 94% com erros associados às temáticas de *herbaceous* e *shrubland*.

Avançando para a classe *herbaceous* foram igualmente registados por parte dos dados de treino F0, F1, F2 e F3 valores elevados. No caso específico da respetiva classe, o *dataset* F1 apresenta a percentagem mais elevada, de 93%, seguido do dado F0, com 92%. Em relação aos dados de treino F2 e F3, as medidas de *f1-score* assinaladas não desfavoreceram significativamente a sua capacidade de distribuir geograficamente a classe herbáceas, visto que foram registados 84% para ambos os dados de treinos. Nesta situação, prevê-se uma forte aptidão das informações encontradas nos quatro primeiros dados de treino para individualizar a respetiva classe na área de estudo, contudo F4 foi isento disso. Com um *f1-score* de 6%, os dados de treino F4 mostraram uma total incapacidade de classificar informações sobre *herbaceous*. Esta ideia é reforçada com o erro de omissão associado, de 97%.

Perante a interpretação estatística de *f1-score* para a classe *forest*, depara-se com a hegemonia dos valores registados pelos dados de treino F0 e F1, que nesta situação concreta preservaram percentagens idênticas de 79%. Segue-se as informações relativas a F4, com o terceiro valor mais alto de 42%, de *f1-score*, superior aos restantes dados de F2 e F3 que assumiram as percentagens mais baixas de todas as classes temáticas. Os *f1-score* registados por estes *datasets* remetem para as altas percentagens de erros de omissão de 96% para F2 e de 99% para F3, admitindo inconsistências nas novas regras filtragem. Na grande maioria, as confusões dos respetivos dados de treino passam pelas informações da classe *forest* e *herbaceous* ou *shrubland*.

Segundo a análise feita até ao momento, podem desde já destacar-se os treinos F0 e F1 para as classes *sealed surface*, *herbaceous* e *forest* com percentagens satisfatórias, seguida de uma correta individualização, mas de valores inferiores, para as classes aliadas a áreas artificializadas e herbáceas por parte das amostras F2 e F3. Os dados F4 merecem um certo destaque para as informações sobre florestas. Através de uma análise sustentada com a discussão sobre a cobertura dos dados de treino nas secções anteriores, prevê-se que os critérios de filtragem de F4 detenham de um sentido assente para a individualização de informações sobre *forest*, enquanto as condições de F2 e F3 estabelecem uma relação para informações ligadas às classes *herbaceous* e *sealed furface*.

Para a classe *shrubland*, a análise dos resultados obtidos pela medida *f1-score* remete para uma situação distinta do que se tem vindo a verificar. O valor mais alto registado para a classe foi apresentado pelos dados de treino F3, com 47%, representando superioridade relativamente aos dados de F0 e F1. Estes *datasets*, com o segundo valor mais alto, igualaram-se com 38% da medida de *f1-score*. Praticamente a seguir estão os dados de treino F2, com uma percentagem de 35% na melhor classificação e na avaliação mais baixa, o treino de F4, com

apenas 8%. Neste quadro, o conjunto de informações dos dados F3 detêm da maior capacidade para classificar ocupação do solo na temática das *shrublands*, uma vez que apresentou o erro de omissão mais baixo de todos os dados de treino.

Em alusão às percentagens de *f1-score* da classe *partly vegetated or non-vegetated* obtidas para os diferentes dados de treino, foi possível assinalar uma discordância nos resultados, em função da estratégia de validação. Nesta situação, a respetiva inconsistência associa-se à estratégia de amostra aleatória, uma vez que a sua credibilidade deve ser sempre questionada. Tendo em conta que a representatividade da amostra de pontos é feita de forma aleatória, a interseção em determinadas classes informativas pode ser insuficiente e desequilibrada, acarretando inconsistências na validação de mapas. Posto isto, para os presentes resultados, distinguem-se os *datasets* F0 e F1 com evidências para os valores mais elevados, de 48% e 49% respetivamente, seguida de uma redução sequencial das percentagens F2, F3 e F4. A decrescência dos valores, registaram para F2, uma percentagem de 40%, F3 de 35% e F4 de 29%. Tal situação, remete para uma evidente incapacidade das novas regras de filtragem realizarem uma seleção de amostras suficientemente representativa, conduzindo para uma possível perda de informação considerável nos dados de treino. Assim como na discussão feita para as primeiras classes temáticas, a supremacia das regras originais de F1 prevalece.

Os valores registados para a última classe correspondente, com referência para as informações de *water* o comportamento das percentagens de *f1-score*, perante os vários dados de treino, remetem para uma situação contrária de todo o inventário estatístico anterior. Face ao exposto, os resultados aparentam padrões mais baixos na utilização dos dados de treino F1, com o valor de 27% seguido de valores ligeiramente superiores para as classificações F0 de 29%. Para os restantes registos de *f1-score*, a subida da percentagem inicia com o *dataset* F2, e assim aumenta até alcançar o treino de F4 que assinala a padrão da classe mais alto da medida com 51%. Este cenário acaba por se considerar como insólito dada a hegemonia dos dados F4, tendo em vista que assinalou o valor mais baixo do erro de omissão.

6.2.6 Síntese de resultados

Deve refletir-se que numa primeira fase, onde foi realizada a comparação do treino produzido com a referência, os dados de treino produzidos da aplicação de novas regras de filtragem, apresentaram uma superioridade consistente, em relação aos dados F0 e F1. Quanto aos valores de exatidão temática, os resultados assinalados, caracterizaram-se por uma sequência lógica de F0 para F1, de F1 para F2 e F3. No respetivo ponto, os dados de treino F2 e F3, destacaram-se com as exatidões mais altas, assim como na análise estatística de *f1-score*, para a grande maioria das classes temáticas analisadas. Este cenário conferiu um certo peso, na avaliação da metodologia desenvolvida neste estudo, uma vez que a estratégia adotada para identificação de pixéis puros, tornou possível a definição de novas regras de filtragem capazes de realizar uma seleção detalhada das informações exatas e a eliminação de impurezas nos dados de treino.

Para a segunda fase de resultados, aliada à validação da classificação de imagens de satélite e produção de mapas de ocupação, evidenciou-se um cenário distinto. A partir das duas estratégias de validação adotadas, foi assinalada uma hegemonia inversa nos resultados analisados, conferindo destaque aos dados de treino F0 e F1. Considerando o cálculo de medidas de validação, como a exatidão temática auxiliada do *f1-score* para os mapas produzidos, os respetivos dados de treino, conferiram um grau de superioridade na avaliação das classificações, enquanto os padrões mais baixos registados, remeteram para os dados de treino F2, F3 e F4. A título de referência para o estudo Fonte, *et al.* (2020), os resultados de exatidão dos mapas gerados para as duas áreas de estudo, mostraram que as regras de filtragem aplicadas nos dados de treino detinham a capacidade de excluir as inconsistências nos vários *datasets,* melhorando exponencialmente a exatidão temática das classificações. Neste exercício em particular, acabou por se verificar o contrário, uma vez que as novas regras de filtragem se tornaram prejudiciais aos resultados da classificação de imagens.

O respetivo cenário, conduziu para a ideia de que a implementação de métodos para a criação de novas regras de filtragem favorece a fase do treino, mas no fundo, desfavorece as avaliações dos mapas de ocupação do solo. Isto foi comum para as duas estratégias de validação, destacando a visão de que a filtragem dos dados abarca uma influência significativa na determinação dos resultados das classificações. Dada à definição de regras de filtragem de um grau de rigor superior, os dados de treino passam por processos de seleção exigente, conduzindo a uma perda significativa da diversidade da informação contida no conjunto dos dados. Tal fator, concede a carência de heterogeneidade das diferentes classes temáticas,

fazendo com que as informações presentes nos dados de treino deixem de ser suficientemente consideráveis para que o algoritmo treinado realize a correta individualização entre as classes informativas, causando confusões na tentativa de distinção de classes de ocupação do solo.

7 Conclusão

7.1 Considerações finais

Mediante do desenvolvimento desta investigação, além das contribuições resultantes da sua exploração, surgiram um conjunto de considerações importantes. Estas procuraram tornar mais claro, conhecimentos aliados ao desenvolvimento de metodologias para filtrar dados de treino recolhidos de comunidades IGV (como foi o caso do OSM), tendo em vista os métodos apropriados à identificação automática da informação crucial para a classificação de imagens de satélite. No caso desta investigação, acredita-se que, para além de inovadora, apresenta questões que, ao pormenor, respondem à carência de trabalhos específicos para temas associados à produção de dados de treino por intermédio de processos de filtragem e classificação de imagens, tendo em vista a exploração de técnicas inovadoras e automáticas.

Numa fase inicial, teve-se o cuidado de abordar as rápidas alterações da superfície do solo e a crescente necessidade de acompanhar este avanço, problematizando as desvantagens dos métodos tradicionais, determinados como técnicas que requerem longos períodos de produção, desde a aquisição de dados, etapas de processamento e tratamento até alcançar a informação bidimensional relativa às características da superfície terrestre. No combate às referentes metodologias de produção mapas, que envolvem despesas avultadas, nasce a procura constante por métodos automáticos, como forma de tirar proveito do aglomerado de dados de imagens e informação disponível gratuitamente, agilizando o custo monetário e auxiliando na resposta às necessidades de criação de mapas de uma forma rápida e de fácil atualização. Ademais, ficou bem patente que procurava-se produzir informação de ocupação do solo para a área de estudo estabelecida, atendendo ao contexto de estreiteza de cobertura de dados.

Considerando os problemas específicos que se foram manifestando ao longo da exploração do trabalho, julga-se que foi possível organizar um conjunto de critérios, que passaram pelas tomadas de decisão de interpretação técnica mais adequada, permitindo superar fases metodológicas, tanto ao nível de tratamento de imagens de satélite e dos dados de elevação, bem como na concretização da prática metodológica através de métodos programadas inovadores tendo como referência a metodologia implementada no artigo de Fonte, *et al.* (2020). Aliada a uma metodologia empreendedora, o desenvolvimento de conhecimentos a nível pessoal são despertados, e pelo fato deste estudo ter exigido programação de etapas

metodológicas, finaliza-se esta dissertação com um maior domínio por parte destas tecnologias.

Voltando o olhar para os objetivos iniciais que se tinham sido propostos para este trabalho, perante os resultados adquiridos no decorrer de todo o exercício, configura-se desde já uma verdadeira contribuição informativa da ocupação do solo para o território insular da ilha Terceira, ponto que se mostrou ser cumprido. Na avaliação do impacto do aumento da utilização do número de imagens, bem como na adição de outras variáveis informativas, auxiliada de dados de elevação e imagens de outros satélites foram encontradas algumas tendências relacionadas por meio dos resultados analisados. De um modo geral, verificou-se que o comportamento das classificações se correlacionava com o número de informações adicionais, mostrando que uma maior combinação de *features* superava as classificações de dados de um único satélite. Por sua vez, entendeu-se que a harmonização da resolução espacial dos diferentes conjuntos de dados e o contributo informativo restrito de determinadas bandas, das composições de imagens, provinham para percentagens de exatidões globais próximas. Mesmo que não se tenha efetivado uma análise investigativa aprofundada na etapa do respetivo exercício, em conjunto com a pesquisa bibliográfica realizada levou a admitir que, o aumento de informações combinadas pelos diferentes dados, despertavam para uma tendência de redundância da informação, escorando, de certa maneira, as exatidões dos mapas, sem contribuição para melhorias ou aumento da exatidão temática.

Os dados OSM têm vindo a ganhar uma crescente atenção na criação de mapas de ocupação do solo, mas é importante ter consciência de que a grande maioria das informações encontradas nos dados originais podem abarcar um número elevado de imprecisões e inconsistências que necessitam de ser filtradas. A iniciativa dada pelo estudo investigado de Fonte, *et al.* (2020), foi sem dúvida um primeiro passo exploratório das potencialidades destes dados e uma base fundamental para esta dissertação, dado aos conceitos e conhecimentos transmitidos. Nesta lógica e em alusão para o que diz respeito, à atualização da metodologia para produção automática de dados de treino para classificação da ocupação do solo, com ênfase para a identificação de novas regras para a filtragem de informações, foram desvendadas um conjunto de evidências investigadas.

Como consequência do caminho exploratório adotado ao longo do estudo, entendeu-se, a partir da discussão de resultados, que as classificações através dos dados de treino F1, decorrentes das regras filtragens originais proporcionam avaliações mais altas e um melhor desempenho na individualização de classes temáticas. Para os conjuntos dos dados de treino

procedentes de novas regras filtragem, intitulados por F2, F3 e F4 conduziram a classificações menos eficazes, acabando por desenvolver exatidões mais imprecisas do que com a utilização dos dados brutos descarregados do OSM. Contudo, dada a uma análise de resultados aprofundada, mostrou-se que na fase do treino, as informações contidas nos dados de treino resultantes das novas regras de filtragem, superavam nas várias medidas de validação discutidas. A este ponto, reconhece-se uma certa eficácia na metodologia implementada para a definição de novas regras de filtragem, visto que a fase do treino do classificador é excelente e acarreta confiança nos dados filtrados, mas em incompatibilidade a etapa de classificação é insuficiente. Tendo em conta os problemas encontrados, parece que ao filtrar demasiado os dados de treino, através da definição de regras rigorosas leva à perda de muita diversidade da informação e dados de treino adequados à classificação de imagens. Neste sentido, a perda de heterogeneidade da informação é um fator que contribui para uma resposta espetral insuficiente, incapaz de determinar todas as características das classes e gerar classificações desajustadas. Perante um confronto direto entre as regras originais do artigo de referência de Fonte, *et al.* (2020) e as novas condições de filtragem desenvolvidas no presente estudo, confirma-se melhores classificações para as regras do artigo, com a conceção de mapas fidedignos.

7.2 Recomendações de trabalho futuro

Esta investigação poderia ter contribuído com mais conhecimento, caso o tempo de exploração do trabalho fosse, de facto, mais alargado, colocando a hipótese de ampliar investigação de novos métodos para tratar o conjunto de dados de treino, na procura de outros resultados das classificações de imagens. Mas, não foi possível, deixando estas tarefas para trabalhos futuros. Neste pensamento, poder-se-á deixar claro para etapas pendentes os seguintes pontos:

i. Tentativa de melhorar a qualidade dos dados de treino auxiliado apenas com índices radiométricos parece insuficiente e, portante, parece ser válido realizar a combinação com outras estratégias que visam alcançar o respetivo propósito, através de análise de *clusters*;
ii. Analisar iterativamente a probabilidade de pertença de uma classe, através da realização do processo de classificação, e remover as células que não têm

pelo menos 50% de probabilidade de pertencerem a uma classe (caso das confusões comuns entre classes) e realizar novamente a classificação;

iii. Realizar o processo de classificação, obter a probabilidade de pertença a cada classe e identificar pixéis com probabilidades distribuídas por menos de duas classes (i.e. probabilidade referente a uma só classe). Verificar se essas células foram utilizadas nos dados de treino e caso não tenham sido introduzidas, passam a ser incluídas e repete-se o processo de classificação de imagens;

iv. Em vez de se procurar a remoção de células problemáticas, pode-se procurar classificar o treino em função do grau de certeza que se tem sobre a pertença de um pixel a uma classe e manter as informações problemáticas nos dados de treino;

v. Utilizar outro algoritmo de classificação de imagens, em vez do *random forest*, realizar testes com *support vector machine*, *neural networks*;

vi. Alargar as características da informação das respostas espetrais, com a introdução de novos métodos para aplicação da metodologia;

vii. A extração dos pixéis puros da mesma área de classificação pode representar de forma adequada a variação dos valores dos índices radiométricos nas determinadas classes, e, consequentemente, incluir ao longo do processo de filtragem a informação necessária à correta classificação de imagens;

viii. Utilizar outras áreas para obter novos valores de filtragem dos dados, que detenham a capacidade de abranger todas as características necessárias à individualização de classes mantendo a sua heterogeneidade;

ix. Na realidade, não foi feito nada relacionado com o EAGLE, mesmo que se tenha dado referência a este. No futuro faz, sentido tentar fazer classificações baseadas nas classes do EAGLE, tendo em vista a mudança do paradigma na produção destes mapas.

Referências bibliográficas

1. ABRAMS, Michael; CRIPPEN, Robert & FUJISADA, Hiroyuki (2020) – "ASTER Global Digital Elevation Model (GDEM) and ASTER Global Water Body Dataset (ASTWBD)". **Remote Sensing**, Vol. 12, Issue 7, 1-12.
2. ACHARKI, Siham (2022) – "PlanetScope contributions compared to Sentinel-2, and Landsat-8 for LULC mapping". **Remote Sensing Applications: Society and Environment**, Vol. 27, 1-15.
3. ALBARAKAT, Reyadh & LAKSHMI, Venkataraman (2019) – "Comparison of Normalized Difference Vegetation Index Derived from Landsat, MODIS, and AVHRR for the Mesopotamian Marshes Between 2002 and 2018". **Remote Sensing**, Vol. 11, Issue 10, 1-16.
4. AMOAKOH, Alex; APLIN, Paul; AWUAH, FERNANDEZ, Delgado *et al.* (2021) – "Testing de Contribution of Multi-Source Remote Sensing Features for Random Forest Classification of the Greater Amanzule Tropical Peatland". **Sensors**, Vol. 21, Issue 10, 1-25.
5. AMRO, Israa; MATEOS, Javier; VEGA, Miguel; MOLINA, Rafael & KATSAGGELOS, Aggelos (2011) – "A survey of classical methods and new trends in pansharpening of multispectral images". **EURASIP Journal on Advances in Signal Processing**. N.º 79, 1-22.
6. ANDERSON, James; HARDY, Ernest; ROACH, John & WITMER, Richard (1976) – A Land Use and Land Cover Classification System for use with Remote Sensor Data. *United States Government Printing Office*, Vol. 964.
7. ARNOLD, Stephan; KOSZTRA, Barbara; BANKO, Gebhard *et al.* (2013) – The EAGLE concept – A vision of a future European Land Monitoring Framework. *In Proceedings of the 33rd EARSeL Symposium towards Horizon*, Matera, Italy, 3-6 junho 2013 (1-17).
8. ARNOLD, Stephan; KOSZTRA, Barbara; BOCK, Michael *et al.* (2021) – Explanatory Documentation of the EAGLE Concept. *European Environment Agency*, Copenhagen, v. 3.1.2.
9. ARONOFF, Stan (1989) – "Geographic Information Systems: A Management Perspective". **Geocarto International**, Vol. 4, Issue 4, 58-58.

10. ASF (s.d.) – *ALOS Phased Array type L-band Synthetic Aperture Radar*. **Disponível** no endereço URL: https://asf.alaska.edu/data-sets/sar-data-sets/alos-palsar/alos-palsar-about/ (acedido em 5 fevereiro, 2023).
11. ASF (s.d.) – *ASF's Radiometric Terrain Correction Project*. **Disponível** no endereço URL: https://asf.alaska.edu/data-sets/derived-data-sets/alos-palsar-rtc/alos-palsar-radiometric-terrain-correction/ (acedido em 5 fevereiro, 2023).
12. BELGIU, Mariana & DRĂGUȚ, Lucian (2016) – "Random forest in remote sensing: A review of applications and future directions". **ISPRS Journal of Photogrammetry and Remote Sensing,** Vol. 114, 24-31.
13. BERGONSE, Rafaello & BIDARRA, João (2010) – "Probabilidade bayesiana e regressão logística na avaliação da susceptibilidade à ocorrência de incêndios de grande magnitude. **Finisterra**, Vol. 45, N.º 89, 79-104.
14. BORGES, Luiz (2014) – **Python para Desenvolvedores**. Novatec, 2ª Edição, São Paulo.
15. BORRA, Surekha; THANKI, Rohit & DEY, Nilanjan (2019) – **Satellite Image Analysis: Clustering and Classification**. Springer Nature, Singapura.
16. BREIMAN, Leo (2001) – "Random Forest". **Machine Learning**, N.º 45, 5-32.
17. BROWN, Christopher; BRUMBY, Steven; GUZDER-WILLIAMS, Brookie *et al.* (2022) – "Dynamic World, Near real-time global 10 m land use land cover mapping". **Scientific Data**, N.º 251, 1-17.
18. BUCHROITHNER, Manfred & FERNÁNDEZ, Pablo (2011) – "Cartography in the Contexto of Scienses: Theoretical and Technological Considerations". **The Cartographic Journal**, Vol. 48, Issue 1, 4-10.
19. BÜTTNER, György; KOSZTRA, Barabara; KLEESCHULTE, Stefan *et al.* (2021) – Copernicus Land Monitoring Service CORINE Land Cover. User Manual. *European Environment Agency*, Copenhagen, Denmark.
20. CAETANO, Mário & MARCELINO, Filipe (2019) – **Especificações Técnicas da Carta de Uso e Ocupação do Solo (COS) de Portugal Continental para 2018**. Direção Geral do Território, Lisboa.
21. CARRÃO, Hugo; CAETANO, Mário; FREIRE, Sérgio & NUNES, António (2002) – Carta de ocupação do solo e avaliação do estado da vegetação com imagens de satélite para prevenção de fogos florestais. *Encontro de Utilizadores de Informação Geográfica (ESIG)*, Oeiras, Portugal, 13-15 novembro 2002 (1-10).
22. CARRASCO, Luis; O'NEIL, Aneurin; MORTON, Daniel & ROWLAND, Clare (2019) – "Evaluating Combinations of Temporally Aggregated Sentinel-1, Sentinel-2 and

Landsat-8 for Land Cover Mapping with Google Earth Engine". **Remote Sensing**, Vol. 11, Issue 3, 1-21.

23. CCI (s.d.) – *GMTED2010*. **Disponível** no endereço URL: https://www.usgs.gov/coastal-changes-and-impacts/gmted2010 (acedido em 3 fevereiro, 2023).
24. COEURDELEY, Laurent & FERNANDEZ, Karine (2012) – Pléiades Imagery User Guide. *Airbus Defense and Space Intelligence*, France CNES, Tech. Rep. Report No. USRPHR-DT-125-SPOT-2.0.
25. COEURDEVEY, Laurent & SOUBIRANE, Jérôme (2013) – SPOT 6/7 Imagery User Guide. *Astrium Services*, France, SI/DC/13034-v1.0.
26. CHEN, Jun; CHEN, Jin; LIAO, Anping *et al.* (2015) – "Global land cover mapping at 30m resolution: A POK-based operational approach". **ISPRS Journal Photogrammetry and Remote Sensing**. Vol. 103, 7-27.
27. CHENG, Qi; VASHNEY, Pramod & ARORA, Manoj (2006) – "Logistic Regression for Feature Selection and Soft Classification of Remote Sensing Data". **IEEE Geoscience and Remote Sensing Letters**, Vol. 3, Issue 4, 491-494.
28. COLLIN, Antoine; HENCH, James; PASTOL, Yves *et al.* (2018) – "High resolution topobathymetry using a Pleiades-1 triplet: Moorea Island in 3D". **Remote Sensing of Environment**, Vol. 208, 109-119.
29. Copernicus (2022) – Product Specification and User Manual: Raster Product. *European Environment Agency*, Issue 2.1a, Service Contract No. EEA/DIS/R0/19/012.
30. Copernicus (s.d) – *Copernicus DEM – Global and European Digital Elevation Model (COP-DEM)*. **Disponível** no endereço URL: https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1?tab=metadata (acedido em 4 fevereiro, 2023).
31. COSME, António (2012) – **Projeto em Sistemas de Informação Geográfica**. LIDEL – Edições Técnicas, Lisboa, 361 p.
32. DANIELSON, Jeffrey & GESCH, Dean (2011) – Global Multi-resolution Terrain Elevation Data 2010 (GMTED2010). 34 p. **Disponível** no endereço URL: https://pubs.usgs.gov/of/2011/1073/pdf/of2011-1073.pdf (acedido 20 abril, 2023).
33. DELIRY, Sayed; AVDAN, Zehra & AVDAN, Uğur (2020) – "Extracting urban impervious surfaces from Sentinel-2 and Landsat-8 satellite data for urban planning and environmental management". **Environmental Science and Pollution Research**, Vol. 28, 6572-6586.

34. DONNAY, Jean-Paul; BARNSLEY, Mike & LONGLEY, Paul (2000) – **Remote Sensing and Urban Analysis.** GISDATA 9, 1ª Edição, Londres.
35. DRUSCH, Matthias; DEL BELLO, Umberto; CARLIER, Stefane *et al.* (2012) – "Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services". **Remote Sensing of Environment**, Vol. 120, 25-36.
36. EROS Center (2018) – *USGS EROS Archive – Digital Elevation – Shuttle Radar Topography Mission (SRTM) 1 Arc-Second Global.* **Diponível** no endereço URL: https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-shuttle-radar-topography-mission-srtm-1#overview (acedido em 2 fevereiro, 2023).
37. ESA (2012a) – *Copernicus: Sentinel-2*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/copernicus-Sentinel-2 (acedido em 20 janeiro, 2023).
38. ESA (2012b) – *Landsat-7*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/Landsat-7 (acedido em 21 janeiro, 2023).
39. ESA (2012c) – *PRISMA (Hyperspectral)*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/prisma-hyperspectral (acedido em 23 janeiro, 2023).
40. ESA (2012d) – *Pleiades- HR (High-Resolution Optical Imaging Constellation of CNES)*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/pleiades (acedido em 24 janeiro, 2023).
41. ESA (2022a) – *Landsat-8/LDCM*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/Landsat-8-ldcm (acedido em 22 janeiro, 2023).
42. ESA (2022b) – *Landsat-9*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/Landsat-9 (acedido em 22 janeiro, 2023).
43. ESA (2023a) – *Copernicus: Sentinel-1*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/copernicus-Sentinel-1 (acedido em 20 janeiro, 2023).
44. ESA (2023b) – *SPOT-6 and SPOT-7 (Azersky)*. **Disponível** no endereço URL: https://www.eoportal.org/satellite-missions/spot-6-7 (acedido em 23 janeiro, 2023).
45. ESA (s.d.) – *Sentinel-1 SAR User Guide*. **Disponível** no endereço URL: https://Sentinel.esa.int/web/Sentinel/user-guides/Sentinel-1-sar (acedido em 17 janeiro, 2023).

46. ESA (s.d.) – *Sentinel-2 MSI User Guide*. **Disponível** no endereço URL: https://Sentinel.esa.int/web/Sentinel/user-guides/Sentinel-2-msi (acedido em 21 janeiro, 2023).
47. ESRI (2022) – *Semntinel-2 10m land use/land cover time series*. **Disponível** no endereço URL: https://www.arcgis.com/home/item.html?id=cfcb7609de5f478eb7666240902d4d3d# (acedido em 10 janeiro, 2023).
48. European Commission (2020) – Mapping guide v6.2 for a European Urban Atlas. Regional Policy.
49. FAOUZI, Johann & JANATI, Hichan (2020) – "pyts: A python package for time series classification". **Journal of Machine Learning Research**, N.º 21, 1720-1725.
50. FERREIRA, Zuleide & CABRAL, Pedro (2021) – Vertical Accuracy Assessment of ALOS PALSAR, GMTED2010, SRTM and Topodata Digital Elevation Models. *International Conference on Geographical Information Systems Theory, Applications and Management*, Virtual Event, 23-25 abril 2021 (116-124).
51. FONTE, Cidália; PATRIARCA, Joaquim.; JESUS, Ismael & DUARTE, Diogo (2020) – "Automatic Extraction and Filtering of OpenStreetMap Data to Generate Training Datasets for Land Use Land Cover Classification". **Remote Sensing**, Vol. 12, Issue 20, 1-31.
52. FORKUOR, Gerald; DIMOBE, Kangbeni; SERME, Idriss & TONDOH, Jerome (2017) – "Landsat-8 vs. Sentinel-2: examining the added value of Sentinel-2's red-edge bands to land-use and land-cover mapping in Burkina Faso". **GIScience & Remote Sensing**, Vol. 55, Issue 3, 331-354.
53. FRIEDL, Mark & BRODLEY, Carla (1997) – "Decision tree classification of land cover from remotely sensed data". **Remote Sensing of Environment**, Vol. 61, Issue 3, 399-409.
54. GAO, Bo-Cai (1996) – "NDWI—A normalized difference water index for remote sensing of vegetation liquid water from space". **Remote Sensing of Environment**, Vol. 58, Issue 3, 257-266.
55. GARBRECHT, Jurgen & MARTZ, Lawrence (2000) – "Digital elevation model issues in water resources modeling". **Hydrologic and hydraulic modeling support with geographic information systems**, 1-28.
56. GILLIES, Sean (2013) – The shapely User Manual. 40 p. **Disponível** no endereço URL: https://sethc23.github.io/wiki/Python/The_Shapely_User_Manual_%E2%80%94_Shapely_1.2_and_1.3_documentation.pdf (acedido em 11 setembro, 2023).

57. GILLIES, Sean (2019) – rasterio Documentation. 269 p. **Disponível** no endereço URL: https://buildmedia.readthedocs.org/media/pdf/rasterio/latest/rasterio.pdf (acedido em 12 setembro, 2023).
58. GOODCHILD, Michael & LI, Linna (2012) – "Assuring the quality of volunteered geographic information". **Spatial Statistics**, Vol. 1, 110–120.
59. GU, Zhujun; SHI, Xuezheng; LI, Lin; YU, Dongsheng; LIU, Liusong & ZHANG, Wentai (2011) – "Using multiple radiometric correction images to estimate leaf area index". **International Journal of Remote Sensing**, Vol. 32, Issue 24, 9441-9454.
60. GUARINI, Rocchina; LOIZZO, Rosa; FACHINETTI, Claudia *et al.* (2018) – Prisma Hyperspectral Mission Products. *IEEE International Geoscience and Remote Sensing Symposium*, Valencia, Spain, 22-27 julho 2018 (179-182).
61. GURJAR, Suresh & Padmanabhan, Narayan (2005) – "Study of various resampling techniques for high-resolution remote sensing imagery". **Journal of the Indian Society of Remote Sensing**, Vol. 33, N.º 1, 113-120.
62. HAN, Jiawei; PEI, Jian & TONG, Hanghang (2022) – **Data Mining: Concepts and Techniques**. Morgan Kaufmann, 4ª Edição, 752 p.
63. HE, Chunyang; SHI, Peijun; XIE, Dingyong & ZHAO, Yuanyuan (2010) – "Improving the normalized difference built-up index to map urban built-up areas using a semiautomatic segmentation approach". **Remote Sensing Letters**, Vol. 1, Issue 4, 213–221.
64. HONKAVAARA, Eija; SAARI, Heikki; KAIVOSOJA, Jere *et al.* (2013) – "Processing and assessment of spectrometric, stereoscopic imagery collected using a lightweight UAV spectral camera for precision agriculture". **Remote Sensing**, Vol. 5, Issue 10, 5006–5039.
65. HU, Tan-gao; XU, Jun-feng; ZHANG, Deng-rong *et al.* (2013) – "Hard and SoftClassification Method of Multi-Spectral Remote Sensing Image Based on Adaptive Thresholds". **Spectroscopy and Spectral Analysis**, Vol. 33, N.º 4, 1038–1042.
66. IHLEN, Vaughn (2019a) – Landsat 7 (L7) Data Users Handbook. 151 p. USGS Landsat User Serv. **Disponível** no endereço URL: https://d9-wret.s3.us-west-2.amazonaws.com/assets/palladium/production/s3fs-public/atoms/files/LSDS-1927_L7_Data_Users_Handbook-v2.pdf (acedido em 6 novembro, 2022).
67. IHLEN, Vaughn (2019b) – Landsat 8 (L8) Data Users Handbook. 114 p. USGS Landsat User Serv. **Disponível** no endereço URL: https://d9-wret.s3.us-west-2.amazonaws.com/assets/palladium/production/s3fs-public/atoms/files/LSDS-1574_L8_Data_Users_Handbook-v5.0.pdf (acedido em 8 novembro, 2022).

68. IMANDOUST, Sadegh & BOLANDRAFTAR, Mohammad (2013) – "Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background". **Int. Journal of Engineering Research and Applications**, Vol. 3, Issue 5, 605–610.
69. JACQUIN, Anne; MISAKOVA, Lucie & GAY, Michel (2008) – "A hybrid object-based classification approach for mapping urban sprawl in periurban environment". **Landscape and Urban Planning**, Vol. 84, Issue 2, 152–165.
70. JESUS, Ismael (2020) – *Metodologias automáticas para criação de dados de treino a partir da COS 2015, UA 2012 E OSM*. **Dissertação de mestrado**, Faculdade de Ciências e Tecnologias da Universidade de Coimbra, 216 p.
71. JORDAHL, Kelsey (2016) – GeoPandas Documentation. 38 p. **Disponível** no endereço URL: https://buildmedia.readthedocs.org/media/pdf/geopandas-doc/latest/geopandas-doc.pdf (acedido em 10 setembro, 2023).
72. KARANAM, Hari (2018) – "Study of Normalized Difference Built-Up (NDBI) Index in automatically mapping urban areas from Landsat TM imagery". **International Journal of Scientific Research and Review**, Vol. 8, 239-248
73. KARNIELI, Arnon; AGAM, Nurit; PINKER, Rachel *et al.* (2010) – "Use of NDVI and land surface temperature for drought assessment: Merits and limitations". **Journal of Climate**, Vol. 23, N.º 3, 618-633.
74. KAVZOGLU, Taskin & MATHER, Paul (2003) – "The use of backpropagating artificial neural networks in land cover classification", **International Journal of Remote Sensing**, Vol. 24, Issue 23, 4907-4938.
75. KEELEY, Jon (2009) – "Fire intensity, fire severity and burn severity: A brief review and suggested usage". **International Journal of Wildland Fire**, Vol. 18, Issue 1, 116–126.
76. LAURENCELLE, Jeanne; LOGAN, Tom & GENS, Rudi (2015) – ASF Radiometrically Terrain Corrected ALOS PALSAR products. 12 p. ASF-Alaska Satell. Facil. **Disponível** no URL: https://asf.alaska.edu/wp-content/uploads/2023/06/rtc_product_guide_v1.2-1.pdf (acedido 2 maio, 2023).
77. LAVALLEY, Michael (2008) – "Logistic Regression". **Circulation**, Vol. 117, N.º 18, 2395-2399.
78. LEE, Saro (2005) – "Application of logistic regression model and its validation for landslide susceptibility mapping using GIS and remote sensing data". **International Journal of Remote Sensing**, Vol. 26, Issue 7, 1477-1491.

79. LI, Erzhu; DU, Peijun; SAMAT, Alim *et al.* (2015) – "An automatic approach for urban land-cover classification from Landsat-8 OLI data". **International Journal of Remote Sensing**, Vol. 36, Issue 24, 5983-6007.
80. LOIZZO, Rosa; GUARINI, Rocchina; LONGO, Francesco *et al.* (2018) – Prisma: The Italian Hyperspectral Mission. *IEEE International Geoscience and Remote Sensing Symposium*, Valencia, Spain, 22-27 julho 2018 (175-178).
81. LUO, Nianxue; WAN, Taili; HAO, Huaixo & LU, Qikai (2019) – "Fusing High-Spatial-Resolution Remotely Sensed Imagery and OpenStreetMap Data for Land Cover Classification Over Urban Areas". **Remote Sensing**, Vol. 11, Issue 1, 1-21.
82. MANDANICI, Emanuele & BITELLI, Gabriele (2016) – "Preliminary Comparison of Sentinel-2 and Landsat 8 Imagery for a Combined Use". **Remote Sensing**, Vol. 8, Issue 12, 1-10.
83. MAROCO, João (2007) – **Análise estatística com utilização do SPSS**. Edições Sílabo, 3ª Edição, Lisboa, 824 p.
84. MCFEETERS, Stuart (1996) – "The use of normalized difference water index (NDWI) in the delineation of open water features". **International Journal of Remote Sensing**, Vol. 17, Issue 7, 1425-1432.
85. MCKINNEY, Wes & TEAM, PD (2015) – Pandas-Powerful python data analysis toolkit. 297 p. Pandas-Powerful Python Data Analysis Toolkit. **Disponível** no endereço URL: https://pandas.pydata.org/pandas-docs/version/0.7.3/pandas.pdf (acedido em 14 setembro, 2023).
86. MEYER, George & NETO, João (2008) – "Verification of color vegetation indices for automated crop imaging applications". **Computers and Electronics in Agriculture**, Vol. 63, Issue 2, 282-293.
87. MISHRA, Sachidananda & MISHRA, Deepak (2012) – "Normalized difference chlorophyll index: A novel model for remote estimation of chlorophyll-*a* concentration in turbid productive waters". **Remote Sensing of Environment**, Vol. 117, 394-406.
88. MOHAMED, Amr (2017) – "Comparative Study of Four Supervised Machine Learning Techniques for Classification". **International Journal of Applied Science and Technology**, Vol. 7, N.º 2, 1-15.
89. MOTA, Andreia; GONÇALVES, António & VIEIRA, António (2012) – "Uso e ocupação do solo em Portugal - Aspetos metodológicos para atualização de cartografia temática". **Aurora Geography Journal**, N.º 4, 101-113.

90. MOUNTRAKIS, Giorgos; IM, Jungho & OGOLE, Caesar (2011) – "Support vector machines in remote sensin: A review". **ISPRS Journal of Photogrammetry and Remote Sensing**, Vol. 66, Issue 3, 247-259.
91. MUKHERJEE, Sandip; JOSHI, Pawan; MUKHERJEE, Samadrita *et al.* (2013) – "Evaluation of vertical accuracy of open source Digital Elevation Model (DEM)". **International Journal of Applied Earth Observation and Geoinformation**, Vol. 21, 205-217.
92. NANDAIA, Morna (2020) – *Interpretação geográfica da classificação do uso e ocupação do solo resultante do processamento e análise de imagens de Satélite*. **Dissertação de Doutoramento**, Faculdade de Ciências e Humanas da Universidade Nova de Lisboa, 278 p.
93. NASA; METI; AIST; Japan Spacesystem; U.S. & Japan ASTER Science Team (2019) – *ASTER Global Digital Elevation Model V003*. **Disponível** no endereço URL: https://lpdaac.usgs.gov/products/astgtmv003/ (acedido em 3 fevereiro, 2023).
94. NETELER, Markus; BOWMAN, M. Hamish; LANDA, Martin & METZ, Markus (2012) – "GRASS GIS: A multi-purpose open source GIS. **Environmental Modelling & Software**, Vol. 31, 124-130.
95. NOMURA, Keiko & MITCHARD, Edward (2018) – "More Than Meets the Eye: Using Sentinel-2 to Map Small Plantations in Complex Forest Landscapes". **Remote Sensing**, Vol. 10, Issue 11, 1-15.
96. O'LOUNGHLIN, Fiachra; PAIVA, Rodrigo; DURAND, Michael *et al.* (2016) – "A multi-sensor approach towards a global vegetation corrected SRTM DEM product". **Remote Sensing of Environment**, Vol. 182, 49-59.
97. OUCHRA, Hafsa & BELANGOUR, Abdessamad (2021) – Satellite image classification methods and techniques: A survey. *IEEE International Conference on Imaging Systems and Techniques (IST)*, Kaohsiung, Taiwan, 24-26 agosto 2021 (1-6).
98. PAL, Mahesh & MATHER, Paul (2003) – "An assessment of the effectiveness of decision tree methods for land cover classification". **Remote Sensing of Environment**, Vol. 86, Issue 4, 554-565.
99. PATIL, Malini (2018) – "Interpolation Techniques in Image Resampling" - **International Journal of Engineering & Technology**. N.º 7, 567-570.
100. PATRIARCA, Joaquim (2016) – *O Software Livre e de Código Aberto na Administração Pública – Dos mitos às questões de natureza legal, ética e de optimização de recursos públicos*. **Dissertação de Mestrado**, Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

101. PATRIARCA, Joaquim; FONTE, Cidália; ESTIMA, Jacinto; ALMEIDA, José & CARDOSO, Alberto (2019) – "Automatic conversion of OSM data into LULC maps: comparing FOSS4G based approaches towards an enhanced performance". **Open Geospatial Data, Software and Standards**, N.º 11, 1-19.

102. PHIRI, Darius; SIMWANDA, Matamyo; SALEKIN, Serajis *et al.* (2020) – "Sentinel-2 Data for Land Cover/Use Mapping: A Review". **Remote Sensing**, Vol. 12, Issue 14, 1-35.

103. POTIN, Pierre; GASCON, Ferran; MECKLENBURG, Susanne *et al.* (2019) – Sentinel High Level Operations Plan (HLOP). *CSC Mission Management Team*, COPE-S1OP-EOPG-PL-15-0020.

104. PRASAD, A. D.; GANASALA, Padma; GUZMÁN, Rafael & Fathian, Farshad (2022) – "Remote sensing satellite data and spectral indices: an initial evaluation for the sustainable development of an urban area". **Sustainable Water Resources Management**, Vol. 8, N.º 19, 1-16.

105. PROBECK, Markus; RUIZ, Inés; RAMMINGER, Gernot *et al.* (2021) – CLC+ Backbone: Set the Scene in Copernicus for the Coming Decade. *IEEE International Symposium on Geoscience and Remote Sensing (IGARSS)*, Brussels, Belgium, 11-16 julho 2021 (2076-2079).

106. QI, Jiaguo; CHEHBOUNI, Abdelghani; HUETE, Alfredo; KERR, Yann & SOROOSHIAN, Soroosh (1994) – "A modified soil adjusted vegetation index". **Remote Sensing of Environment**, Vol. 48, Issue 2, 119-126.

107. QIU, Shi; ZHU, Zhe & HE, Binbin (2019) – "Fmask 4.0: Improved cloud and cloud shadow detection in Landsats 4–8 and Sentinel-2 imagery". **Remote sensing of Environment**, Vol. 231, 1-20.

108. RADOUX, Julien; LAMARCHE, Céline; BOGAERT, Eric *et al.* (2014) – "Automated Training Sample Extraction for Global Land Cover Mapping". **Remote Sensing**, Vol. 6, Issue 5, 3965-3987.

109. RISH, Irina (2001) – "An empirical study of the Naïve Bayes classifier". **IJCAI Workshop on Empirical Methods in Artificial Intelligence**, Vol. 3, N.º 22, 41-46.

110. ROY, David; BOSCHETTI, Luigi & TRIGG, Simon (2006) – "Remote sensing of fire severity: assessing the performance of the normalized burn ratio". **IEEE Geoscience and Remote Sensing Letters**, Vol. 3, Issue 1, 112-116.

111. ROY, David; LI, Jian; ZHANG, Hankui & YAN, Lin (2016) – "Best practices for the reprojection and resampling of Sentinel-2 Multi Spectral Instrument Level 1C data. **Remote Sensing Letters**, Vol. 7, Issue 11, 1023-1032.

112. ROY, David; WULDER, Michael; LOVELAND, Thomas *et al.* (2014) – "Landsat-8: Science and product vision for terrestrial global change research. **Remote Sensing of Environment**, Vol. 145, 154-172.

113. SABINS, Floyd (2007) – **Remote Sensing Principles and Interpretation**. Waveland Pr Inc, 3ª Edição, Illinois.

114. SALEEM, Nayyer; HUQ, Enamul; TWUMASI, Nana *et al.* (2019) – "Parameters Derived from and/or Used with Digital Elevation Models (DEMs) for Landslide Susceptibility Mapping and Landslide Risk Assessment: A Review". **International Journal of Geo-Information**, Vol.8, Issue 12, 1-25.

115. SAYLER, Kristi (2022) – Landsat 9 Data Users Handbook. 115 p. USGS Landsat User Serv. **Disponível** no endereço URL: https://d9-wret.s3.us-west-2.amazonaws.com/assets/palladium/production/s3fs-public/media/files/LSDS-2082_L9-Data-Users-Handbook_v1.pdf (acedido em 11 novembro, 2022).

116. SENARATNE, Hansi; MOBASHERI, Amin; ALI, Ahmed; CAPINERI, Cristina & HAKLAY, Mordechai (2017) – "A review of volunteered geographic information quality assessment methods". **International Journal of Geographical Information Science**, Vol. 31, Issue 1, 139–167.

117. SHAO, Zhenfeng; CAI, Jiajun; FU, Peng *et al.* (2019) – "Deep learning-based fusion of Landsat-8 and Sentinel-2 images for a harmonized surface reflectance product". **Remote Sensing of Environment**, Vol. 235, 1-18.

118. SILVA, Inês (2020) – *Avaliação de metodologias de aprendizagem automática na classificação de culturas agrícolas com base em imagens do Sentinel-2*. **Dissertação de Mestrado**, Faculdade de Ciências da Universidade de Lisboa.

119. SCHULTZ, Michael; VOSS, Janek; AUER, Michael *et al.* (2017) – "Open land cover from OpenStreetMap and remote sensing". **Int J Appl Obs Geoinformation,** Vol. 63, 206-203.

120. TEILLET, Philippe; STÁENZ, Karl & WILLIAM, Daniel (1997) – "Effects of spectral, spatial, and radiometric characteristics on remote sensing vegetation indices of forested regions". **Remote Sensing of Environment**, Vol. 61, Issue 1, 139-149.

121. VHP (s.d.) – *InSAR – Satellite-based technique captures overall deformation "picture"*. **Disponível** no endereço URL: https://www.usgs.gov/programs/VHP/insar-satellite-based-technique-captures-overall-deformation-picture (acedido em 2 fevereiro, 2023).

122. VUOLO, Francesco; NEUWIRTH, Martin; IMMITZER, Markus; ATZBERGER, Clement & NG, Wai-Tim (2018). – "How much does multi-temporal Sentinel-2 data improve crop type classification?". **International Journal of Applied Earth Observation and Geoinformation**, Vol. 72, 122-130.

123. WALT, Stefan; COLBERT, Steven & VAROQUAUX, Gael (2011) – "The NumPy array: a structure for efficient numerical computation". **Computing in science & engineering**, Vol. 13, Issue 2, 22-30.

124. WANG, Qunming; BLACKBURN, George; ONOJEGHUO, Alex *et al.* (2017) – "Fusion of Landsat 8 OLI and Sentinel-2 MSI Data". **IEEE Transactions on Geoscience and Remote Sensing**, Vol. 55, Issue 7, 3885-3899.

125. WAŚNIEWSKI, Adam; HOŚCIŁO, Agata; ZAGJEWSKI, Bogdan & TARAZEWICS, Dieudonné (2020) – "Assessment of Sentinel-2 Satellite Images and Random Forest Classifier for Rainforest Mapping in Gabon". **Forests**, Vol.11, Issue 9, 1-17.

126. WIJEDASA, Lahiru; SLOAN, Sean; MICHELAKIS, Dimitrios & CLEMENTS, Gopalasamy (2012) – "Overcoming Limitations with Landsat Imagery for Mapping of Peat Swamp Forests in Sundaland". **Remote Sensing**, Vol. 4, Issue 9, 2595-2618.

127. WILLIAMSON, Andrew; BANWELL, Alison; WILLIS, Ian & ARNOLD, Neil (2018) – "Dual-satellite (Sentinel-2 and Landsat 8) remote sensing of supraglacial lakes in Greenland. **The Cryosphere**, Vol. 12, Issue 9, 3045-3065.

128. WOLF, Paul; DEWITT, Bon & WILKINSON, Benjamim (2014) – **Elements of Photogrammetry with Applications in GIS**. McGraw-Hill Education, 4ª Edição, New York.

129. XIANG, Shiming; NIE, Feiping & ZHANG, Changshui (2008) – "Learning a Mahalanobis distance metric for data clustering and classification". **Pattern Recognition**, Vol. 41, Issue 12, 3600-3612.

130. XU, Hanqiu (2005) – "Modification of normalised difference water index (NDWI) to enhance open water features in remotely sensed imagery". **International Journal of Remote Sensing**, Vol. 27, Issue 14, 3025-3033.

131. YANG, Xiucheng; QIN, Qiming; GRUSSENMEYER, Pierre & KOEHL, Mathieu (2018) – "Urban surface water body detection with suppressed built-up noise based on

water indices from Sentinel-2 MSI imagery". **Remote Sensing of Environment**, Vol. 219, 259-270.

132. ZAMBELLI, Pietro; GEBBERT, Sören & CIOLLI, Marco (2013) – "Pygrass: An Object Oriented Python Application Programming Interface (API) for Geographic Resources Analysis Support System (GRASS) Geographic Information System (GIS)". **ISPRS International Journal of Geo-Information**. Vol. 2, Issue 1, 201-219.

Legislação Consultada e documentos

1. Decreto-Lei n.º 130/2019 de 30 de agosto. Diário da República, 1.ª Série – N.º 166 – 30 de agosto de 2019. Lisboa. Portugal.

Anexos

- Anexo A – Nomenclaturas
    - Anexo A1 – Nomenclatura *CORINE Land Cover* (CLC)
    - Anexo A2 – Nomenclatura Carta de Ocupação do Solo (COS)
    - Anexo A3 – Nomenclatura *Urban Atlas* (UA)
    - Anexo A4 – Nomenclatura *United States Geological Survey* (USGS)
    - Anexo A5 – Nomenclatura *GlobeLand30*
    - Anexo A6 – Nomenclatura *Environmental System Research Institute* (Esri)
    - Anexo A7 – Nomenclatura *Dynamic World*

- Anexo B – Harmonização de nomenclaturas
    - Anexo B1 – Harmonização entre a nomenclatura COS 2018 e CLC+ *Backbone*

- Anexo C – Histogramas representativos da variação dos valores das classes relativamente ao índice radiométrico
    - Anexo C1 – *Forest*
    - Anexo C2 – *Herbaceous*
    - Anexo C3 – *Partly vegetated*
    - Anexo C4 – *Shrubland*
    - Anexo C5 – *Sealed Surface: Asphalt*
    - Anexo C6 – *Sealed Surface: Grey tile*
    - Anexo C7 – *Sealed Surface: Orange tile*
    - Anexo C8 – *Sealed Surface: White tile*
    - Anexo C9 – *Water*

- Anexo D – Repositório de código para o desenvolvimento metodológico
    - Anexo D1 – Programa 1: *Resampling Sentinel-2 images to 10 meter spatial resolution* (código existente)

- Anexo D2 – Programa 2: *Remove clouds from Sentinel-2 bands* (código existente)
- Anexo D3 – Programa 3: *Atmosferic and topografhic Landsat-8 images correction* (código programado no âmbito deste trabalho)
- Anexo D4 – Programa 4: *Resampling Landsat-8 images to a 10 meter spatial resolution* (código existente e adaptado no âmbito deste trabalho)
- Anexo D5 – Programa 5: *Projection and resampling* MDE *to a 10 meter spatial resolution* (código programado no âmbito deste trabalho)
- Anexo D6 – Programa 6: *Split area* (código programado no âmbito deste trabalho)
- Anexo D7 – Programa 7: *Make grid* (código programado no âmbito deste trabalho)
- Anexo D8 – Programa 8: *Intersection of grids with* OSM *classes* (código programado no âmbito deste trabalho)
- Anexo D9 – Programa 9: *Dissolve with* GDAL/OGR (código programado no âmbito deste trabalho)
- Anexo D10 – Programa 10: *Selection and extraction of attributes with 1 class and total occupied area* (código programado no âmbito deste trabalho)
- Anexo D11 – Programa 11: *Updating class legend values* (código programado no âmbito deste trabalho)
- Anexo D12 – Programa 12: *Vector grids to raster* (código programado no âmbito deste trabalho)
- Anexo D13 – Programa 13: *Aggregation of rasters* (código programado no âmbito deste trabalho)
- Anexo D14 – Programas 14, 15, 16, 17 e 18: *Example code for calculating the radiometric index* (código existente e adaptado no âmbito deste trabalho)
- Anexo D15 – Programa 19: *Final code for data filtering* (código programado no âmbito deste trabalho)
- Anexo D16 – Programa 20: *Random forest – Training model file* (código existente)
- Anexo D17 – Programa 21: *Execute classification* (código existente)

- Anexo E – Repositório de matrizes de confusão e tabelas de contingência

Anexo A – Nomenclaturas

Anexo A1 – Nomenclatura *CORINE Land Cover* (CLC)

Nível 1	Nível 2	Nível 3
1 Territórios artificializados	1.1 Tecido urbano	1.1.1 Tecido urbano contínuo
		1.1.2 Tecido urbano descontínuo
	1.2 Indústria, comércio e transportes	1.2.1 Indústria, comércio e equipamentos gerais
		1.2.2 Redes viárias e ferroviárias e espaços associados
		1.2.3 Áreas portuárias
		1.2.4 Aeroportos e aeródromos
	1.3 Áreas de extração de inertes, áreas de deposição de resíduos e estaleiros de construção	1.3.1 Áreas de extração de inertes
		1.3.2 Áreas de deposição de resíduos
		1.3.3 Áreas em construção
	1.4 Espaços verdes urbanos, equipamentos desportivos, culturais e de lazer, e zonas históricas	1.4.1 Espaços verdes urbanos
		1.4.2 Equipamentos desportivos, culturais e de lazer e zonas históricas
2 Áreas agrícolas e agro-florestais	2.1 Culturas temporárias	2.1.1 Culturas temporárias de sequeiro
		2.1.2 Culturas temporárias de regadio
		2.1.3 Arrozais
	2.2 Culturas permanentes	2.2.1 Vinhas
		2.2.2 Pomares
		2.2.3 Olivais
	2.3 Pastagens permanentes	2.3.1 Pastagens permanentes
	2.4 Áreas agrícolas heterogéneas	2.4.1 Culturas temporárias e/ ou pastagens associadas a culturas permanentes
		2.4.2 Sistemas culturais e parcelares complexos
		2.4.3 Agricultura com espaços naturais e semi-naturais
		2.4.4 Sistemas agro-florestais
3 Florestas e meios naturais e semi-naturais	3.1 Florestas	3.1.1 Florestas folhosas
		3.1.2 Florestas de resinosas
		3.1.3 Florestas mistas
	3.2 Florestas abertas, vegetação arbustiva e herbácea	3.2.1 Vegetação herbácea natural
		3.2.2 Matos
		3.2.3 Vegetação esclerofila
		3.2.4 Florestas aberta, cortes e novas plantações
	3.3 Zonas descobertas e com pouca vegetação	3.3.1 Praias, dunas e areias
		3.3.2 Rocha nua
		3.3.3 Vegetação esparsa
		3.3.4 Áreas ardidas
4 Zonas húmidas	4.1 Zonas húmidas interiores	4.1.1 Pauis
		4.1.2 Turfeiras
	4.2 Zonas húmidas litorais	4.2.1 Sapais
		4.2.2 Salinas e aquicultura litoral
		4.2.3 Zonas entre-marés
5 Corpos de água	5.1 Águas interiores	5.1.1 Cursos de água
		5.1.2 Planos de água
	5.2 Águas marinhas e costeiras	5.2.1 Lagoas costeiras

		5.2.2 Desembocaduras fluviais
		5.2.3 Oceano

Anexo A2 – Nomenclatura Carta de Ocupação do Solo (COS)

Nível 1	Nível 2	Nível 3	Nível 4
1.Territórios artificializados	1.1 Tecido edificado	1.1.1 Tecido edificado contínuo	1.1.1.1 Tecido edificado contínuo predominantemente vertical
			1.1.1.2 Tecido edificado contínuo predominantemente horizontal
		1.1.2 Tecido edificado descontínuo	1.1.2.1 Tecido edificado descontínuo
			1.1.2.2 Tecido edificado descontínuo esparso
		1.1.3 Espaços vazios em tecido edificado	1.1.3.1 Áreas de estacionamentos e logradouros
			1.1.3.2 Espaços vazios sem construção
	1.2 Indústria, comercio e instalações agrícolas	1.2.1 Indústria	1.2.1.1 Indústria
		1.2.2 Comércio	1.2.2.1 Comércio
		1.2.3 Instalações agrícolas	1.2.3.1 Instalações agrícolas
	1.3 Infraestruturas	1.3.1 Infraestruturas de produção de energia	1.3.1.1 Infraestruturas de produção de energia renovável
		1.3.2 Infraestruturas de águas e tratamento de resíduos	1.3.2.1 Infraestruturas para captação, tratamento e abastecimento de águas para consumo
	1.4 Transportes	1.4.1 Redes viárias e ferroviárias e espaços associados	1.4.1.1 Rede viária e espaços associados
			1.4.1.2 Rede ferroviária e espaços associados
		1.4.2 Áreas portuárias	1.4.2.1 Terminais portuários de mar e de rio
			1.4.2.2 Estaleiros navais e docas secas
			1.4.2.3 Marinas e docas pesca
		1.4.3 Aeroportos e aeródromos	1.4.3.1 Aeroportos
			1.4.3.2 Aeródromos
	1.5 Áreas de extração de inertes, áreas de deposição de resíduos e estaleiros de construção	1.5.1 Áreas de extração de inertes	1.5.1.1 Minas a céu aberto
			1.5.1.2 Pedreiras
		1.5.2 Áreas de deposição de resíduos	1.5.2.1 Aterros
			1.5.2.2 Lixeiras e Sucatas
		1.5.3 Áreas em construção	1.5.3.1 Áreas em construção
	1.6 Equipamentos	1.6.1 Equipamentos desportivos	1.6.1.1 Campos de golfe
			1.6.1.2 Instalações desportivas
		1.6.2 Equipamentos de lazer e parques de campismo	1.6.2.1 Parques de campismo
			1.6.2.2 Equipamentos de lazer
		1.6.3 Equipamentos culturais	1.6.3.1 Equipamentos culturais

		1.6.4 Cemitérios	1.6.4.1 Cemitérios
		1.6.5 Outros equipamentos e instalações turísticas	1.6.5.1 Outros equipamentos e instalações turísticas
	1.7 Parques e jardins	1.7.1 Parques e jardins	1.7.1.1 Parques e jardins
2.Agricultura	2.1 Culturas temporárias	2.1.1 Culturas temporárias de sequeiro e regadio e arrozais	2.1.1.1 Culturas temporárias de sequeiro e regadio
			2.1.1.2 Arrozais
	2.2 Culturas permanentes	2.2.1 Vinhas	2.2.1.1 Vinhas
		2.2.2 Pomares	2.2.2.1 Pomares
		2.2.3 Olivais	2.2.3.1 Olivais
	2.3 Áreas agrícolas heterogéneas.	2.3.1 Culturas temporárias e/ou pastagens melhoradas associadas a culturas permanentes	2.3.1.1 Culturas temporárias e/ou pastagens melhoradas associadas a vinha
			2.3.1.2 Culturas temporárias e/ou pastagens melhoradas associadas a pomar
			2.3.1.3 Culturas temporárias e/ou pastagens melhoradas associadas a olival
		2.3.2 Mosaicos culturais e parcelares complexos	2.3.2.1 Mosaicos culturais e parcelares complexos
		2.3.3 Agricultura com espaços naturais e seminaturais	2.3.3.1 Agricultura com espaços naturais e seminaturais
	2.4 Agricultura protegida e viveiros	2.4.1 Agricultura protegida e viveiros	2.4.1.1 Agricultura protegida e viveiros
3.Pastagens	3.1 Pastagens	3.1.1 Pastagens melhoradas	3.1.1.1 Pastagens melhoradas
		3.1.2 Pastagens espontâneas	3.1.2.1 Pastagens espontâneas
4.Superfícies agroflorestais (SAF)	4.1 Superfícies agroflorestais (SAF)	4.1.1 Superfícies agroflorestais (SAF)	4.1.1.1 SAF de sobreiro
			4.1.1.2 SAF de azinheira
			4.1.1.3 SAF de outros carvalhos
			4.1.1.4 SAF de pinheiro manso
			4.1.1.5 SAF de outras espécies
			4.1.1.6 SAF de sobreiro com azinheira
			4.1.1.7 SAF de outras misturas
5.Florestas	5.1 Florestas	5.1.1 Florestas de folhosas	5.1.1.1 Florestas de sobreiro
			5.1.1.2 Florestas de azinheira
			5.1.1.3 Florestas de outros carvalhos
			5.1.1.4 Florestas de castanheiro
			5.1.1.5 Florestas de eucalipto
			5.1.1.6 Florestas de espécies invasoras

			5.1.1.7 Florestas de outras folhosas
		5.1.2 Florestas de resinosas	5.1.2.1 Florestas de pinheiro-bravo
			5.1.2.2 Florestas de pinheiro manso
			5.1.2.3 Florestas de outras resinosas
6.Matos	6.1 Matos	6.1.1 Matos	6.1.1.1 Matos
7.Espaços descobertos ou com pouca vegetação	7.1 Espaços descobertos ou com pouca vegetação	7.1.1 Praias, dunas e areias	7.1.1.1 Praias, dunas e areais interiores
			7.1.1.2 Praias, dunas e areais costeiros
		7.1.2 Rocha nua	7.1.2.1 Rocha nua
		7.1.3 Vegetação esparsa	7.1.3.1 Vegetação esparsa
8.Zonas húmidas	8.1 Zonas húmidas	8.1.1 Zonas húmidas interiores	8.1.1.1 Pauis
		8.1.2 Zonas húmidas litorais	8.1.2.1 Sapais
			8.1.2.2 Zonas entremarés
9.Massas de água superficiais	9.1 Massas de água interiores	9.1.1 Cursos de água	9.1.1.1 Cursos de água naturais
			9.1.1.2 Cursos de água modificados ou artificializados
		9.1.2 Planos de água	9.1.2.1 Lagos e lagoas interiores artificiais
			9.1.2.2 Lagos e lagoas interiores naturais
			9.1.2.3 Albufeiras de barragens
			9.1.2.4 Albufeiras de represas ou de açudes
			9.1.2.5 Charcas
	9.2 Aquicultura	9.2.1 Aquicultura	9.2.1.1 Aquicultura
	9.3 Massas de água de transição e costeiras	9.3.1 Salinas	9.3.1.1 Salinas
		9.3.2 Lagoas costeiras	9.3.2.1 Lagoas costeiras
		9.3.3 Desembocaduras fluviais	9.3.3.1 Desembocaduras fluviais
		9.3.4 Oceano	9.3.4.1 Oceano

Anexo A3 – Nomenclatura *Urban Atlas* (*UA*)

Nível 1	Nível 2	Nível 3	Nível 4	2 Classes de áreas não classificadas
1.Territórios Artificializados	1.1 Fábricas Urbanas	1.1.1 Tecido industrial urbano contínuo (S.L. > 80%)		9.1 Sem dados (nuvens e sombras)
		1.1.2 Tecido industrial urbano descontínuo (S.L. 10%-80%)	1.1.2.1 Tecido industrial urbano denso descontínuo (S.L. 50%-80%)	9.2 Sem dados (ausência de imagens)
			1.1.2.2 Tecido industrial urbano de densidade descontínua média (S.L. 30%-50%)	
			1.1.2.3 Tecido industrial urbano de densidade descontínua baixa (S.L. 10%-30%)	
			1.1.2.4 Tecido industrial urbano de densidade descontínua muito baixa (S.L. < 10%)	
		1.1.3 Estruturas isoladas		
	1.2 Industrial, comercial, público, militar, privado e unidades de transportes	1.2.1 Industrial, comercial, público, militar e áreas privadas		
		1.2.2 Redes rodoviárias e ferroviárias e outras associadas	1.2.2.1 Estradas de circulação rápida e áreas associadas	
			1.2.2.2 Outras estradas e áreas associadas	
			1.2.2.3 Ferrovias e áreas associadas	
		1.2.3 Áreas portuárias		
		1.2.4 Aeroportos		
	1.3 Minas, lixeiras e zonas de obras	1.3.1 Extrações minerais e aterros locais		
		1.3.3 Zonas de construção e obras		
		1.3.4 Territórios sem utilização		
	1.4 Áreas artificias vegetadas não agrícolas	1.4.1 Áreas verdes urbanas		
		1.4.2 Instalações desportivas e de lazer		
2.Áreas Agrícolas	2.1 Terras aráveis (culturas anuais)			
	2.2 Culturas permanentes			
	2.3 Pastos			
	2.4 Cultivos complexos e mistos			
3. Áreas Naturais e Seminaturais	3.1 Florestas			
	3.2 Vegetação herbácea e associados			

	3.3 Espaços abertos com pouca ou nenhuma vegetação			
4. Zonas Húmidas				
5. Água				

Anexo A4 – Nomenclatura *United States Geological Survey* (*USGS*)

Nível 1	Nível 2	Nível 3 e Nível 4
1.Área Urbana ou Área Construída	11.Residencial	(...)
	12.Comercial e Serviços	
	13.Industrial	
	14.Tranportes, Comunicações e Serviços Públicos	
	15.Industrial e Comercial Complexos	
	16.Mistura de Área Urbana e Área Construída	
	17.Outras Áreas Urbanas ou Construções	
2.Área Agrícola	21.Terras de Cultivo e Pastagens	(...)
	22.Pomares, Bosques, Vinhas, Viveiros e Áreas de Hortas	
	23.Operações Confinadas de Alimentação	
	24.Outras Terras Agrícolas	
3.Pastagens	31.Pastagens Herbáceas	(...)
	32.Pastagens de Arbustos	
	33.Mistura de Pastagens	
4.Área Florestal	41.Florestas Botânicas	(...)
	42.Florestas de folhas Perenes	
	43.Mistura de Florestas	
5.Água	51.Linhas de Águas e Canais	(...)
	52.Lagos	
	53.Reservatórios	
	54.Baías e Estuários	
6.Pântanos	61.Pântano Florestal	(...)
	62.Pântano não Florestal	
7.Solo Descoberto	71.Salinas Secas	(...)
	72.Praias	
	73.Areais sem ser Praias	
	74.Rochas Expostas	
	75.Pedreiras e Poços de Cascalho	
	76.Áreas de Transição	
	77.Mistura de Terras Estéreis	
8.Tundra	81. Arbustos de Tundra	(...)
	82.Herbácea de Tundra	
	83.Solo Descoberto da Tundra	
	84.Tundra Húmida	
	85.Mistura de Tundras	
9.Neve ou Gelo	91.Campos de Neve	(...)
	92.Glaciares	

Anexo A5 – Nomenclatura *GlobeLand30*

Nível 1	Descrição
10.Terra Cultivada	Refere-se às terras utilizadas para o cultivo de culturas. Arrozais, terras altas irrigadas pela chuva, campos de vegetais, pastagens cultivadas, terras cultivadas de árvores de fruto ou outras, campos de chá, campos de café e outras terras agrícolas para fins económicos
20.Floresta	Refere-se às terras cobertas por árvores, cuja densidade é superior a 30%. Florestas de folha caduca, florestas folhosas de folhas perenes, florestas coníferas de folha caduca, florestas coníferas de folha perene, floresta mista e floresta esparsa cuja densidade cobre entre a 10% e 30%
30.Pastagens	Refere-se a terras cobertas por erva natural com densidade de cobertura superior a 10%. Os prados, pastos alpinos, etc. estão incluídos nesta categoria
40.Arbustos	Refere-se às terras cobertas por arbustos e com uma densidade de cobertura superior a 30%. Arbustos de montanhas, arbustos com floração no verão e de folhas perenes e selvas em áreas de deserto com uma densidade de cobertura superior a 10% estão incluídos nesta classe
50.Pântanos	Refere-se às terras de junção entre terra e áreas com água, constantemente cobertas por águas rasas ou solos em constante humidade. Pântanos interiores, lagos de pântanos, pântanos originados por rios, pântanos de florestas e arbustos, pântanos originados pelo mar etc.
60.Corpos de Água	Refere-se a regiões cobertas por água líquida na superfície terrestre. Rio, lago, lagoas, reservatórios, etc.
70.Tundra	Refere-se às terras cobertas por líquen, musgo, erva perene e arbustos resistentes a regiões frias montanhosas. Tundra de arbustos, tundra de erva, tundra húmida, tundra alpina e tundra árida, etc.
80.Superfícies Artificiais	Refere-se às superfícies formadas por edifícios construídos pelo Homem. Todos os tipos de habitação em áreas urbanas e rurais, áreas industriais e de minas, meios de transporte, etc.
90.Terra exposta	Refere-se a coberturas naturais com densidade de cobertura inferior a 10%. Desertos, areias, cascalhos, rochas expostas, salinas, etc.
100.Neve e gelo permanentes	Refere-se às terras cobertas por neve permanente, glaciares e calotes polares. Neve permanente e gelos permanentes em áreas montanhosas e glaciares nas calotes polares, etc.

Anexo A6 – Nomenclatura *Environmental System Research Institute* (*Esri*)

Nível 1	Descrição
1. Água	Áreas onde água se encontra presente ao longo do ano; não considera áreas com água esporádica; contém pouca ou nenhuma vegetação esparsa, nenhum afloramento rochoso nem elementos de áreas construídas como docas; exemplos: rios, lagoas, lagos, oceanos
2. Árvores	Aglomerados de vegetação densa, tipicamente com copa fechada e densa; exemplo: pântanos, vegetação arbórea, conjuntos de vegetação densa e alta
3. Vegetação ripícola	Áreas de vegetação com evidência de água durante maior parte do ano; áreas inundadas sazonalmente; exemplos: culturas de arroz, agricultares irrigadas e inundadas
4. Culturas	Cereais e plantas cultivadas pelo Homem sem considerar árvores; exemplos: milho, trigo, soja e terras em pousio
5. Área Construída	Estruturas construídas pelo Homem, grandes vias de circulação e ferroviárias; amplas superficíes impermeáveis e homogéneas, incluindo estruturas de parques de estacionamento, edifícios de escritórios e habitações; exemplos: casas, vilas, aglomerados citadinos, estadas, asfalto
6. Solo descoberto	Áreas de rochas ou solos com pouca ou nenhuma vegetação durante o ano inteiro; vastas áreas de areia e desertos sem qualquer presença de vegetação; exemplos: rochas ou solo descoberto, desertos e dunas de areia, salinas/ redes hidrográficas secas e minas

7. Neve/Gelo	Áreas homogéneas com neve e gelo durante todo o ano, normalmente em áreas montanhosas e em latitudes mais altas; exemplos: glaciares e áreas com neve
8. Nuvens	Nenhuma informação acerca do uso e ocupação do solo devido à presença de nuvens
9. Pastagens	Áreas compostas por erva homogénea sem sinais de vegetação alta; exemplos: prados e campos naturais com pouca ou nenhuma cobertura de árvores, savana sem presença de árvores, campos de golfe, relvas e pastos

Anexo A7 – Nomenclatura *Dynamic World*

Nível 1	Descrição
0.Água	Presença de água na imagem; baixa presença de vegetação esparsa, nenhum afloramento rochoso e não são considerados elementos construídos como docas; não inclui terrenos que podem vir a ser cobertos por água ou já foram cobertos por água; exemplos: Rios, Lagoas e Lagos, Oceano e Salinas inundadas
1.Árvores	Conjuntos significativos de vegetação densa, geralmente com copas fechadas e densas; mais alta e mais escura do que a vegetação ao redor; exemplos: Vegetação arbórea, Arbustos verdes e densos, Aglomerados de vegetação densa e alta em savanas, Pomares de maçãs, peras, etc., Pântanos, Áreas queimadas das classes anteriores referidas
2.Relva	Amplas áreas cobertas por vegetação rasteira com pouco ou nenhum sinal de vegetação alta; cereais silvestres e ervas desenvolvidas sem intervenção do Homem; exemplos: Pastos e campos naturais com pouca cobertura arbórea, Savana, Campos de golf e campos de futebol, Cortes de árvores para linhas de energia, gás, etc., Campos de canas e pântanos sem inundações
3.Vegetação ripícola	Áreas de qualquer tipo de vegetação com evidência de água; áreas inundadas sazonalmente; exemplos: Culturas de arroz
4.Plantações	Cereais e culturas plantadas pelo Homem; exemplo: Milho, trigo, soja, etc., Feno e terras em pousio
5.Arbustos	Pequenos aglomerados de plantas ou plantas individuais dispersas em áreas de solo descoberto e de rocha exposta; grupos de arbustos em florestas densas que se distinguem por não serem tão altas como árvores com tonalidades acastanhadas devido à cobertura de folhas ser menos densa; exemplos: Cobertura esparsa e aglomerada de arbustos e tufos de erva, Savanas com árvores ou plantas muito esparsas
6.Área Construída	Grupos de estruturas construídas pelo Homem ou estruturas individuais muito altas; edifícios industriais, comerciais e privados e parques de estacionamentos associados; conjuntos de edifícios residenciais, ruas, árvores, construções residenciais, jardins e vegetação que circundam edifícios; principais redes de circulação e redes ferroviárias afastadas das zonas residenciais; grandes superfícies impermeáveis, incluindo parques de estacionamento, grandes prédios residenciais e de escritórios; exemplos: Aglomerados de casas, incluindo pequenos jardins com árvores de pequeno porte, Aldeias, vilas e cidades com aglomerado de edifícios compacto, Conjuntos de estradas pavimentadas e grandes vias de circulação, Asfalto e outras superfícies produzidas pelo Homem
7.Superficíe exposta	Áreas rochosas ou solos com pouca vegetação; vastas áreas de areia e desertos com pouca ou nenhuma vegetação; grandes redes de circulação ou estradas de terra; exemplos: Rochas expostas, Solo descoberto, Salinas secas, Lagos secos, Minas, Grandes lotes vazios em áreas urbanas
8.Neve e Gelo	Grandes áreas homogéneas de neve ou gelo, presentes em montanhas ou em latitudes mais altas; grandes áreas homogéneas com queda regular de neve; exemplos: Glaciares, Blocos de neve permanentes, Quedas de neve

Anexo B – Harmonização de nomenclaturas

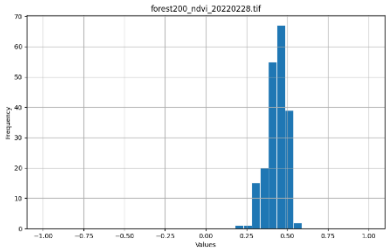
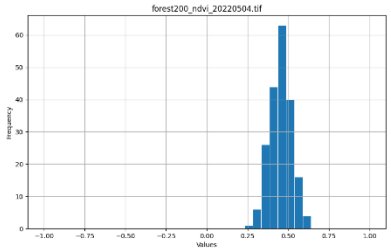
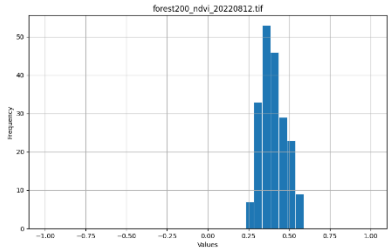
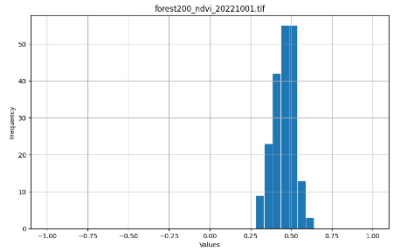
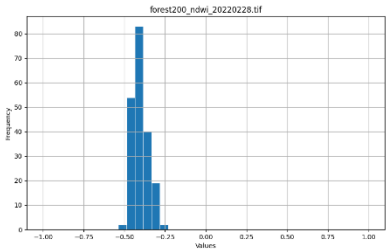
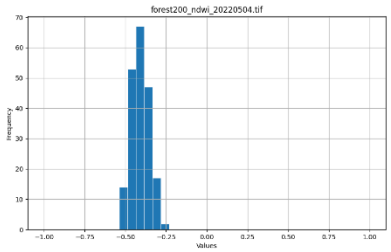
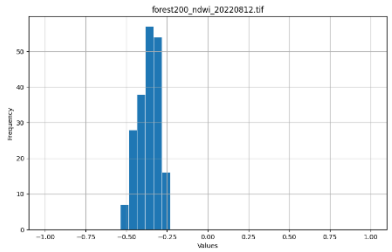
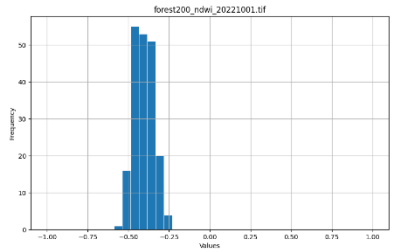
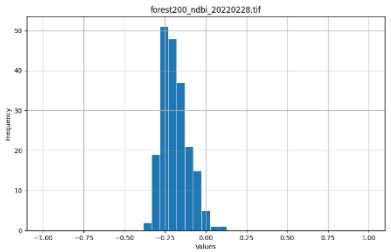
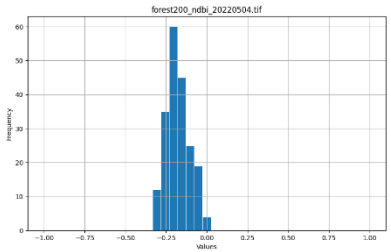
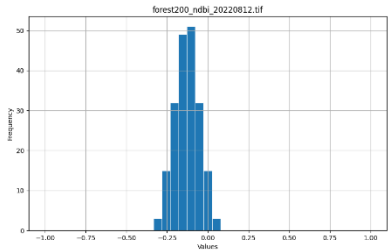
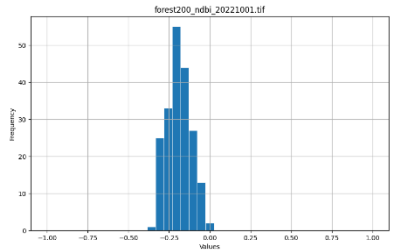
Anexo B1 – Harmonização entre a nomenclatura COS 2018 e CLC+ *Backbone*

Harmonização da Nomenclatura COS 2018 e CLC+ *Backbone*	
1.1.1.1 Tecido edificado contínuo predominantemente vertical 1.1.1.2 Tecido edificado contínuo predominantemente horizontal 1.1.2.1 Tecido edificado descontínuo 1.1.2.2 Tecido edificado descontínuo esparso 1.1.3.1 Áreas de estacionamentos e logradouros 1.2.1.1 Indústria 1.2.2.1 Comércio 1.2.3.1 Instalações agrícolas 1.3.1.1 Infraestruturas de produção de energia renovável 1.3.1.2 Infraestruturas de produção de energia não renovável 1.3.2.1 Infraestruturas para captação, tratamento e abastecimento de águas para consumo 1.3.2.2 Infraestruturas de tratamento de resíduos e águas residuais 1.4.1.1 Rede viária e espaços associados 1.4.1.2 Rede ferroviária e espaços associados 1.4.2.1 Terminais portuários de mar e de rio 1.4.2.2 Estaleiros navais e docas secas 1.4.2.3 Marinas e docas pesca 1.4.3.1 Aeroportos 1.4.3.2 Aeródromos 1.6.1.2 Instalações desportivas 1.6.3.1 Equipamentos culturais 1.6.5.1 Outros equipamentos e instalações turísticas 2.4.1.1 Agricultura protegida e viveiros	1 Espaços fechados/ selados (*Sealed Surface*)
2.2.2.1 Pomares 2.2.3.1 Olivais 2.3.1.2 Culturas temporárias e/ou pastagens melhoradas associadas a pomar 2.3.1.3 Culturas temporárias e/ou pastagens melhoradas associadas a olival 4.1.1.1 SAF de sobreiro 4.1.1.2 SAF de azinheira 4.1.1.3 SAF de outros carvalhos 4.1.1.4 SAF de pinheiro manso 4.1.1.5 SAF de outras espécies 4.1.1.6 SAF de sobreiro com azinheira 4.1.1.7 SAF de outras misturas 5.1.1.1 Florestas de sobreiro 5.1.1.2 Florestas de azinheira 5.1.1.3 Florestas de outros carvalhos 5.1.1.4 Florestas de castanheiro 5.1.1.5 Florestas de eucalipto 5.1.1.6 Florestas de espécies invasoras 5.1.1.7 Florestas de outras folhosas 5.1.2.1 Florestas de pinheiro-bravo 5.1.2.2 Florestas de pinheiro manso 5.1.2.3 Florestas de outras resinosas 2.3.2.1 Mosaicos culturais e parcelares complexos	2 Floresta (*Forest*)
1.6.2.2 Equipamentos de lazer 1.7.1.1 Parques e jardins	3 Arbustos (*Shrubland*)

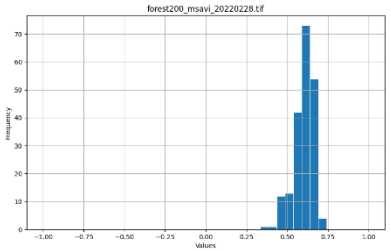
2.2.1.1 Vinhas 2.3.1.1 Culturas temporárias e/ou pastagens melhoradas associadas a vinha 2.3.3.1 Agricultura com espaços naturais e seminaturais 6.1.1.1 Matos	
1.6.1.1 Campos de golfe 2.1.1.1 Culturas temporárias de sequeiro e regadio 3.1.1.1 Pastagens melhoradas 3.1.2.1 Pastagens espontâneas	4 Herbáceas (*Herbaceous*)
2.1.1.2 Arrozais 8.1.1.1 Pauis 8.1.2.1 Sapais 8.1.2.2 Zonas entremarés	5 Áreas húmidas (*Wetlands*)
1.1.3.2 Espaços vazios sem construção 1.5.1.1 Minas a céu aberto 1.5.1.2 Pedreiras 1.5.2.1 Aterros 1.5.2.2 Lixeiras e Sucatas 1.5.3.1 Áreas em construção 1.6.2.1 Parques de campismo 1.6.4.1 Cemitérios 7.1.1.1 Praias, dunas e areais interiores 7.1.1.2 Praias, dunas e areais costeiros 7.1.2.1 Rocha nua 7.1.3.1 Vegetação esparsa	6 Áreas de pouca vegetação ou solo descoberto (*Partly vegetated or Non-vegetated land*)
9.1.1.1 Cursos de água naturais 9.1.1.2 Cursos de água modificados ou artificializados 9.1.2.1 Lagos e lagoas interiores artificiais 9.1.2.2 Lagos e lagoas interiores naturais 9.1.2.3 Albufeiras de barragens 9.1.2.4 Albufeiras de represas ou de açudes 9.1.2.5 Charcas 9.2.1.1 Aquicultura 9.3.1.1 Salinas 9.3.2.1 Lagoas costeiras 9.3.3.1 Desembocaduras fluviais 9.3.4.1 Oceano	7 Corpos de água (*Water*)

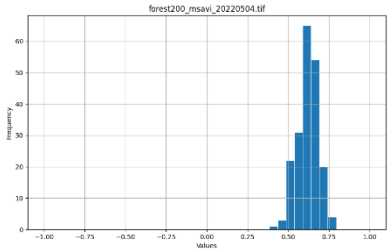
Anexo C – Histogramas representativos da variação dos valores das classes relativamente ao índice radiométrico

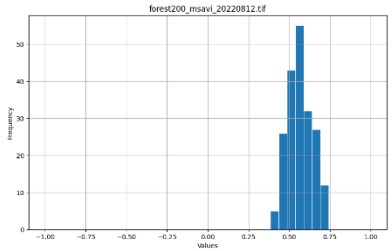
Anexo C1 – *Forest*

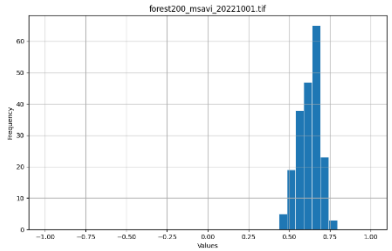
	20220228	20220504	20220812	20221001
NDVI				
NDWI				
NDBI				

MSAVI

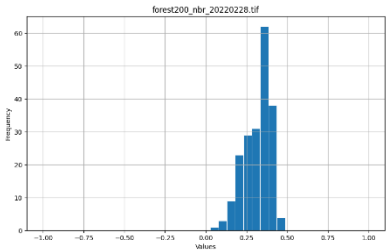


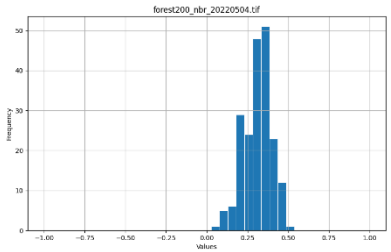


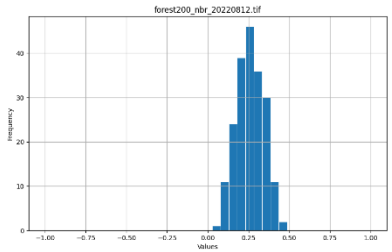


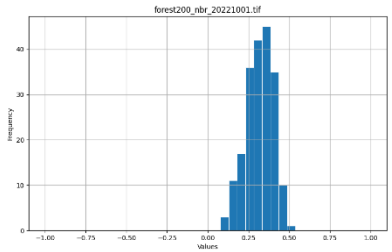


NBR





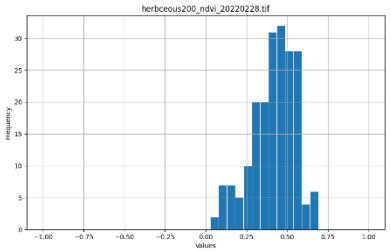


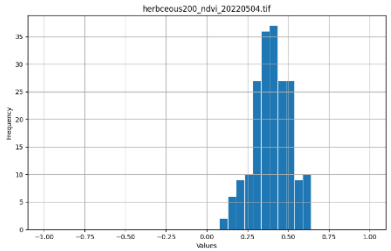


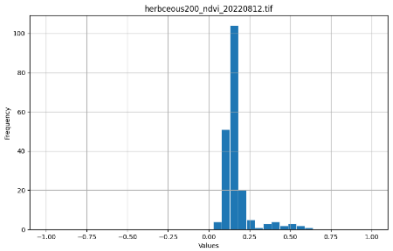
Anexo C2 – *Herbaceous*

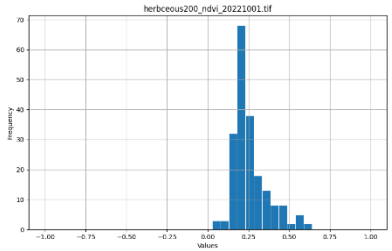
20220228 20220504 20220812 20221001

NDVI

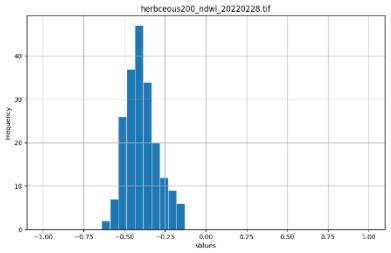


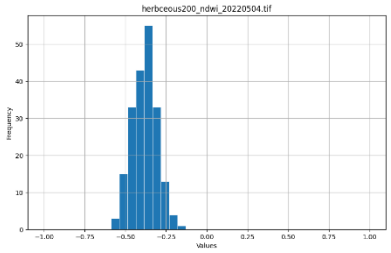


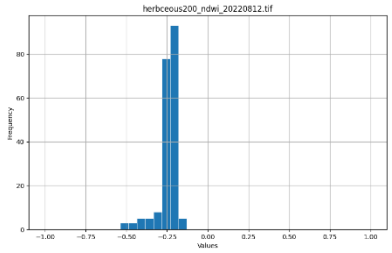


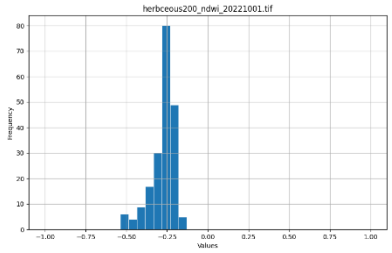


NDWI

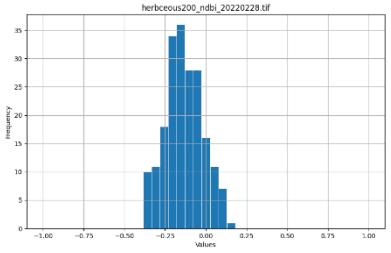


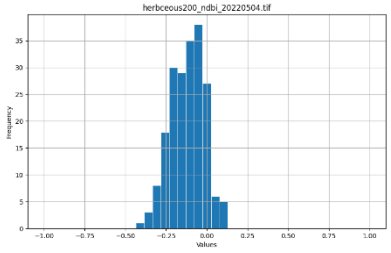


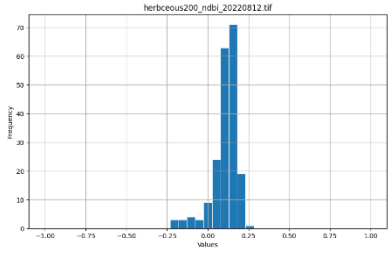


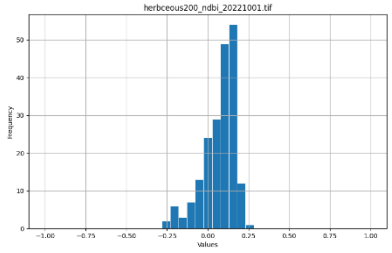


NDBI

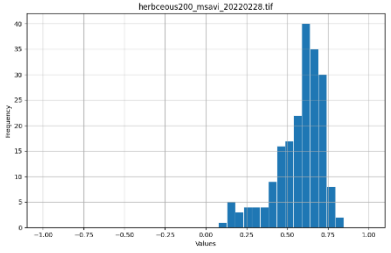


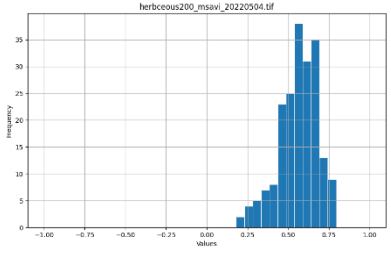


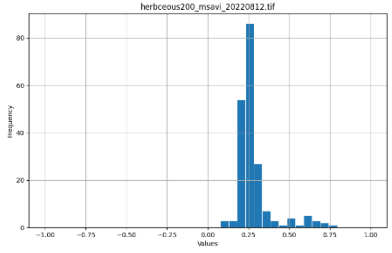


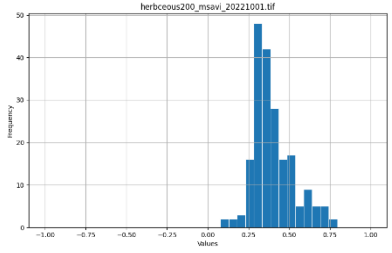


MSAVI

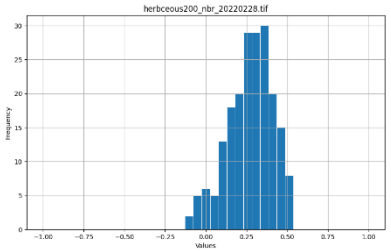


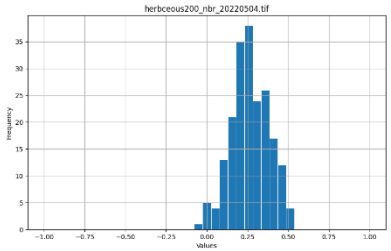


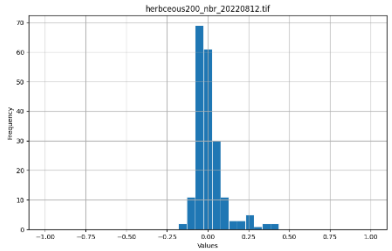


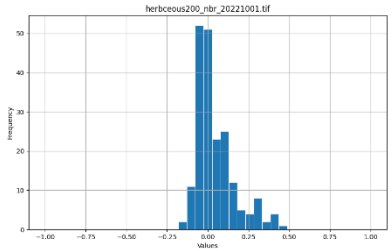


NBR





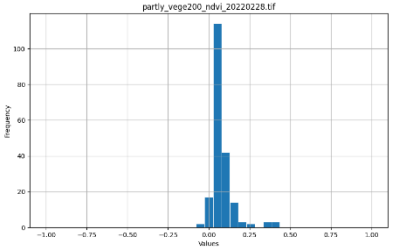


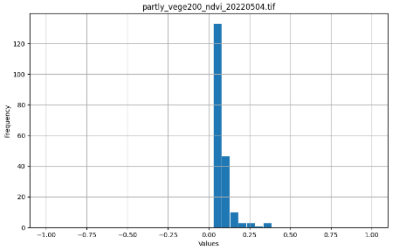


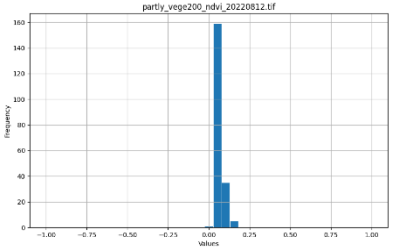
Anexo C3 – *Partly vegetated or Non-vegetated land*

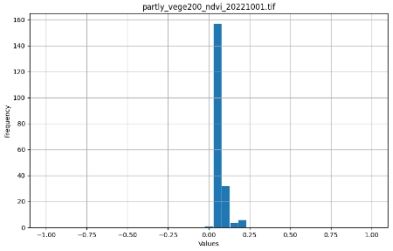
20220228 20220504 20220812 20221001

NDVI

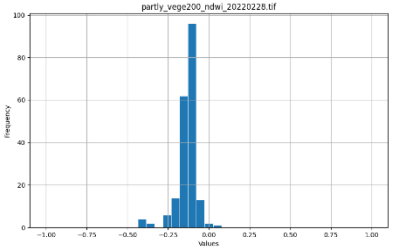


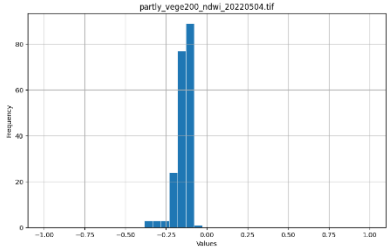


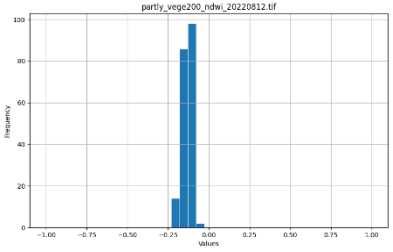


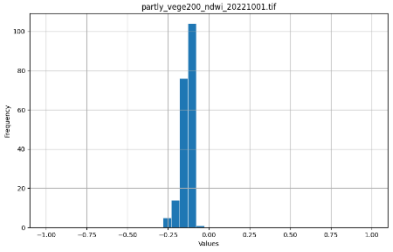


NDWI

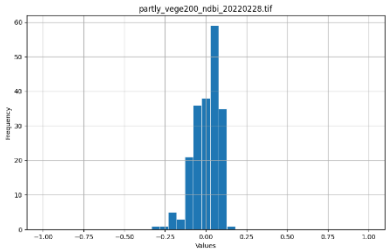


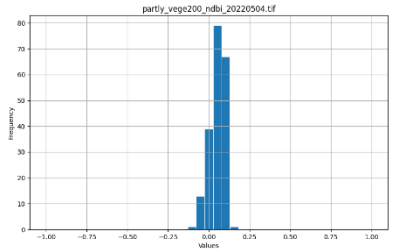


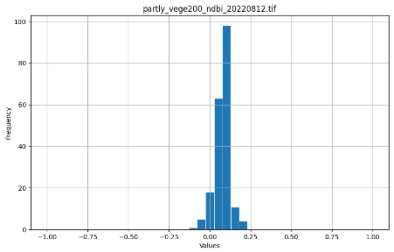


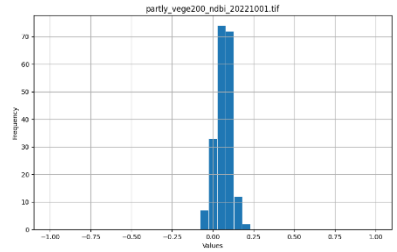


NDBI

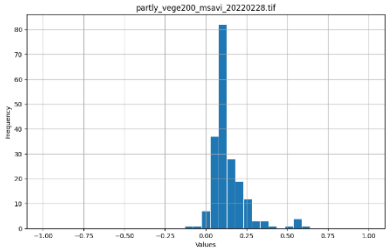


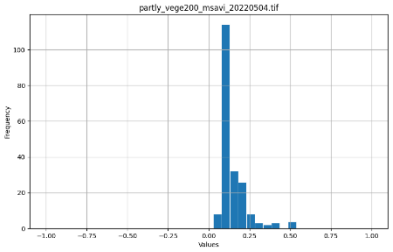


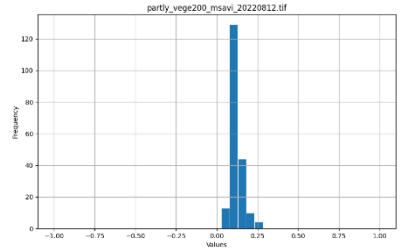


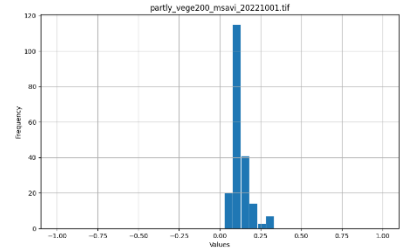


MSAVI

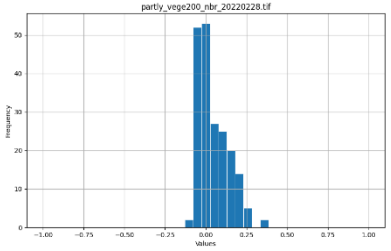


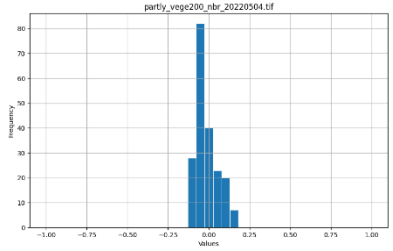


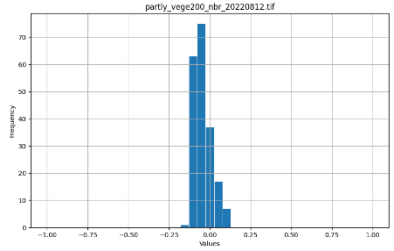


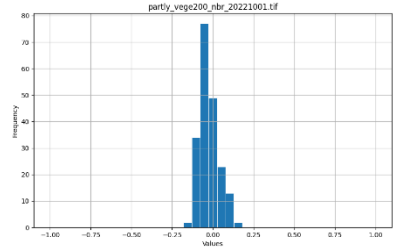


NBR

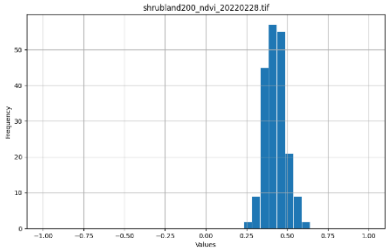
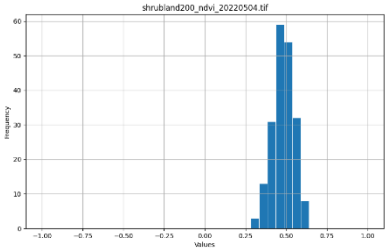
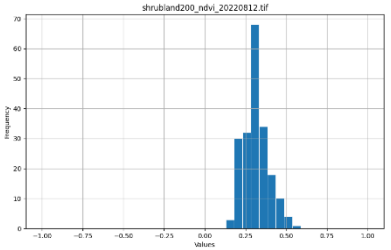
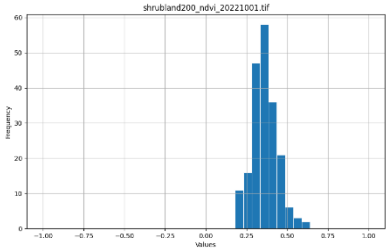
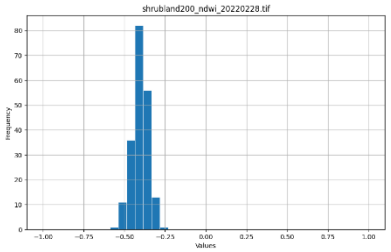
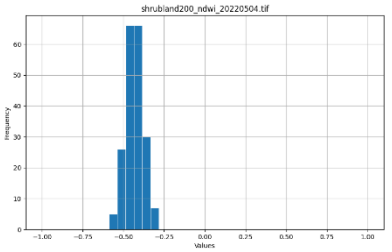
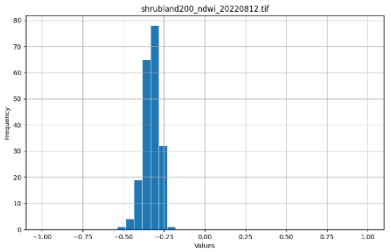
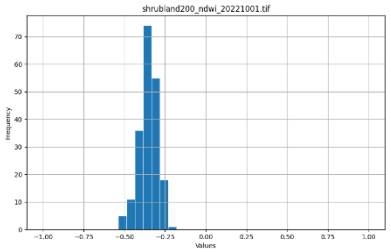
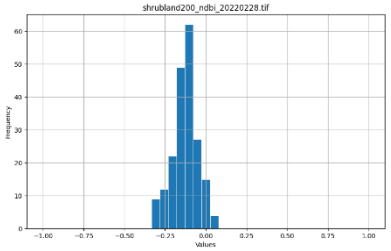
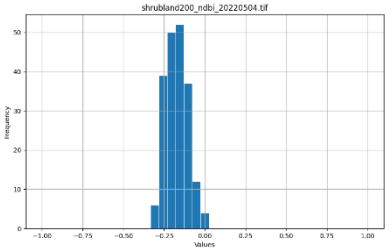
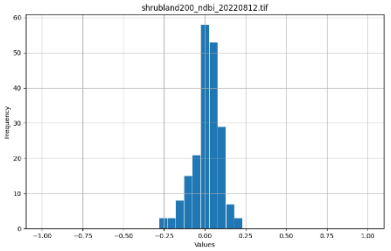
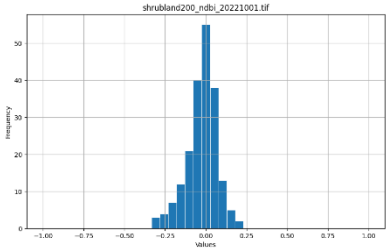




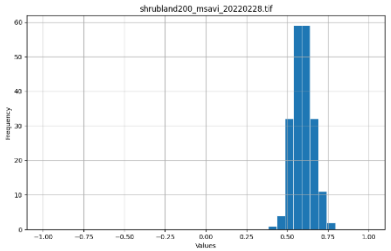


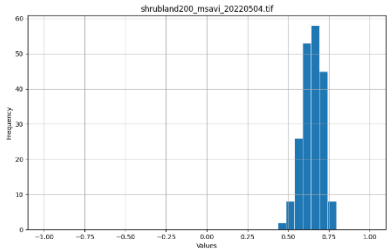


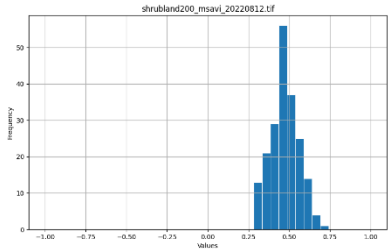
Anexo C4 – *Shrubland*

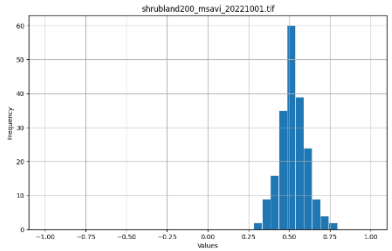
	20220228	20220504	20220812	20221001
NDVI				
NDWI				
NDBI				

MSAVI

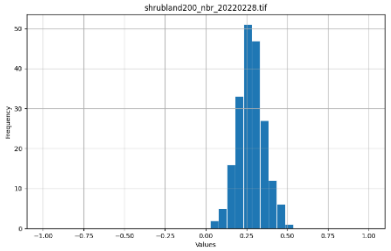


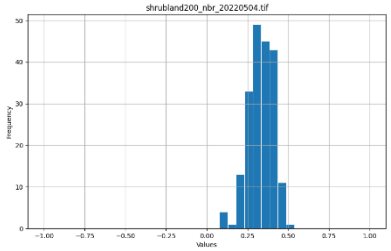


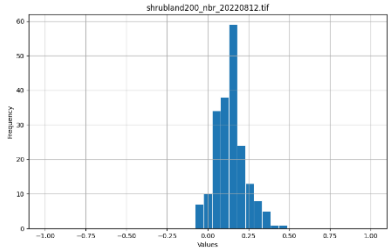


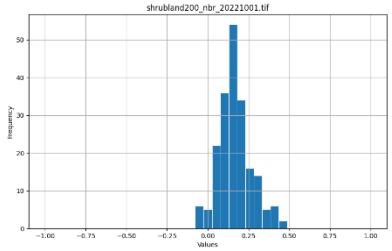


NBR





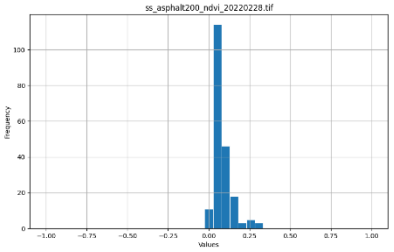


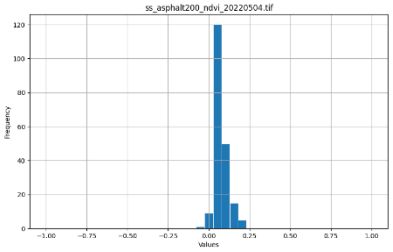


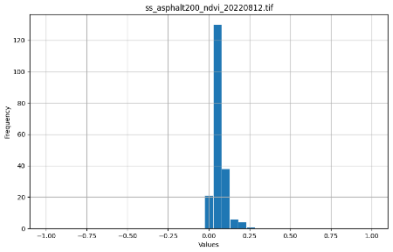
Anexo C5 – *Sealed Surface: Asphalt*

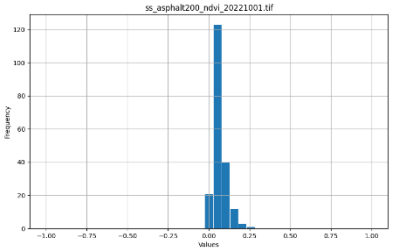
20220228 20220504 20220812 20221001

NDVI

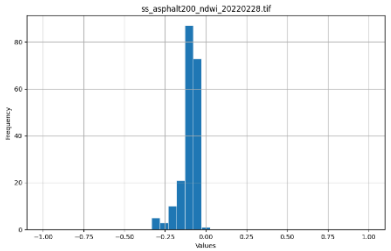


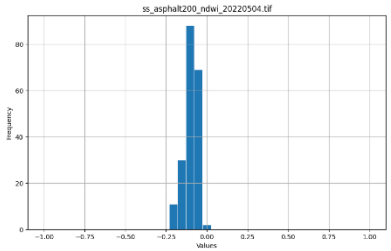


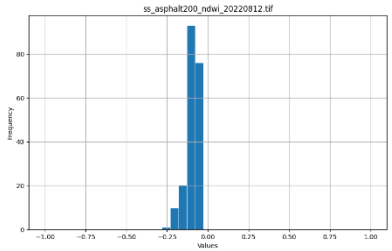


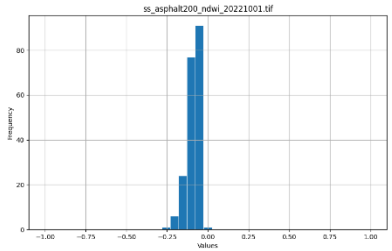


NDWI

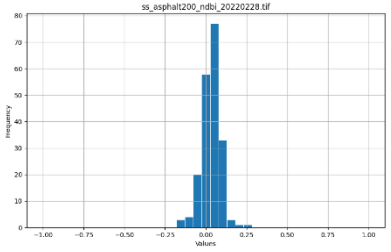


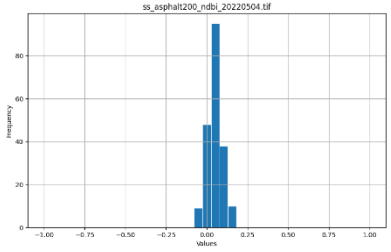


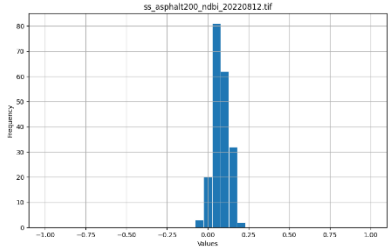


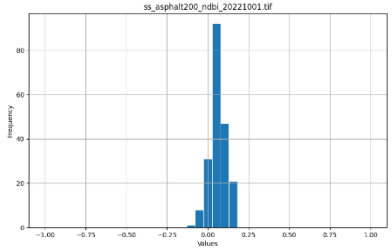


NDBI

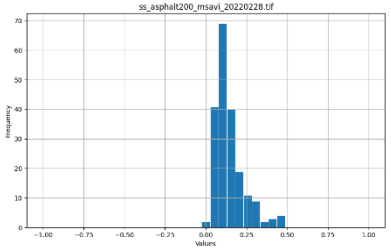


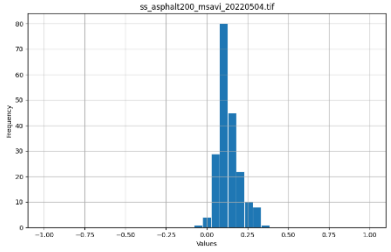


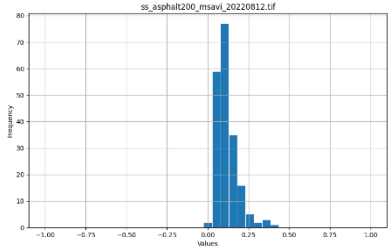


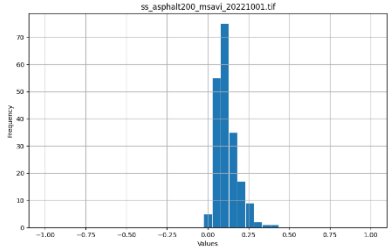


MSAVI

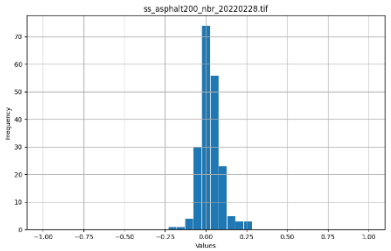


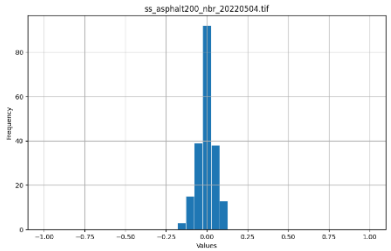


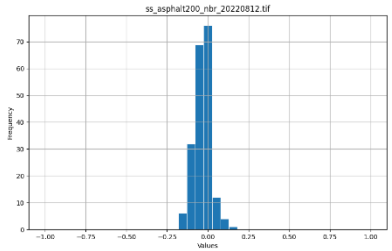


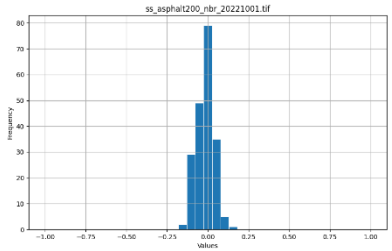


NBR

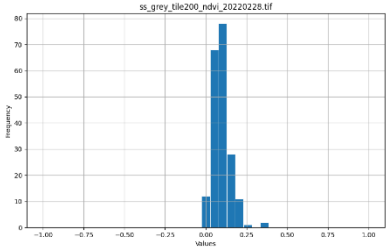
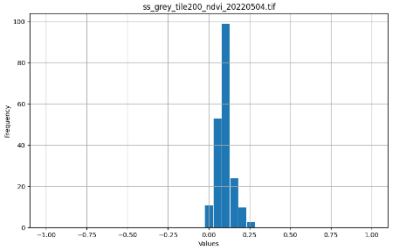
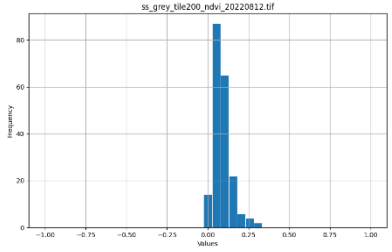
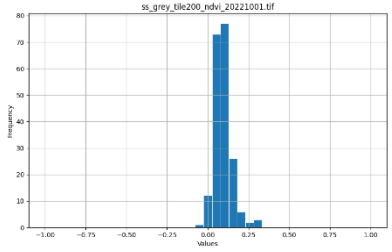
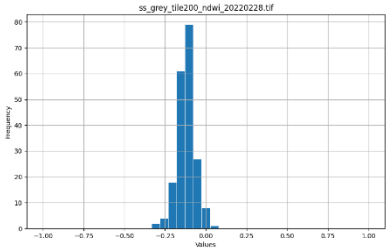
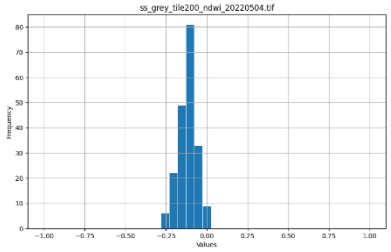
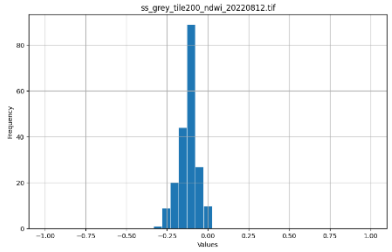
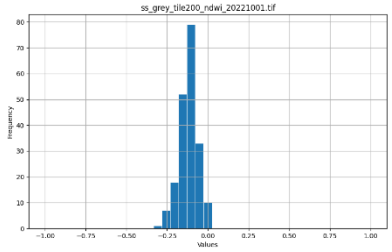




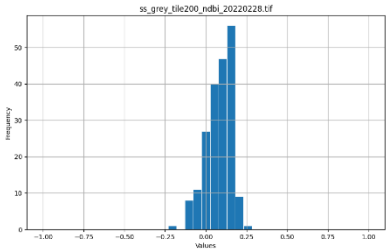


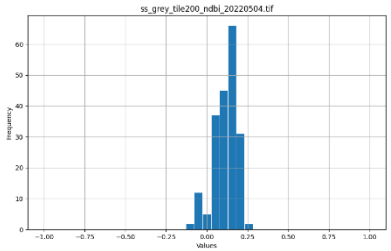


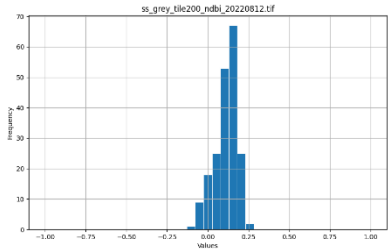
Anexo C6 – *Sealed Surface: Grey tile*

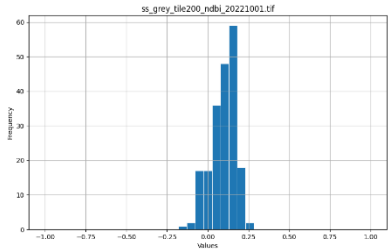
	20220228	20220504	20220812	20221001
NDVI				
NDWI				

NDBI

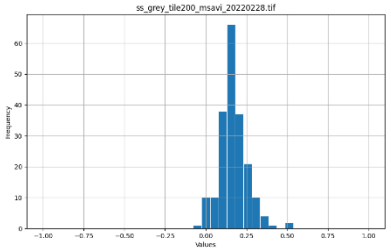


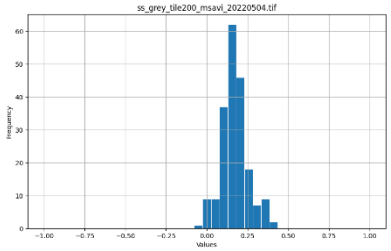


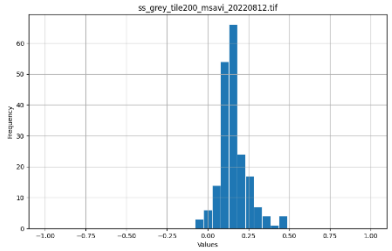


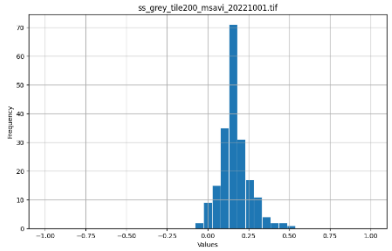


MSAVI

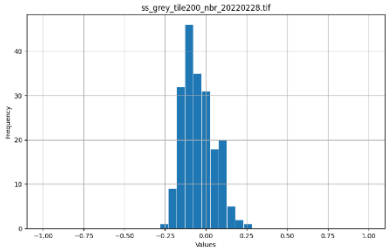


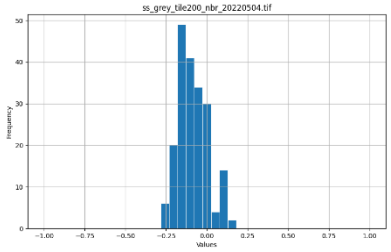


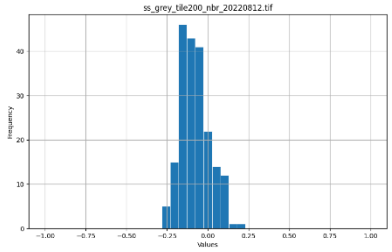


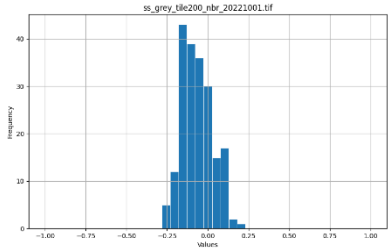


NBR

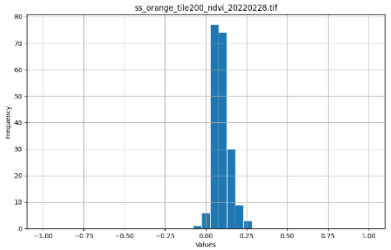
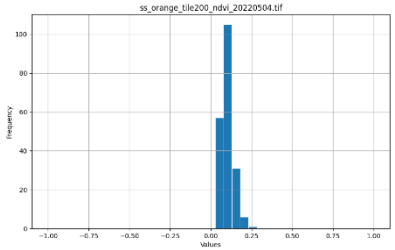
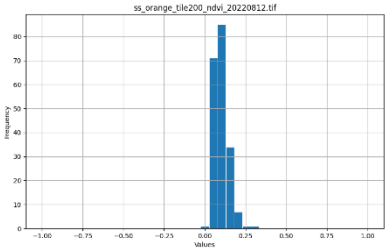
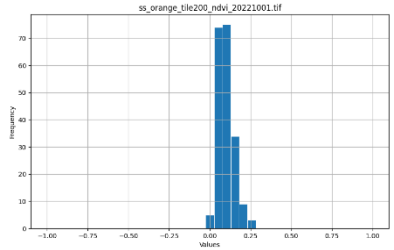
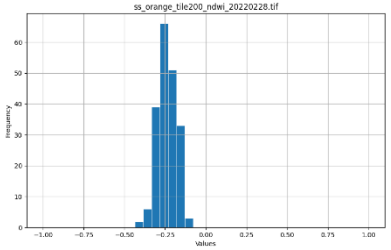
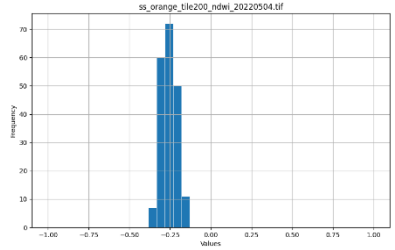
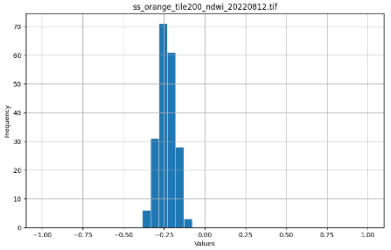
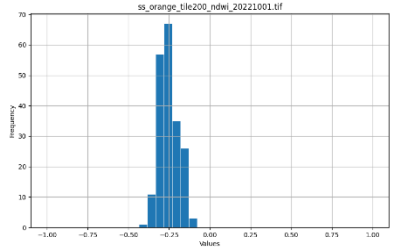
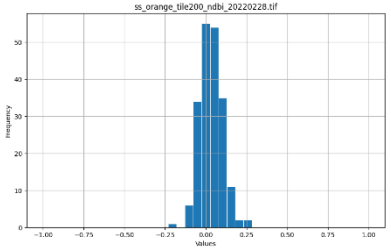
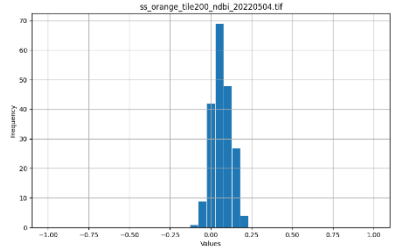
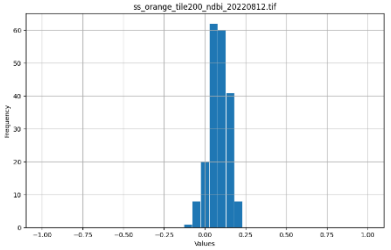
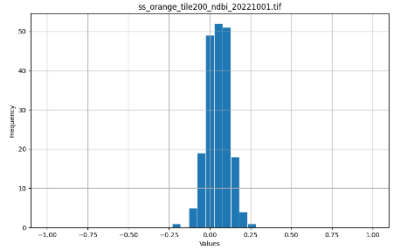




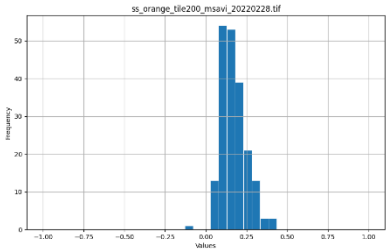


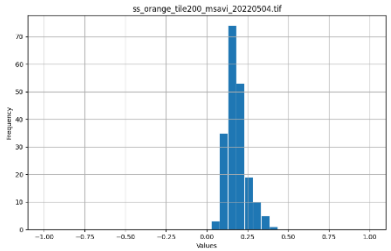


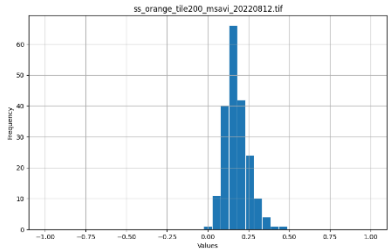
Anexo C7 – *Sealed Surface: Orange tile*

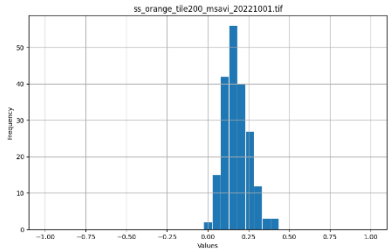
	20220228	20220504	20220812	20221001
NDVI				
NDWI				
NDBI				

MSAVI

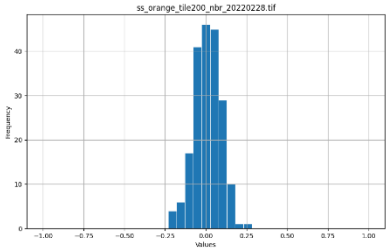


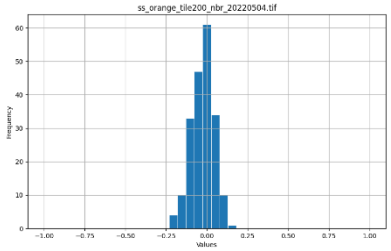


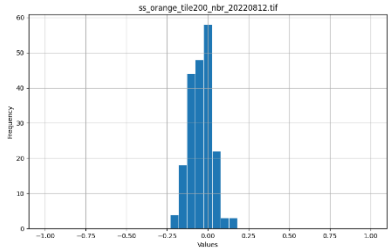


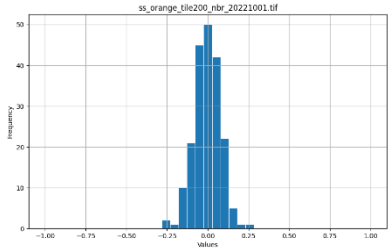


NBR





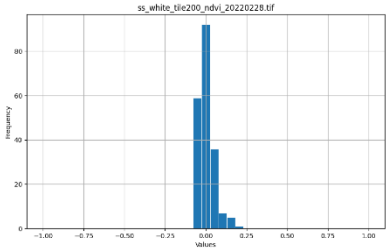


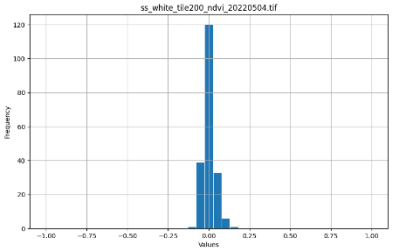


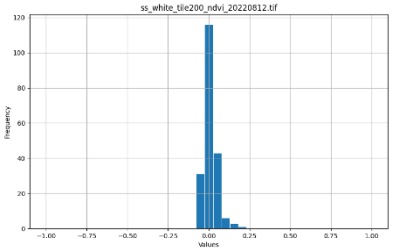
Anexo C8 – *Sealed Surface: White tile*

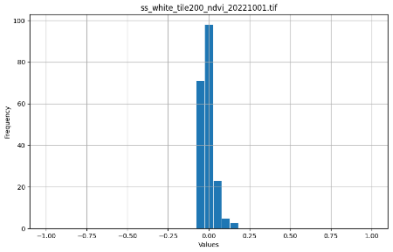
20220228 20220504 20220812 20221001

NDVI

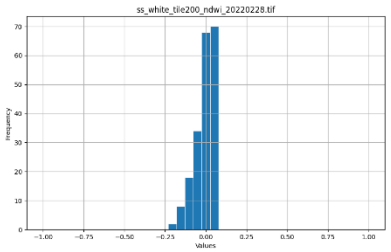


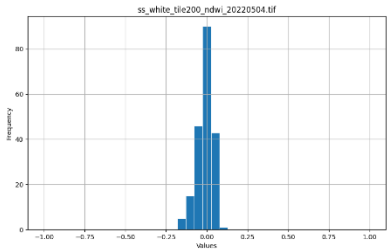


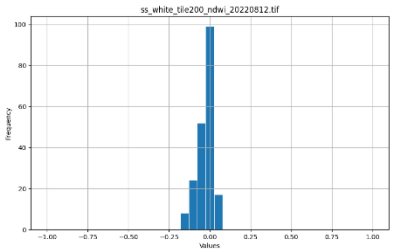


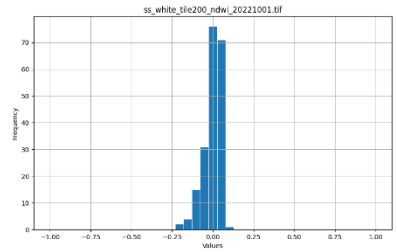


NDWI

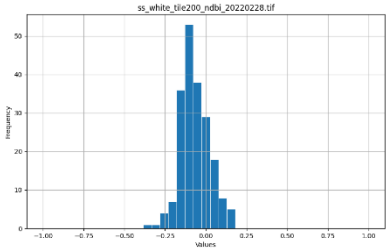


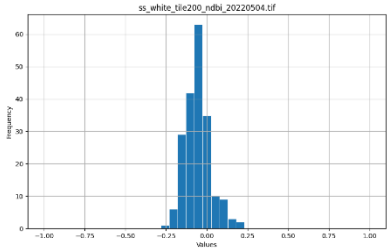


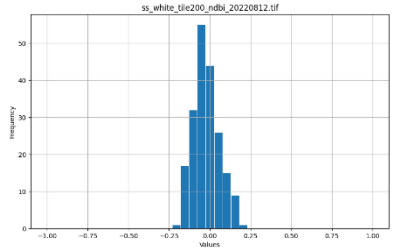


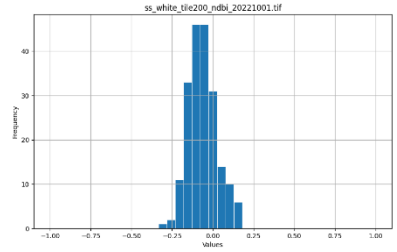


NDBI

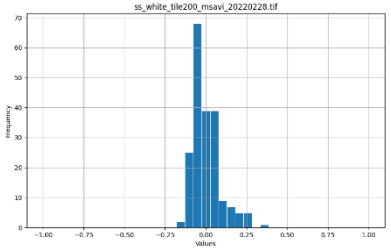


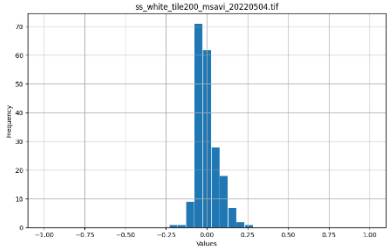


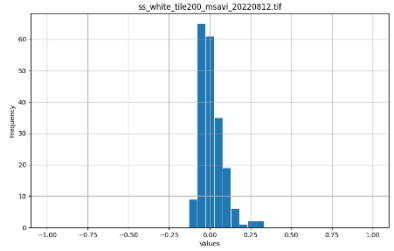


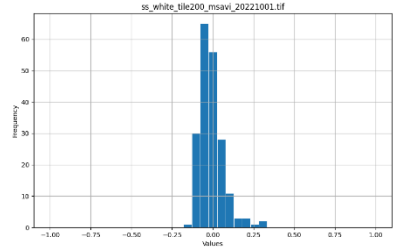


MSAVI

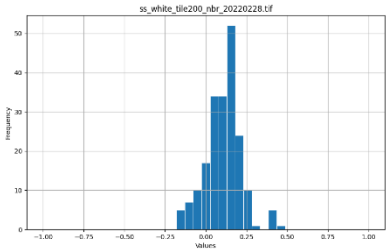


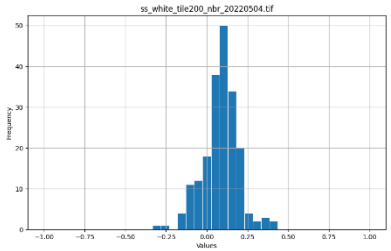


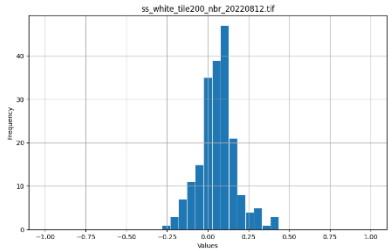


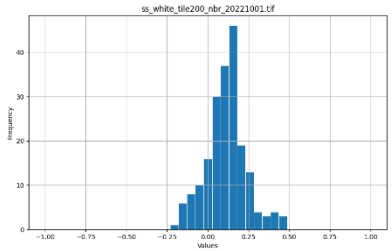


NBR

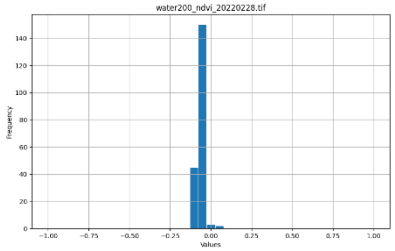
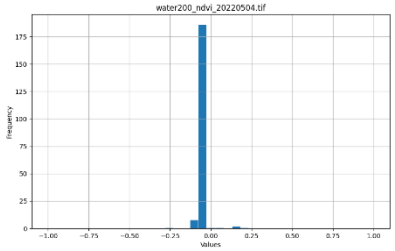
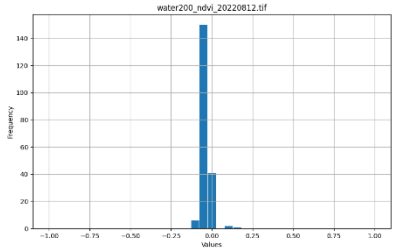
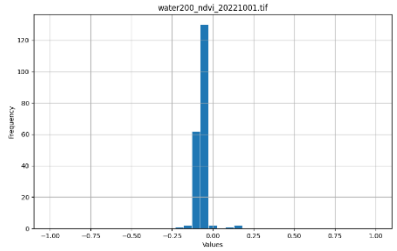
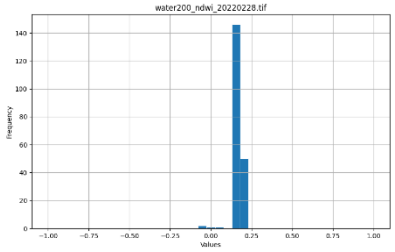
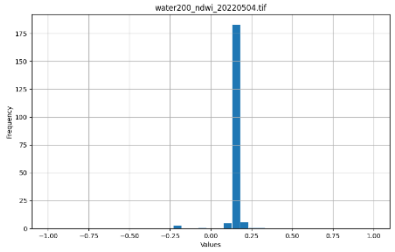
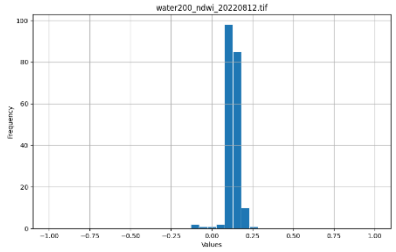
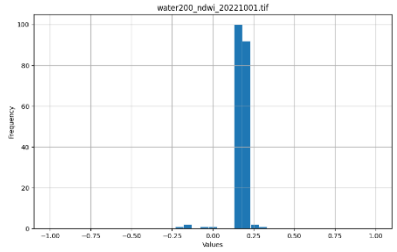




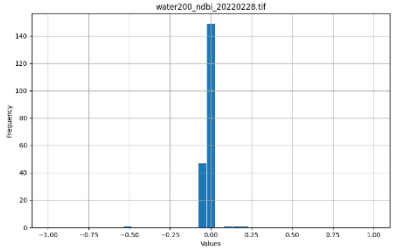


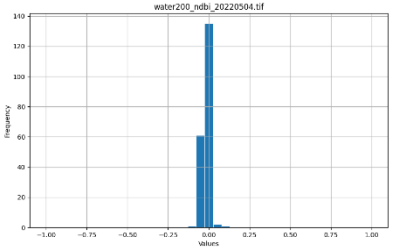


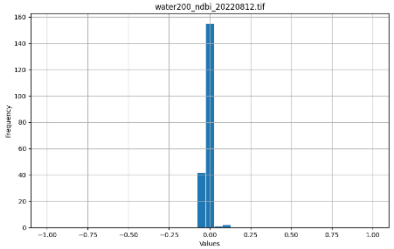
Anexo C9 – *Water*

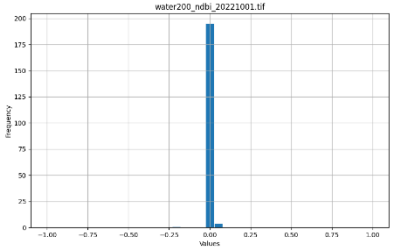
	20220228	20220504	20220812	20221001
NDVI				
NDWI				

NDBI

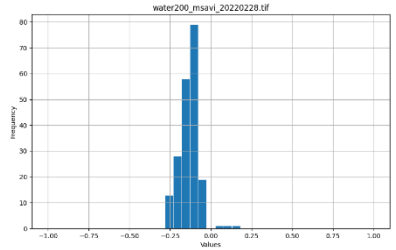


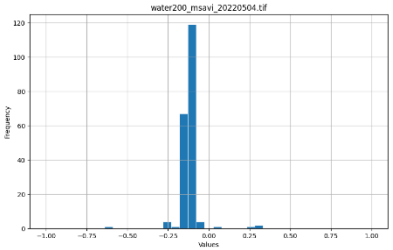


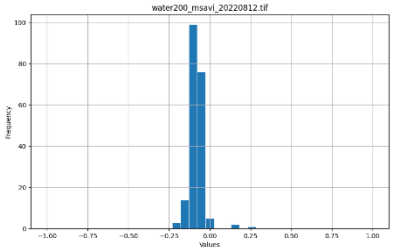


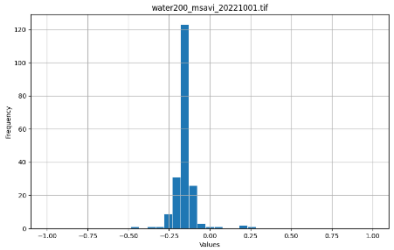


MSAVI

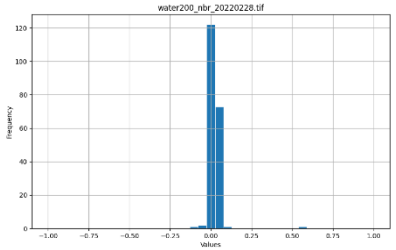


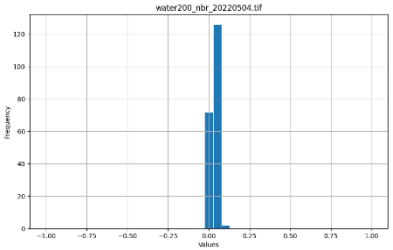


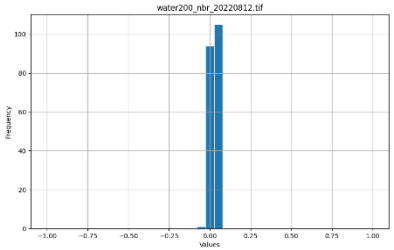


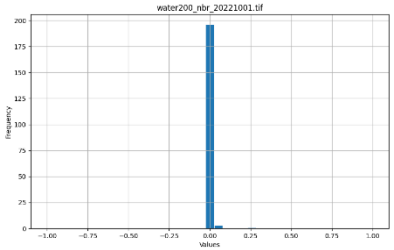


NBR









Anexo D – Repositório de código para o desenvolvimento metodológico

Anexo D1 – Programa 1: *Resampling Sentinel-2 images to 10 meter spatial resolution* (código existente).

Bloco de código, em *pybton*, preparado para realizar a reamostragem do tamanho do pixel das imagens Sentinel-2 para 10 metros e recorte pelo limite da área de estudo, seguido do seu estudo de bibliotecas e funções implementadas, uma vez que o presente código foi disponibilizado pela orientação.

Resampling Sentinel-2 images to a 10 meter spatial resolution

In [1]:
```
img_folder = '/home/francisco/create_grid/stl_2/stl2020'
ofolder    = '/home/francisco/create_grid/stl_2/step1_2020'

cliprst = '/home/francisco/create_grid/stl_2/ref_raster/band_ref.tif'
```

In [2]:
```
import os

from glass.pys.oss  import lst_ff, mkdir
from glass.pys.tm   import now_as_str
from glass.dtr.stl  import unzip_img
from glass.rst.rmp  import match_cellsize_and_clip
```

In [3]:
```
imgs = lst_ff(img_folder, file_format='.zip')
```

In [4]:
```
# Image Resampling

for img in imgs:
    # Create folder for temp files
    zfolder = mkdir(os.path.join(ofolder, now_as_str()))

    # Unzip images
    bands = unzip_img(img, zfolder)
    # Match cellsize and clip
    bands = match_cellsize_and_clip(
        [bands[b] for b in bands],
        bands["B02_10m"], ofolder,
        isint=True, clipgeo=cliprst,
        ws=zfolder
    )
```

Análise e estudo do Programa 1: *Resampling Sentinel-2 images to 10 meter spatial resolution*

Resampling Sentinel-2 images to a 10 meter spatial resolution

```
[1]: img_folder = '/home/francisco/lisboa/img_s2'
     ofolder    = '/home/francisco/lisboa/paco1'

     cliprst = '/home/francisco/lisboa/area_rst/lm_lisboa.shp'
```

Caminho utilizado para chegar às imagens Sentinel-2

Caminho para os *outputs* das bandas com resolução de 10 m

Raster referente à área de estudo utilizado como referência para operações de recorte das várias bandas

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

```
[3]: imgs = lst_ff(img_folder, file_format='.zip')
```

```
[4]: # Image Resampling

     for img in imgs:
         # Create folder for temp files
         zfolder = mkdir(os.path.join(ofolder, now_as_str()))

         # Unzip images
         bands = unzip_img(img, zfolder)

         # Match cellsize and clip
         bands = match_cellsize_and_clip(
             [bands[b] for b in bands],
             bands["B02_10m"], ofolder,
             isint=True, clipgeo=cliprst,
             ws=zfolder
         )
```

Programa 1: Extremamente importante para iniciar o pré-processamento das imagens de satélite. O código tem a principal tarefa de aceder às imagens Sentinel-2, realizar o *resampling* das diferentes bandas para uma resolução espacial de 10 m (a resolução espacial das diferentes bandas do Sentinel-2 variam) e ainda fazer o recorte pela área de estudo que se pretende trabalhar.

Imports:

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

- *import* OS: o módulo OS em *python* fornece funções para interagir com o sistema operacional. O sistema operacional vem sob os módulos padrão do *python*. Este módulo

fornece uma maneira flexível de usar a funcionalidade dependendo do sistema operacional. Os módulos OS e "*os.path*" incluem muitas funções para interagir com o sistema de pastas.

- *from* glass.pys.oss *import* lst_ff, mkdir (…): são diferentes *scripts* pré-criados e importados de outras pastas, introduzidos no Programa 1 através do caminho de pastas. Todos os *imports* seguintes serão explicados nas secções seguintes.

Módulos e ferramentas:

#*Resampling* das imagens

#Criação de uma pasta para ficheiros temp

```
[4]: # Image Resampling

for img in imgs:
    # Create folder for temp files
    zfolder = mkdir(os.path.join(ofolder, now_as_str()))
```

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes de caminho de forma inteligente. Este método concatena vários componentes de caminho com exatamente um separador de pastas ('/') após cada parte não vazia, exceto o último componente do caminho. Exemplo: '/home/glass/francisco/'

Principais *Imports* implementados no Programa 1 e descrição dos mesmos

Na figura seguinte são apresentados os diferentes *Imports* incorporados no Programa 1 com funções e objetivos extremamente importantes para o desempenho do código.

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

Import OS, único módulo como biblioteca universal criada para utilizar a partir de linguagem *pyhton*, enquanto os *imports* restantes são *scripst*/códigos, desenvolvidos pelo orientador, integrados no repositório GLASS.

Import OS

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

O módulo OS em *python* fornece funções para interagir com o sistema operacional. OS, vem nos módulos padrão do *python*. Este módulo fornece uma maneira flexível de usar a funcionalidade dependendo do sistema operacional. Os módulos "OS" e "*os.path*" incluem muitas funções para interagir com o sistema de ficheiros e pastas.

Import **lst_ff –** ***from*** **glass.pys.oss** ***import*** **lst_ff**

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

"Lista o caminho onde se encontram todos os ficheiros com uma extensão específica"

```
def lst_ff(w, file_format=None, filename=None, fnpart=None, rfilename=N
    """
    List the abs path of all files with a specific extension on a folde
    """

    from glass.pys import obj_to_lst

    filename = None if filename and fnpart else filename

    # Prepare file format list
    if file_format:
        formats = obj_to_lst(file_format)

        for f in range(len(formats)):
            if formats[f][0] != '.':
                formats[f] = '.' + formats[f]

    # List files
    r = []
    for (d, _d_, f) in os.walk(w):
        r.extend(f)
        break

    # Filter files by format or not
    if not file_format:
        if not rfilename:
            t = [os.path.join(w, i) for i in r]
        else:
            t = [i for i in r]

    else:
        if not rfilename:
            t = [
                os.path.join(w, i) for i in r
                if os.path.splitext(i)[1] in formats
            ]
        else:
            t = [i for i in r if os.path.splitext(i)[1] in formats]

    # Filter by filename
    if not filename and not fnpart: return t

    elif filename and not fnpart:
        filename = obj_to_lst(filename)

        _t = []
        for i in t:
            fn = fprop(i, 'fn') if not rfilename else i
            if fn in filename:
                _t.append(i)

        return _t

    elif not filename and fnpart:
        fnp = obj_to_lst(fnpart)

        _t = []
        for i in t:
            fn = fprop(i, 'fn') if not rfilename else i

            for _f in fnp:
                if _f in fn:
                    _t.append(i)
                    break

        return _t
```

Módulos e Ferramentas:

#Lista de ficheiros

```
93      # List files
94      r = []
95      for (d, _d_, f) in os.walk(w):
96          r.extend(f)
97          break
```

- *os.walk*: apresenta os nomes dos ficheiros em uma sequência de pastas percorrendo essa sequência de cima para baixo e de baixo para cima.

- *extend*: método que adiciona todos os elementos de um iterável (lista, tupla, *string*, etc.) ao final da lista.

#Filtrar os ficheiros em função do formato ou não

```
99      # Filter files by format or not
100     if not file_format:
101         if not rfilename:
102             t = [os.path.join(w, i) for i in r]
103         else:
104             t = [i for i in r]
105
106     else:
107         if not rfilename:
108             t = [
109                 os.path.join(w, i) for i in r
110                 if os.path.splitext(i)[1] in formats
111             ]
112         else:
113             t = [i for i in r if os.path.splitext(i)[1] in formats]
114
```

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes dos caminhos de forma inteligente. Este método concatena vários componentes dos caminhos com exatamente um separador de pasta (‘/’) após cada parte não vazia, exceto o último elemento do caminho. Exemplo: ‘/home/glass/francisco/’

- *os.path.splitext*: o método *OS.path* é um submódulo do módulo *OS* em *python* utilizado para manipular nomes dos caminhos. *Os.path.splitext()* é o método usado em *python* para dividir o nome do caminho em um par *root* e *ext*. Aqui, *ext* significa extensão e possui a parte da extensão do caminho especificado, enquanto *root* é tudo, exceto a parte *ext*.

#Filtrar em função do nome do ficheiro

```
    # Filter by filename
    if not filename and not fnpart: return t

    elif filename and not fnpart:
        filename = obj_to_lst(filename)

        _t = []
        for i in t:
            fn = fprop(i, 'fn') if not rfilename else i
            if fn in filename:
                _t.append(i)

        return _t

    elif not filename and fnpart:
        fnp = obj_to_lst(fnpart)

        _t = []
        for i in t:
            fn = fprop(i, 'fn') if not rfilename else i

            for _f in fnp:
                if _f in fn:
                    _t.append(i)
                    break

        return  t
```

- *append*: método que acrescenta um elemento a uma lista final.

Descrição:

No fundo, o *from* glass.pys.oss *import* lst_ff, é um método criado para definir caminhos de ficheiros em formatos de lista, que ajuda a percorrer as diferentes pastas até chegar aos ficheiros pretendidos, importante para procurar dados e colocá-los no Programa 1. Para além disso, este método ajuda na definição da extensão do ficheiro de forma que o Programa 1, não tenha qualquer dificuldade em realizar a sua leitura, pois filtra os ficheiros em função do formato e em função do nome. Sem dúvida que é importante para realizar os caminhos das pastas dos ficheiros e assim como perceber em que formato se encontram os ficheiros.

Imports dentro do def lst_ff:

```
    from glass.pys import obj_to_lst
```

Import obj_to_lst dentro do código def lst_ff

"Método que utiliza uma lista, mas o utilizador fornece outro tipo de objeto"

"Este método vai ver se o objeto oferecido pelo utilizador é ou não uma lista e se não for vai converter para uma lista"

```
17 def obj_to_lst(obj):
18     """
19     A method uses a list but the user gives other type of object
20 
21     This method will see if the object is not a list and convert it to a list
22     """
23 
24     return obj if type(obj) == list else [obj] if obj != None else None
```

Descrição:

Script importante, para realizar a conversão de qualquer objeto dado pelo utilizador para um objeto do tipo lista.

Import mkdir – *from* glass.pys.oss *import* mkdir

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

"Cria uma pasta"

"Substitui a pasta, se já existir"

```
203 def mkdir(folder, randName=None, overwrite=True):
204     """
205     Create a new folder
206     Replace the given folder if that one exists
207     """
208 
209     if randName:
210         import random
211         chars = '0123456789qwertyuiopasdfghjklzxcvbnm'
212 
213         name = ''
214         for i in range(10):
215             name+=random.choice(chars)
216 
217         folder = os.path.join(folder, name)
218 
219     if os.path.exists(folder):
220         if overwrite:
221             import shutil
222 
223             shutil.rmtree(folder)
224         else:
225             raise ValueError(f"{folder} already exists")
226 
227     os.mkdir(folder)
228 
229     return folder
```

Módulos e Ferramentas:

```
209     if randName:
210         import random
211         chars = '0123456789qwertyuiopasdfghjklzxcvbnm'
```

- *import random*: Este módulo implementa resultados de números e letras pseudoaleatórias para várias distribuições.

```
213         name = ''
214         for i in range(10):
215             name+=random.choice(chars)
216 
217         folder = os.path.join(folder, name)
```

- *random.choice:* devolve um elemento aleatório da sequência de números e letras dadas pelo utilizador.

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes dos caminhos de forma inteligente. Este método concatena vários componentes dos caminhos com exatamente um separador de pasta (‘/’) após cada parte não vazia, exceto o último elemento do caminho. Exemplo: ‘/home/glass/francisco/’.

```
219     if os.path.exists(folder):
220         if overwrite:
221             import shutil
222 
223             shutil.rmtree(folder)
224         else:
225             raise ValueError(f"{folder} already exists")
```

- *os.path.exists*: este método em *python* é usado para verificar se o caminho especificado existe. Este método também pode ser usado para verificar se o caminho fornecido de refere a uma pasta aberto ou não.

- *import shutil*: é um modelo que oferece várias operações de alto nível para ficheiros e conjuntos de ficheiros. São fornecidas, em particular, funções que suportam a cópia ou exclusão de ficehiros.

- *shutil.rmtree*: elemina uma pasta ou um caminho da pasta.

```
226
227     os.mkdir(folder)
```

- *os.mkdir*: método em *phyton* usado para criar uma pasta chamado *path* com o modo numérico especificado. Este método gera *FileExistsError* se a pasta a ser criada já existir.

Descrição:

No fundo, o *from* glass.pys.oss *import* mkdir é um método criado para atribuir caminhos e nomes a pastas criadas pelo Programa 1. Cada vez que o Programa 1 é corrido, os nomes das pastas são criados de forma aleatória, e em caso de criação de pastas com o mesmo nome, a mais antiga é eliminada e substituída pela pasta criada mais recente.

***Import* now_as_str – *from* glass.pys.tm *import* now_as_str**

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

"Devolve a data e a hora como uma *string*"

```
23  def now_as_str():
24      """
25      Return Datetime.now as string
26      """
27
28      mapDic = {
29          '00' : '9',
30          '01' : 'a', '02' : 'b', '03' : 'c', '04' : 'd', '05'  : 'e',
31          '06' : 'f', '07' : 'g', '08' : 'h', '09' : 'i', '10' : 'j',
32          '11' : 'k', '12' : 'l', '13' : 'm', '14' : 'n', '15' : 'o',
33          '16' : 'p', '17' : 'q', '18' : 'r', '19' : 's', '20' : 't',
34          '21' : 'u', '22' : 'v', '23' : 'w', '24' : 'x', '25' : 'y',
35          '26' : 'z', '27' : 'A', '28' : 'B', '29' : 'C', '30' : 'D',
36          '31' : 'E', '32' : 'F', '33' : 'G', '34' : 'H', '35' : 'I',
37          '36' : 'J', '37' : 'K', '38' : 'L', '39' : 'M', '40' : 'N',
38          '41' : 'O', '42' : 'P', '43' : 'Q', '44' : 'R', '45' : 'S',
39          '46' : 'T', '47' : 'U', '48' : 'V', '49' : 'W', '50' : 'X',
40          '51' : 'Y', '52' : 'Z', '53' : '1', '54' : '2', '55' : '3',
41          '56' : '4', '57' : '5', '58' : '6', '59' : '7', '60' : '8',
42      }
43
44      N = str(now_as_int())[2:]
45      L = ''
46      for c in range(0, len(N), 2):
47          n = N[c:c+2]
48          L += mapDic[n]
49
50      return L
```

Módulos e Ferramentas:

```
    N = str(now_as_int())[2:]
    L = ''
    for c in range(0, len(N), 2):
        n = N[c:c+2]
        L += mapDic[n]

    return L
```

- *str*: converte o valor especificado numa *string*.

Imports dentro do now_as_str:

```
    N = str(now_as_int())[2:]
    L = ''
    for c in range(0, len(N), 2):
        n = N[c:c+2]
        L += mapDic[n]

    return L
```

Import now_as_int dentro do código now_as_str

"Devolve a data e a hora como um inteiro"

```
def now_as_int():
    """
    Return Datetime.now as integer
    """

    import datetime

    _now = str(datetime.datetime.now())

    _now = _now.replace('-', '')
    _now = _now.replace(' ', '')
    _now = _now.replace(':', '')
    _now = _now.split('.')[0]

    return int(_now)
```

Módulos e Ferramentas:

- *import datetime*: módulo que fornece classes para manipular a data e horas. Embora a aritmética da data e da hora seja suportada, a implementação tem o objetivo de extrair de forma eficiente os atributos de saída.

- *datetime.now*: função que fornece atributos extras para extrair dados individuais.

- *replace*: substitui uma frase ou um elemento por outra frase ou elemento específico.

- *split*: método que divide uma *string* em uma lista. Pode especificar o separador, como um separador padrão ou até mesmo um espaço em branco.

Descrição:

Script importante para fazer a conversão de elementos relativos à data e à hora de um ficheiro, considerados como um valor inteiro e transformá-los numa *string*.

No fundo, este *script*, auxilia na atribuição do nome para os ficheiros de saída do Programa 1 das bandas do Sentinel-2. Ou seja, vai pegar na informação dada pelo nome de cada banda (nº da banda e data) e convertê-la para *string*. Posteriormente, essa informação recolhida vai ser dada a cada ficheiro de saída.

Import **unzip_img –** ***from*** **glass.dtr.stl** ***import*** **unzip_img**

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

"Extrai de uma pasta zipada as imagens Sentinel-2"

```
def unzip_img(zip_file, out_folder):
    """
    Unzip Sentinel-2 Image
    """

    import os
    from zipfile        import ZipFile
    from glass.cons.stl import get_ibands
    from glass.pys.oss  import copy_file

    intbands = get_ibands()


    with ZipFile(zip_file, 'r') as zipo:
        zipff = zipo.namelist()

        rbands = {}
        for b in intbands:
            for f in zipff:
                if b in f:
                    ob = os.path.join(out_folder, os.path.basename(f))

                    zipo.extract(f, out_folder, pwd=None)
                    copy_file(
                        os.path.join(out_folder, f), ob
                    )
                    rbands[b] = ob

                    break

    return rbands
```

Módulos e Ferramentas:

```
11    from zipfile       import ZipFile
```

- *import zipfile*: este módulo fornece ferramentas para criar, ler, gravar, anexar e listar um ficheiro .*zip*.

```
18    with ZipFile(zip_file, 'r') as zipo:
19        zipff = zipo.namelist()
```

- *zipfile.namelist*: devolve uma lista com os elementos dos ficheiros dentro de uma pasta .*zip*.

```
21        rbands = {}
22        for b in intbands:
23            for f in zipff:
24                if b in f:
25                    ob = os.path.join(out_folder, os.path.basename(f))
```

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes dos caminhos de forma inteligente. Este método concatena vários componentes dos caminhos com exatamente um separador de pasta (‘/’) após cada parte não vazia, exceto o último elemento do caminho. Exemplo: ‘/home/glass/francisco/’.

- *os.path.basename*: é utilizado em *python* para obter o nome base do caminho especifico. Este método utiliza internamente o método *os.path.split()* para dividir o caminho especifico em parte da frente e parte de trás. O método *os.path.basename()* devolve a parte final depois de dividir o caminho específico em parte da frente e parte de trás.

```
27                    zipo.extract(f, out_folder, pwd=None)
28                    copy_file(
29                        os.path.join(out_folder, f), ob
30                    )
31                    rbands[b] = ob
32
33                    break
34
35    return rbands
```

- *zipfile.extract*: extrai um membro do ficheiro para a pasta de trabalho atual.

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes dos caminhos de forma inteligente. Este método concatena vários componentes dos caminhos com exatamente um separador de pasta (‘/’) após cada parte não vazia, exceto o último elemento do caminho. Exemplo: ‘/home/glass/francisco/’.

Descrição:

Contudo, este *script* ajuda a extrair as bandas e a sua informação das pastas zipadas (originais do seu descarregamento). Para além de extrair a informação das imagens, copia as bandas de cada imagem para a pasta onde se pretende trabalhar.

Imports dentro do unzip_img:

```
13      from glass.pys.oss  import copy_file
```

Import get_ibands dentro do código unzip_img

```
12      from glass.cons.stl import get_ibands
```

```
20  def get_ibands():
21      return [
22          'B02_10m', 'B03_10m', 'B04_10m', 'B08_10m',
23          'B05_20m', 'B06_20m', 'B07_20m', 'B8A_20m',
24          'B11_20m', 'B12_20m', 'B09_60m', 'B01_60m',
25          'AOT_10m', 'SCL_20m'
26      ]
```

Descrição:

Código get_ibands dentro do *script* unzip_img, com o objetivo de devolver as bandas mais importantes que compõem a imagem Sentinel-2. Função que serve para extrair as bandas, com o seu respetivo número e resolução.

Import copy_file dentro do código unzip_img

```
12      from glass.cons.stl import get_ibands
13      from glass.pys.oss  import copy_file
```

"Copia ficheiros"

```
417 def copy_file(src, dest, move=None):
418     """
419     Copy a file
420     """
421 
422     if not move:
423         from shutil import copyfile
424 
425         copyfile(src, dest)
426 
427     else:
428         from shutil import move as mv
429 
430         mv(src, dest)
431 
432     return dest
```

Métodos e Ferramentas:

- *shutil.copyfile*: em *python* é utilizado para copiar conteúdo da pasta de origem para a pasta de destino. Os metadados dos ficheiros não são copiados. A origem e o destino devem representar uma pasta onde é permitido gravar ficheiros. Se o destino já existir, ele será substituído pela pasta de origem, caso contrário, uma nova pasta será criada.

- *shutil.move*: movimenta ficheiros da pasta (origem) para outro local (destino) com devolução do destino. Se a pasta de destino já existir, o *src* será colocado dentro dessa pasta. Se o destino já existir, mas se não for uma pasta, ele poderá ser substituído, dependendo da *os.rename()* semântica.

Descrição:

Este *script*, copy_file é importantes para realizar a cópia de ficheiros, assim como permite movê-los e colocá-los em pastas de destinos pré-definidas pelo utilizador.

Import match_cellsize_and_clip – _from_ glass.rst.rmp _import_ match_cellsize_and_clip

```
[2]: import os

     from glass.pys.oss  import lst_ff, mkdir
     from glass.pys.tm   import now_as_str
     from glass.dtr.stl  import unzip_img
     from glass.rst.rmp  import match_cellsize_and_clip
```

"*Resample* das imagens para que tenham, a mesma resolução e sejam cortadas pela área de estudo"

"Excelente para realizar a reamostragem das bandas do Sentinel-2 com uma resolução superior a 10 metros"

"Dependências: - *GRASS GIS*; - *GDAL/OGR*"

```
def match_cellsize_and_clip(rstBands, refRaster, outFolder,
                            clipgeo=None, isint=None, ws=None):
    """
    Resample images to make them with the same resolution and clip

    Good to resample Sentinel bands with more than 10 meters.

    Dependencies:
    * GRASS GIS;
    * GDAL/OGR.
    """

    import os
    from glass.prop     import is_rst
    from glass.prop.prj import get_rst_epsg
    from glass.wenv.grs import run_grass
    from glass.pys.oss  import fprop, mkdir
    from glass.pys.tm   import now_as_str

    # Check if outfolder exists
    if not os.path.exists(outFolder):
        mkdir(outFolder, overwrite=None)

    # Get EPSG from refRaster
    epsg = get_rst_epsg(refRaster, returnIsProj=None)

    """
    Start GRASS GIS Session
    """
    ws = mkdir(os.path.join(outFolder, now_as_str())) \
        if not ws else ws
    grsb = run_grass(
        ws, grassBIN='grass78', location='resample',
        srs=epsg
    )

    import grass.script.setup as gsetup

    gsetup.init(grsb, ws, 'resample', 'PERMANENT')

    """
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst

    # Send Ref Raster to GRASS GIS and set region
    extRst = rst_to_grs(refRaster, 'ext_rst')
    rst_to_region(extRst)

    # Import all bands in rstBands
    grs_bands = [rst_to_grs(i, fprop(i, 'fn')) for i in rstBands]

    if clipgeo:
        clip_is_rst = is_rst(clipgeo)

        # Add clipgeo to GRASS
        if not clip_is_rst:
            grs_clip = shp_to_grs(clipgeo, fprop(clipgeo, 'fn'), asCMD=True)

            # SHP to Raster
            rst_clip = shp_to_rst(
                grs_clip, 1, f'rst_{grs_clip}',
                cmd=True
            )
```

(Parte final do *script* match_cellsize_and_clip)

```
        else:
            rst_clip = rst_to_grs(clipgeo, fprop(clipgeo, 'fn'))

        # Set region using
        rst_to_region(rst_clip)

        # Set mask
        grs_to_mask(rst_clip)

    # Export bands
    return [grs_to_rst(
        i, os.path.join(outFolder, i + '.tif'),
        is_int=isint
    ) for i in grs_bands]
```

Métodos e Ferramentas:

#Verifica se a pasta de saída já existe

```
    # Check if outfolder exists
    if not os.path.exists(outFolder):
        mkdir(outFolder, overwrite=None)
```

- *os.path.exists*: este método em *python* é usado para verificar se o caminho especificado existe. Este método também pode ser usado para verificar se o caminho fornecido de refere a uma pasta aberto ou não. Tem como referência o *script* mkdir, uma vez que este código é relevante para a criação de pastas e substituir a criação de pastas se estas já existirem.

"Início de Sessão GRASS GIS"

```
    Start GRASS GIS Session
    """
    ws = mkdir(os.path.join(outFolder, now_as_str())) \
        if not ws else ws
    grsb = run_grass(
        ws, grassBIN='grass78', location='resample',
        srs=epsg
```

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipulação comum de nomes de caminho. O método *os.path.join()* em *python* une um ou mais componentes de caminho de forma inteligente. Este método concatena vários componentes de caminho com exatamente um separador de diretório ('/') após cada parte não vazia, exceto o último componente do caminho. Exemplo: '/home/glass/francisco/'. Utilizado com referência ao código mkdir, para localizar e guardar o início de sessão do GRASS GIS numa determinada pasta.

```
42      import grass.script.setup as gsetup
43
44      gsetup.init(grsb, ws, 'resample', 'PERMANENT')
```

- *import.grass.script.setup as gsetup*: importa algumas partes convenientes do GRASS GIS a partir do *Pyhton* API.

- *gsetup.init*: importante para iniciar a sessão do GRASS, é esta linha que faz com que a sessão se inicie.

#Exporta as bandas das imagens Sentinel-2

```
83      # Export bands
84      return [grs_to_rst(
85          i, os.path.join(outFolder, i + '.tif'),
86          is_int=isint
87      ) for i in grs_bands]
```

- os.path.join: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes dos caminhos de forma inteligente. Este método concatena vários componentes dos caminhos com exatamente um separador de pasta (‘/’) após cada parte não vazia, exceto o último elemento do caminho. Exemplo: ‘/home/glass/francisco/’. Utilizado para encaminhar as bandas retiradas das imagens Sentinel-2 e acrescentar à definição de cada ficheiro a extensão final *.tif*.

Descrição:

Este código match_cellsize_and_clip, é extremamente complexo e importa várias funções. As tarefas principais variam entre assegurar uma pasta de saída para guardar os ficheiros das diferentes bandas extraídas das imagens Sentinel-2 com uma resolução espacial de 10 m, receber o código EPSG do *raster* de área de estudo utilizado para fazer o corte de delimitação da área de estudo, iniciar a sessão no GRASS GIS com a extensão da área de estudo do *raster* de referência e atribuir o sistema de coordenadas em que todas as imagens vão ser trabalhadas. No fim de realizar o *resampling* das diferentes bandas que compõem as imagens e respetivo recorte a partir da referência do *raster* da área de estudo, exporta os ficheiros para o caminho definido pelo utilizador.

Imports dentro do match_cellsize_and_clip:

```
def match_cellsize_and_clip(rstBands, refRaster, outFolder,
                            clipgeo=None, isint=None, ws=None):
    """
    Resample images to make them with the same resolution and clip

    Good to resample Sentinel bands with more than 10 meters.

    Dependencies:
    * GRASS GIS;
    * GDAL/OGR.
    """

    import os
    from glass.prop     import is_rst
    from glass.prop.prj import get_rst_epsg
    from glass.wenv.grs import run_grass
    from glass.pys.oss  import fprop, mkdir
    from glass.pys.tm   import now_as_str
```

Imports packages relacionados com a sessão GRASS GIS:

```
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

Import is_rst dentro do código match_cellsize_and_clip

```
def match_cellsize_and_clip(rstBands, refRaster, outFolder,
                            clipgeo=None, isint=None, ws=None):
    """
    Resample images to make them with the same resolution and clip

    Good to resample Sentinel bands with more than 10 meters.

    Dependencies:
    * GRASS GIS;
    * GDAL/OGR.
    """

    import os
    from glass.prop     import is_rst
    from glass.prop.prj import get_rst_epsg
    from glass.wenv.grs import run_grass
    from glass.pys.oss  import fprop, mkdir
    from glass.pys.tm   import now_as_str
```

```
def is_rst(_file):
    from glass.pys.oss import fprop

    rst_lst = raster_formats()

    file_ext = fprop(_file, 'ff')

    if file_ext not in rst_lst:
        return None
    else:
        return True
```

Descrição:

Script importante para definir que o ficheiro a ser utilizado é um *raster* e não uma *shapefile*, ou outro tipo de dado. Serve para certificar se esse ficheiro será um *raster* e não uma *shape*.

Imports dentro do is_rst:

```
20 def is_rst( file):
21     from glass.pys.oss import fprop
22 
23     rst_lst = raster_formats()
24 
25     file_ext = fprop(_file, 'ff')
26 
27     if file_ext not in rst_lst:
28         return None
29     else:
30         return True
```

Import fprop dentro do código is_rst

```
def is_rst(_file):
    from glass.pys.oss import fprop
```

"Devolve a propriedade de um certo ficheiro"

"Opções prop: - filename ou fn – devolve o nome e as propriedades do ficheiro"

```
def fprop(__file, prop, forceLower=None, fs_unit=None):
    """
    Return some property of file

    prop options:
    * filename or fn - return filename
    """

    from glass.pys import obj_to_lst

    prop = obj_to_lst(prop)

    result = {}

    if 'filename' in prop or 'fn' in prop:
        fn, ff = os.path.splitext(os.path.basename(__file))

        result['filename'] = fn if not forceLower else fn.lower()

        if 'fileformat' in prop or 'fn' in prop:
            result['fileformat'] = ff

    elif 'fileformat' in prop or 'ff' in prop:
        result['fileformat'] = os.path.splitext(__file)[1]

    if 'filesize' in prop or 'fs' in prop:
        fs_unit = 'MB' if not fs_unit else fs_unit

        fs = os.path.getsize(__file)

        if fs_unit == 'MB':
            fs  = (fs / 1024.0) / 1024

        elif fs_unit == 'KB':
            fs = fs / 1024.0

        result['filesize'] = fs

    if len(prop) == 1:
        if prop[0] == 'fn':
            return result['filename']
        elif prop[0] == 'ff':
            return result['fileformat']
        elif prop[0] == 'fs':
            return result['filesize']
        else:
            return result[prop[0]]
    else:
        return result

```

Métodos e Ferramentas:

```
    if 'filename' in prop or 'fn' in prop:
        fn, ff = os.path.splitext(os.path.basename(__file))

        result['filename'] = fn if not forceLower else fn.lower()
```

- *os.path.splitext*: o método *OS.path* é um submódulo do módulo *OS* em *python* utilizado para manipular nomes dos caminhos. *Os.path.splitext()* é o método usado em *python* para dividir o nome do caminho em um par *root* e *ext*. Aqui, *ext* significa extensão e possui a parte da extensão do caminho especificado, enquanto *root* é tudo, exceto a parte *ext*.

- *os.path.basename*: é utilizado em *python* para obter o nome base do caminho especifico. Este método utiliza internamente o método *os.path.split()* para dividir o caminho especifico em parte da frente e parte de trás. O método *os.path.basename()* devolve a parte final depois de dividir o caminho específico em parte da frente e parte de trás.

- *lower*: devolve uma *string* em que todos os caracteres são minúsculos.

```
50      if 'filesize' in prop or 'fs' in prop:
51          fs_unit = 'MB' if not fs_unit else fs_unit
52
53          fs = os.path.getsize(__file)
```

- *os.path.getcize*: é usado em *python* para verificar o tamanho do caminho (ficheiro) específico. Este método devolve o tamanho do caminho em *bytes*.

Descrição:

Este *script* fprop é sem dúvida um dos mais importantes e utilizados para este tipo de trabalhos, uma vez que consegue verificar as propriedades do ficheiro que é importado para o *script* e tem a capacidade de devolver as suas características e o nome para distinguir de outros ficheiros. Nesta situação como é utilizado para definição de *rasters*, acaba por ser importante para a definição das diferentes bandas das imagens Sentinel-2 e para além disso, tem a capacidade de definir o nome de cada banda e extensão para que seja possível distingui-las uma das outras. O código devolve o primeiro e o último nome do ficheiro, o seu formato e ainda o seu tamanho.

Imports dentro do fprop:

```
25  def fprop(__file, prop, forceLower=None, fs_unit=None):
26      """
27      Return some property of file
28
29      prop options:
30      * filename or fn - return filename
31      """
32
33      from glass.pys import obj_to_lst
34
35      prop = obj_to_lst(prop)
36
37      result = {}
38
39      if 'filename' in prop or 'fn' in prop:
40          fn, ff = os.path.splitext(os.path.basename(__file))
41
42          result['filename'] = fn if not forceLower else fn.lower()
43
44          if 'fileformat' in prop or 'fn' in prop:
45              result['fileformat'] = ff
46
47      elif 'fileformat' in prop or 'ff' in prop:
48          result['fileformat'] = os.path.splitext(__file)[1]
49
50      if 'filesize' in prop or 'fs' in prop:
51          fs_unit = 'MB' if not fs_unit else fs_unit
52
53          fs = os.path.getsize(__file)
54
```

Import obj_to_lst dentro do código fprop

```
33      from glass.pys import obj_to_lst
```

"Método onde o utilizador fornece qualquer tipo de objeto para uma lista"

"Este método certifica-se se o objeto é uma lista e se não for vai converter para uma lista"

```
17  def obj_to_lst(obj):
18      """
19      A method uses a list but the user gives other type of object
20
21      This method will see if the object is not a list and convert it to a list
22      """
23
24      return obj if type(obj) == list else [obj] if obj != None else None
```

Descrição:

Código importante que realiza a conversão de qualquer objeto, de um certo tipo, para uma extensão em *python* definida, nomeadamente uma lista. Está incorporado no *script* fprop porque vai fazer com que as propriedades extraídas de um ficheiro sejam guardadas numa lista.

Import get_rst_epsg dentro do código match_cellsize_and_clip

```
6   def match_cellsize_and_clip(rstBands, refRaster, outFolder,
7                               clipgeo=None, isint=None, ws=None):
8       """
9       Resample images to make them with the same resolution and clip
10
11      Good to resample Sentinel bands with more than 10 meters.
12
13      Dependencies:
14      * GRASS GIS;
15      * GDAL/OGR.
16      """
17
18      import os
19      from glass.prop     import is_rst
20      from glass.prop.prj import get_rst_epsg
21      from glass.wenv.grs import run_grass
22      from glass.pys.oss  import fprop, mkdir
23      from glass.pys.tm   import now_as_str
```

"Devolve o código EPSG do sistema de referência espacial de um *raster*".

```
142 def get_rst_epsg(rst, returnIsProj=None):
143     """
144     Return the EPSG Code of the Spatial Reference System of a Raster
145     """
146 
147     import os
148     from osgeo import gdal
149     from glass.prop.img import rst_epsg
150 
151     if not os.path.exists(rst):
152         raise ValueError((
153             f"{rst} does not exist! Please give a valid "
154             "path to a raster file"
155         ))
156 
157     d = gdal.Open(rst)
158 
159     if not returnIsProj:
160         epsg = rst_epsg(d, isproj=None)
161 
162         return epsg
163     else:
164         epsg, isproj = rst_epsg(d, isproj=True)
165 
166         return epsg, isproj
167 
```

Métodos e Ferramentas:

```
147     import os
148     from osgeo import gdal
```

- *from osgeo import gdal:* GDAL é uma biblioteca C++ para mais de 200 formatos de dados geospaciais *raster* e vetoriais. É lançado sob uma licença *open source pela Open Source Geospacial Foudation.*

```
151     if not os.path.exists(rst):
152         raise ValueError((
153             f"{rst} does not exist! Please give a valid "
154             "path to a raster file"
155         ))
```

- *os.path.exists*: este método em *python* é usado para verificar se o caminho especificado existe. Este método também pode ser utilizado para verificar se o caminho indicado se relaciona com uma pasta aberto ou não.

```
157     d = gdal.Open(rst)
158 
159     if not returnIsProj:
160         epsg = rst_epsg(d, isproj=None)
161 
162         return epsg
163     else:
164         epsg, isproj = rst_epsg(d, isproj=True)
165 
166         return epsg, isproj
```

- *gdal.Open*: função utilizada para abrir um *dataset*.

Descrição:

Código importante para adquirir o código EPSG do sistema de referência de um *raster* que já contenha nas suas propriedades esse sistema de coordenadas definido. O que este *script* vai fazer é retirar essa informação e guardá-la para utilizar como referência para o início de sessão de GRASS GIS.

Imports dentro do get_rst_epsg:

```
142 def get_rst_epsg(rst, returnIsProj=None):
143     """
144     Return the EPSG Code of the Spatial Reference System of a Raster
145     """
146 
147     import os
148     from osgeo import gdal
149     from glass.prop.img import rst_epsg
150 
151     if not os.path.exists(rst):
152         raise ValueError((
153             f"{rst} does not exist! Please give a valid "
154             "path to a raster file"
155         ))
156 
157     d = gdal.Open(rst)
158 
159     if not returnIsProj:
160         epsg = rst_epsg(d, isproj=None)
161 
162         return epsg
163     else:
164         epsg, isproj = rst_epsg(d, isproj=True)
165 
166         return epsg, isproj
167 
```

***Import* rst_epsg dentro do código get_rst_epsg**

"Devolve o EPSG do *raster*"

```
23 def rst_epsg(img, isproj=None):
24     """
25     Return Raster EPSG
26     """
27 
28     from osgeo import osr
29 
30     proj = osr.SpatialReference(wkt=img.GetProjection())
31 
32     if not proj:
33         raise ValueError(
34             'img obj has not Spatial Reference assigned!'
35         )
36 
37     epsg = int(str(proj.GetAttrValue('AUTHORITY', 1)))
38 
39     if not isproj:
40         return epsg
41 
42     else:
43         if proj.IsProjected:
44             mod_proj = proj.GetAttrValue(str('projcs'))
45 
46             if not mod_proj:
47                 return epsg, None
48 
49             else:
50                 return epsg, True
51         else:
52             return epsg, None
```

Métodos e Ferramentas:

```
28      from osgeo import osr
29
30      proj = osr.SpatialReference(wkt=img.GetProjection())
31
32      if not proj:
33          raise ValueError(
34              'img obj has not Spatial Reference assigned!'
35          )
```

- *from osgeo import osr*: submódulo direcionado para a manipular projeções e sistemas de referência espacial.

- *osr.SpatialReference*: serve para indicar que se vai chamar o sistema de referência.

- *GetProjection*: serve para procurar a projeção do ficheiro definido.

```
37      epsg = int(str(proj.GetAttrValue('AUTHORITY', 1)))
38
39      if not isproj:
40          return epsg
```

- *GetAttrValue*: método utilizado para recolher os parâmetros necessários para o *SpatialReference ()*.

```
42      else:
43          if proj.IsProjected:
44              mod_proj = proj.GetAttrValue(str('projcs'))
45
46              if not mod_proj:
47                  return epsg, None
48
49              else:
50                  return epsg, True
51          else:
52              return epsg, None
```

- *IsProjected*: se o ficheiro já se encontrar com sistema de referência associado.

- *GetAttrValue*: método utilizado para recolher os parâmetros necessários para o *SpatialReference ()*. Se o ficheiro já se encontrar com um sistema de referência associado, esse será extraído e guardado.

Definição:

Código com relevância para verificar se um determinado ficheiro *raster* já tem definido um sistema de referência. Se tiver, esse será retirado e guardado como código EPSG. Nota-se que neste Script são chamadas ferramentas *osr* do submódulo *osgeo*, importantes para a manipular sistemas de referência e projeções definidas nos ficheiros.

Import run_grass dentro do código match_cellsize_and_clip

```
6  def match_cellsize_and_clip(rstBands, refRaster, outFolder,
7                              clipgeo=None, isint=None, ws=None):
8      """
9      Resample images to make them with the same resolution and clip
10
11     Good to resample Sentinel bands with more than 10 meters.
12
13     Dependencies:
14     * GRASS GIS;
15     * GDAL/OGR.
16     """
17
18     import os
19     from glass.prop     import is_rst
20     from glass.prop.prj import get_rst_epsg
21     from glass.wenv.grs import run_grass
22     from glass.pys.oss  import fprop, mkdir
23     from glass.pys.tm   import now_as_str
```

"Método genérico que pode ser utilizado para iniciar o GRASS GIS em qualquer sistema operacional"

"Para trabalhar em Windows, a pasta para o GRASS deve encontrar-se no mesmo caminho das variáveis a utilizar"

```
198 def run_grass(workspace, grassBIN=GRASS_BIN, location=None, srs=None):
199     """
200     Generic method that could be used to put GRASS GIS running in any Os
201
202     To work on MSWindows, Path to GRASS Must be in the PATH Environment
203     Variables
204     """
205
206     from glass.pys.oss import os_name
207
208     __os = os_name()
209
210     if location:
211         from glass.pys.oss import lst_fld
212
213         # Check if location exists
214         flds = lst_fld(workspace, name=True)
215
216         if location in flds:
217             base = start_grass_linux_existLocation(
218                 workspace, grassBin=grassBIN
219             ) if __os == 'Linux' else start_grass_win_exisLocation(
220                 workspace, grassBin=grassBIN)
221
222         else:
223             base = start_grass_linux_newLocation(
224                 workspace, location, srs=srs, grassBin=grassBIN
225             ) if __os == 'Linux' else start_grass_win_newLocation(
226                 workspace, location, srs=srs, grassBin=grassBIN
227             ) if __os == 'Windows' else None
228
229     else:
230         base = start_grass_linux_existLocation(
231             workspace, grassBin=grassBIN
232         ) if __os == 'Linux' else start_grass_win_exisLocation(
233             workspace, grassBin=grassBIN
234         )
235
236     if not base:
237         raise ValueError((
238             'Could not identify operating system'
239         ))
240
241     return base
```

Descrição:

Código importante para iniciar qualquer sessão do *software* GRASS GIS, independentemente do sistema operacional. Este código inicia o GRASS GIS da forma mais simples, no entanto deve ter-se em atenção em que pasta deve ser guardada a sessão. Para trabalhar com os diferentes dados geográficos é importante direcionar a sessão para o mesmo caminho ou pasta das variáveis utilizadas.

Imports dentro do run_grass:

```
198 def run_grass(workspace, grassBIN=GRASS_BIN, location=None, srs=None):
199     """
200     Generic method that could be used to put GRASS GIS running in any Os
201 
202     To work on MSWindows, Path to GRASS Must be in the PATH Environment
203     Variables
204     """
205 
206     from glass.pys.oss import os_name
207 
208     __os = os_name()
209 
210     if location:
211         from glass.pys.oss import lst_fld
212 
```

***Import* os_name dentro do código run_grass**

```
5  import os
6  import re
7 
8  def os_name():
9      import platform
10     return str(platform.system())
```

Métodos e Ferramentas:

- *import OS*: este módulo fornece uma maneira portátil de usar a funcionalidade dependente do sistema operacional.

- *import RE*: esta função fornece operações de correspondência de expressão regular semelhantes às encontradas em *Perl*.

- *os.system*: executa o comando (uma *string*) em um *subshell*. Esse método é implementado aquando se chama a função C padrão *system ()* e tem as mesmas limitações. Se o comando gerar qualquer saída, ela será enviada para o fluxo de saída padrão do interpretador. Sempre

que este método é usado, o respetivo *shell* do sistema operacional é aberto e o comando é executado nele.

- *subshell*: basicamente é um novo *shell* apenas para executar um programa desejado. Um *subshell* pode aceder às variáveis globais definidas pelo "pai *shell*", mas não as variáveis locais. Quaisquer alterações feitas por um *subshell* numa variável global não são passadas para o "pai *shell*".

Descrição:

Código importante para iniciar a sessão do GRASS GIS em qualquer sistema operacional e ajuda no acesso às diferentes variáveis a partir do *shell*, essencial para executar o programa GRASS.

Import lst_fld dentro do código run_grass

```
198 def run_grass(workspace, grassBIN=GRASS_BIN, location=None, srs=None):
199     """
200     Generic method that could be used to put GRASS GIS running in any Os
201 
202     To work on MSWindows, Path to GRASS Must be in the PATH Environment
203     Variables
204     """
205 
206     from glass.pys.oss import os_name
207 
208     __os = os_name()
209 
210     if location:
211         from glass.pys.oss import lst_fld
212 
```

"Direciona o caminho ou o nome das pastas numa pasta"

```
144 def lst_fld(w, name=None):
145     """
146     List folders path or name in one folder
147     """
148 
149     foldersname = []
150     for (dirname, dirsname, filename) in os.walk(w):
151         foldersname.extend(dirsname)
152         break
153 
154     if name:
155         return foldersname
156 
157     else:
158         return [os.path.join(w, fld) for fld in foldersname]
```

Métodos e Ferramentas:

- *os.walk*: apresenta os nomes dos ficheiros numa sequência de pastas percorrendo essa sequência de cima para baixo e de baixo para cima.

- *extend*: método que adiciona todos os elementos de um iterável (lista, tupla, *string*, etc.) ao final da lista.

- *os.path.join*: o módulo *os.path* é um submódulo do módulo *OS* em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes de caminho de forma inteligente. Este método concatena vários componentes de caminho com exatamente um separador de pastas ('/') após cada parte não vazia, exceto o último componente do caminho. Exemplo: '/home/glass/francisco/'.

Descrição:

Código que ajuda na identificação da sequência e no caminho das diferentes pastas e apresenta-as em uma lista com a intenção de auxiliar o utilizador a chegar a diferentes dados e/ou variáveis. Ajuda o *script* principal a percorrer as diferentes pastas utilizadas pelo utilizador para correr o código sem problemas de erros.

Import fprop dentro do código match_cellsize_and_clip

```
6  def match_cellsize_and_clip(rstBands, refRaster, outFolder,
7                              clipgeo=None, isint=None, ws=None):
8      """
9      Resample images to make them with the same resolution and clip
10
11     Good to resample Sentinel bands with more than 10 meters.
12
13     Dependencies:
14     * GRASS GIS;
15     * GDAL/OGR.
16     """
17
18     import os
19     from glass.prop     import is_rst
20     from glass.prop.prj import get_rst_epsg
21     from glass.wenv.grs import run_grass
22     from glass.pys.oss  import fprop, mkdir
23     from glass.pys.tm   import now_as_str
```

(Já explicados: *Import* fprop dentro do código is_rst)

Import mkdir dentro do código match_cellsize_and_clip

```
6  def match_cellsize_and_clip(rstBands, refRaster, outFolder,
7                              clipgeo=None, isint=None, ws=None):
8      """
9      Resample images to make them with the same resolution and clip
10
11     Good to resample Sentinel bands with more than 10 meters.
12
13     Dependencies:
14     * GRASS GIS;
15     * GDAL/OGR.
16     """
17
18     import os
19     from glass.prop     import is_rst
20     from glass.prop.prj import get_rst_epsg
21     from glass.wenv.grs import run_grass
22     from glass.pys.oss  import fprop, mkdir
23     from glass.pys.tm   import now_as_str
```

(Já explicados: *Import* mkdir – from glass.pys.oss import mkdir)

Import now_as_str dentro do código match_cellsize_and_clip

```
6  def match_cellsize_and_clip(rstBands, refRaster, outFolder,
7                              clipgeo=None, isint=None, ws=None):
8      """
9      Resample images to make them with the same resolution and clip
10
11     Good to resample Sentinel bands with more than 10 meters.
12
13     Dependencies:
14     * GRASS GIS;
15     * GDAL/OGR.
16     """
17
18     import os
19     from glass.prop     import is_rst
20     from glass.prop.prj import get_rst_epsg
21     from glass.wenv.grs import run_grass
22     from glass.pys.oss  import fprop, mkdir
23     from glass.pys.tm   import now_as_str
```

(Já explicados: *Import* now_as_str – from glass.pys.tm import now_as_str)

***Import packages* (relacionados com GRASS GIS) rst_to_grs dentro do código match_cellsize_and_clip**

```
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

"Importar *raster* na sessão GRASS"

```
def rst_to_grs(rst, grsRst, lmtExt=None, as_cmd=None):
    """
    Raster to GRASS GIS Raster
    """

    if not as_cmd:
        from grass.pygrass.modules import Module

        __flag = 'o' if not lmtExt else 'or'

        m = Module(
            "r.in.gdal", input=rst, output=grsRst, flags=__flag,
            overwrite=True, run_=False, quiet=True,
        )

        m()

    else:
        from glass.pys import execmd

        rcmd = execmd((
            "r.in.gdal input={} output={} -o{} --overwrite "
            "--quiet"
        ).format(rst, grsRst, "" if not lmtExt else " -r"))

    return grsRst
```

Métodos e Ferramentas:

```
    if not as_cmd:
        from grass.pygrass.modules import Module

        __flag = 'o' if not lmtExt else 'or'

        m = Module(
            "r.in.gdal", input=rst, output=grsRst, flags=__flag,
            overwrite=True, run_=False, quiet=True,
```

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*r.in.gdal*": ferramenta do GRASS GIS, para carregar *rasters* dentro da sessão.

```
    else:
        from glass.pys import execmd

        rcmd = execmd((
            "r.in.gdal input={} output={} -o{} --overwrite "
            "--quiet"
        ).format(rst, grsRst, "" if not lmtExt else " -r"))

    return grsRst
```

- *format*: método que formata os valores especificados e insere-os dentro do espaço reservado da *string*. Este método devolve a *string* formatada.

Descrição:

Código que ajuda a carregar ficheiros *raster* para a sessão iniciada no GRASS GIS implementando ferramentas do GRASS para realizar essas operações.

Imports dentro do rst_to_grs:

```
def rst_to_grs(rst, grsRst, lmtExt=None, as_cmd=None):
    """
    Raster to GRASS GIS Raster
    """

    if not as_cmd:
        from grass.pygrass.modules import Module

        __flag = 'o' if not lmtExt else 'or'

        m = Module(
            "r.in.gdal", input=rst, output=grsRst, flags=__flag,
            overwrite=True, run_=False, quiet=True,
        )

        m()

    else:
        from glass.pys import execmd

        rcmd = execmd((
            "r.in.gdal input={} output={} -o{} --overwrite "
            "--quiet"
        ).format(rst, grsRst, "" if not lmtExt else " -r"))

    return grsRst
```

Import execmd dentro do código rst_to_grs

"Executa um comando e fornece informação sobre os resultados"

```
27 def execmd(cmd):
28     """
29     Execute a command and provide information about the results
30     """
31     import subprocess
32 
33     p = subprocess.Popen(cmd, shell=True,
34                          stdout=subprocess.PIPE, stderr=subprocess.PIPE)
35 
36     out, err = p.communicate()
37 
38     if p.returncode != 0:
39         raise ValueError((
40             'Message: Command execution ended with error\n'
41             f'Command was: {cmd}\n'
42             f'Output: {out.decode("utf-8")}\n'
43             f'Error: {err.decode("utf-8")}'
44         ))
45 
46     else:
47         return out.decode('utf-8')
```

Métodos e Ferramentas:

```
31     import subprocess
32 
33     p = subprocess.Popen(cmd, shell=True,
34                          stdout=subprocess.PIPE, stderr=subprocess.PIPE)
```

- *import subprocess*: módulo que permite gerar novos processos, conectar-se aos caminhos de entrada/saída/erro e obter códigos de retorno.

- *subprocess.Popen*: é um módulo muito flexível que serve para processos de criação e tomadas de decisão por parte dos desenvolvedores, permitindo lidar com casos menos comuns não cobertos pelas funções de conveniência.

- *subprocess.PIPE*: valor especial que pode ser usado como o argumento *stdin, stdout* e *stderr* para que o caminho do fluxo padrão deva ser aberto.

```
36     out, err = p.communicate()
37 
38     if p.returncode != 0:
39         raise ValueError((
40             'Message: Command execution ended with error\n'
41             f'Command was: {cmd}\n'
42             f'Output: {out.decode("utf-8")}\n'
43             f'Error: {err.decode("utf-8")}'
44         ))
```

- *p.communicate*: interage com processos para enviar dados para *stdin*. Lê os dados de *stdou* e *stderr*, até que o fim do ficheiro seja atingido. Aguarda-se que o processo termine para definir o *p.returncode* (atributo).

- *decode*: método em *python* para *strings* que serve para descodificar a *string* utilizando o *codec* registado para a codificar. A *string* codificada pode ser descodificada e a *string* original pode

ser obtida com ajuda desta função. Esta função funciona com base nos parâmetros especificados que são o ato de codificar e o erro. Existem vários tipos de codificações, como base 64, *ascii*, *gbk*, *hz*, *iso2022_kr*, *utf_32*, *utf_16* e muito mais. A *string* é descodificada com base na codificação utilizada. Durante esse processo, diferentes fases de tratamento de erros podem ser definidas através do parâmetro de erros. Por fim, este método devolve a *string* descodificada.

Descrição:

Código importante para criar informações sobre a execução de processos, permitindo realizar alterações e tomadas de decisões por parte dos utilizadores, possibilitando em casos menos comuns, lidar com funções de conveniência.

***Import packages* (relacionados com GRASS GIS) grst_to_rst dentro do código match_cellsize_and_clip**

```
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

"*Raster* da sessão GRASS GIS para *raster* fora da sessão"

```
def grs_to_rst(grsRst, rst, as_cmd=None, allBands=None, is_int=None):
    """
    GRASS Raster to Raster
    """

    from glass.prop  import grs_rst_drv
    from glass.pys.oss import fprop

    rstDrv = grs_rst_drv()
    rstExt = fprop(rst, 'ff')

    predictor = "2" if is_int else "3"

    if not as_cmd:
        from grass.pygrass.modules import Module

        m = Module(
            "r.out.gdal", input=grsRst, output=rst,
            format=rstDrv[rstExt], flags='c' if not allBands else '',
            createopt="INTERLEAVE=PIXEL,TFW=YES" if allBands else \
                f'COMPRESS=LZW,PREDICTOR={predictor},BIGTIFF=YES',
            overwrite=True, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        if not allBands:
            opt = f"createopt=\"COMPRESS=LZW,PREDICTOR={predictor},BIGTIFF=YES\""
        else:
            opt = "createopt=\"INTERLEAVE=PIXEL,TFW=YES\""

        rcmd = execmd((
            f"r.out.gdal input={grsRst} output={rst} "
            f"format={rstDrv[rstExt]} {opt} -c --overwrite --quiet"
        ))

    return rst
```

Métodos e Ferramentas:

```
    if not as_cmd:
        from grass.pygrass.modules import Module
```

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

```
        m = Module(
            "r.out.gdal", input=grsRst, output=rst,
            format=rstDrv[rstExt], flags='c' if not allBands else '',
            createopt="INTERLEAVE=PIXEL,TFW=YES" if allBands else \
                f'COMPRESS=LZW,PREDICTOR={predictor},BIGTIFF=YES',
            overwrite=True, run_=False, quiet=True
```

- "*r.out.gdal*": exporta ficheiros *rasters* do GRASS para formatos comuns fora da sessão.

Descrição:

Código importante para correr após os processos aplicados na sessão GRASS. É fundamental para exportar os dados (*rasters*) para fora da sessão e utilizar esses *outputs* para outros procederes.

Imports dentro do grst_to_rst:

```
def grs_to_rst(grsRst, rst, as_cmd=None, allBands=None, is_int=None):
    """
    GRASS Raster to Raster
    """

    from glass.prop  import grs_rst_drv
    from glass.pys.oss import fprop

    rstDrv = grs_rst_drv()
    rstExt = fprop(rst, 'ff')

    predictor = "2" if is_int else "3"

    if not as_cmd:
        from grass.pygrass.modules import Module

        m = Module(
            "r.out.gdal", input=grsRst, output=rst,
            format=rstDrv[rstExt], flags='c' if not allBands else '',
            createopt="INTERLEAVE=PIXEL,TFW=YES" if allBands else \
                f'COMPRESS=LZW,PREDICTOR={predictor},BIGTIFF=YES',
            overwrite=True, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        if not allBands:
            opt = f"createopt=\"COMPRESS=LZW,PREDICTOR={predictor},BIGTIFF=YES\""
        else:
            opt = "createopt=\"INTERLEAVE=PIXEL,TFW=YES\""

        rcmd = execmd((
            f"r.out.gdal input={grsRst} output={rst} "
            f"format={rstDrv[rstExt]} {opt} -c --overwrite --quiet"
        ))

    return rst
```

***Import* grs_rst_drv dentro do código grs_to_rst**

```
def grs_rst_drv():
    return {
        '.tif': 'GTiff',
        '.img': 'HFA'
    }
```

Descrição:

Este código tem a utilidade de dar ao *script* as diferentes extensões existentes para os *rasters* que serão extraídos da sessão GRASS.

Import fprop dentro do código grs_to_rst

```
def fprop(__file, prop, forceLower=None, fs_unit=None):
    """
    Return some property of file

    prop options:
    * filename or fn - return filename
    """

    from glass.pys import obj_to_lst

    prop = obj_to_lst(prop)

    result = {}

    if 'filename' in prop or 'fn' in prop:
        fn, ff = os.path.splitext(os.path.basename(__file))

        result['filename'] = fn if not forceLower else fn.lower()

        if 'fileformat' in prop or 'fn' in prop:
            result['fileformat'] = ff

    elif 'fileformat' in prop or 'ff' in prop:
        result['fileformat'] = os.path.splitext(__file)[1]

    if 'filesize' in prop or 'fs' in prop:
        fs_unit = 'MB' if not fs_unit else fs_unit

        fs = os.path.getsize(__file)

        if fs_unit == 'MB':
            fs  = (fs / 1024.0) / 1024

        elif fs_unit == 'KB':
            fs = fs / 1024.0

```

(Já explicados: *Import* fprop dentro do código is_rst)

Import execmd dentro do código grs_to_rst

```
def execmd(cmd):
    """
    Execute a command and provide information about the results
    """
    import subprocess

    p = subprocess.Popen(cmd, shell=True,
                         stdout=subprocess.PIPE, stderr=subprocess.PIPE)

    out, err = p.communicate()

    if p.returncode != 0:
        raise ValueError((
            'Message: Command execution ended with error\n'
            f'Command was: {cmd}\n'
            f'Output: {out.decode("utf-8")}\n'
            f'Error: {err.decode("utf-8")}'
        ))

    else:
        return out.decode('utf-8')
```

(Já explicados: *Import* execmd dentro do código rst_to_grs)

Import packages (relacionados com GRASS GIS) grst_to_mask dentro do código match_cellsize_and_clip

```
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

"*Raster* do GRASS GIS para máscara"

```
def grs_to_mask(inRst, overwrite=None):
    """
    Grass Raster to Mask
    """

    ow = True if overwrite else False

    from grass.pygrass.modules import Module

    m = Module(
        'r.mask', raster=inRst,
        overwrite=ow, quiet=True, run_=False
    )

    m()
```

Métodos e Ferramentas:

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*r.mask*": cria uma máscara para limitar operações de ficheiros *raster*.

Descrição:

Código importante para a criação de uma máscara a partir de um *raster* que está dentro da sessão GRASS. Esta máscara é importante para a posterior operação de recorte das diferentes bandas que compõem a imagem Sentinel-2.

***Import packages* (relacionados com GRASS GIS) rst_to_region dentro do código match_cellsize_and_clip**

```
47      Import packages related with GRASS GIS
48      """
49      from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
50      from glass.wenv.grs import rst_to_region
51      from glass.it.shp   import shp_to_grs
52      from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

"*Raster* para a região (área de estudo)"

```
248 def rst_to_region(__raster):
249     from grass.pygrass.modules import Module
250 
251     r = Module(
252         "g.region", raster=__raster, run_=False, quiet=True
253     )
254 
255     r()
```

Métodos e Ferramentas:

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*g.region*": Gere as definições de limite para a região geográfica.

Descrição:

Pequeno *script* que auxilia na sobreposição ou posicionamento preciso dos *rasters* utilizados em relação à área de estudo.

***Import packages* (relacionados com GRASS GIS) shp_to_grs dentro do código match_cellsize_and_clip**

```
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

"Adiciona um dado *shape* para a sessão GRASS"

```
def shp_to_grs(ilyr, olyr, filterByReg=None, asCMD=None):
    """
    Add Shape to GRASS GIS
    """

    if not asCMD:
        from grass.pygrass.modules import Module

        f = 'o' if not filterByReg else 'ro'

        m = Module(
            "v.in.ogr", input=ilyr, output=olyr, flags=f,
            overwrite=True, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        f = " -r" if filterByReg else ""

        rcmd = execmd((
            f"v.in.ogr input={ilyr} output={olyr} -o{f} "
            "--overwrite --quiet"
        ))

    return olyr
```

Métodos e Ferramentas:

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*v.in.ogr*": Importa dados vetoriais para uma sessão GRASS.

Descrição:

Pequeno código de grande importância para realizar a importação de dados vetoriais para a sessão GRASS GIS.

Imports dentro do shp_to_grs:

```
194 def shp_to_grs(ilyr, olyr, filterByReg=None, asCMD=None):
195     """
196     Add Shape to GRASS GIS
197     """
198 
199     if not asCMD:
200         from grass.pygrass.modules import Module
201 
202         f = 'o' if not filterByReg else 'ro'
203 
204         m = Module(
205             "v.in.ogr", input=ilyr, output=olyr, flags=f,
206             overwrite=True, run_=False, quiet=True
207         )
208 
209         m()
210 
211     else:
212         from glass.pys import execmd
213 
214         f = " -r" if filterByReg else ""
215 
216         rcmd = execmd((
217             f"v.in.ogr input={ilyr} output={olyr} -o{f} "
218             "--overwrite --quiet"
219         ))
220 
221     return olyr
```

Import execmd dentro do código shp_to_grs

```
27 def execmd(cmd):
28     """
29     Execute a command and provide information about the results
30     """
31     import subprocess
32 
33     p = subprocess.Popen(cmd, shell=True,
34                          stdout=subprocess.PIPE, stderr=subprocess.PIPE)
35 
36     out, err = p.communicate()
37 
38     if p.returncode != 0:
39         raise ValueError((
40             'Message: Command execution ended with error\n'
41             f'Command was: {cmd}\n'
42             f'Output: {out.decode("utf-8")}\n'
43             f'Error: {err.decode("utf-8")}'
44         ))
45 
46     else:
47         return out.decode('utf-8')
```

(Já explicados: *Import* execmd dentro do código rst_to_grs)

***Import packages* (relacionados com GRASS GIS) grsshp_to_grsrst dentro do código match_cellsize_and_clip**

```
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

"Vetor GRASS para *raster* GRASS"

"api: - PyGRASS; - GRASS"

"Geometria vetorial para *raster*"

"Se o *source* for *none*, a conversão será baseada no campo *cat*"

"Se o *source* for uma *string*, a conversão será baseada no campo com o nome igual a uma *string*"

"Se o *source* for um valor numérico, todas as células do *raster* de saída terão esse valor numérico"

```
def grsshp_to_grsrst(inshp, src, outrst, cmd=None):
    """
    GRASS Vector to GRASS Raster

    api:
    * pygrass
    * grass

    Vectorial geometry to raster

    If source is None, the convertion will be based on the cat field.

    If source is a string, the convertion will be based on the field
    with a name equal to the given string.

    If source is a numeric value, all cells of the output raster will have
    that same value.
    """

    __USE = "cat" if not src else "attr" if type(src) == str \
        else "val" if type(src) == int or \
        type(src) == float else None

    if not __USE:
        raise ValueError('\'source\' parameter value is not valid')

    if not cmd:
        from grass.pygrass.modules import Module

        m = Module(
            "v.to.rast", input=inshp, output=outrst, use=__USE,
            attribute_column=src if __USE == "attr" else None,
            value=src if __USE == "val" else None,
            overwrite=True, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        rcmd = execmd((
            "v.to.rast input={} output={} use={}{} "
            "--overwrite --quiet"
        ).format(
            inshp, outrst, __USE,
            "" if __USE == "cat" else f" attribute_column={src}" \
                if __USE == "attr" else f" val={src}"
        ))

    return outrst
```

Métodos e Ferramentas:

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*v.to.rast*": Converte um ficheiro vetorial para *raster*.

Descrição:

Código mais extenso e complexo do que os anteriores, no entanto percebe-se que a ferramenta fundamental utilizada neste *script* é "*v.to.rast*", onde o principal objetivo é realizar a conversão de um dado vetorial para um dado *raster* dentro da sessão GRASS GIS.

Imports dentro do shp_to_grs:

```
def grsshp_to_grsrst(inshp, src, outrst, cmd=None):
    """
    GRASS Vector to GRASS Raster

    api:
    * pygrass
    * grass

    Vectorial geometry to raster

    If source is None, the convertion will be based on the cat field.

    If source is a string, the convertion will be based on the field
    with a name equal to the given string.

    If source is a numeric value, all cells of the output raster will have
    that same value.
    """

    __USE = "cat" if not src else "attr" if type(src) == str \
        else "val" if type(src) == int or \
        type(src) == float else None

    if not __USE:
        raise ValueError('\'source\' parameter value is not valid')

    if not cmd:
        from grass.pygrass.modules import Module

        m = Module(
            "v.to.rast", input=inshp, output=outrst, use=__USE,
            attribute_column=src if __USE == "attr" else None,
            value=src if __USE == "val" else None,
            overwrite=True, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        rcmd = execmd((
            "v.to.rast input={} output={} use={}{} "
            "--overwrite --quiet"
        ).format(
            inshp, outrst, __USE,
            "" if __USE == "cat" else f" attribute_column={src}" \
                if __USE == "attr" else f" val={src}"
        ))

    return outrst
```

Import execmd dentro do código grsshp_to_grsrst

```
def execmd(cmd):
    """
    Execute a command and provide information about the results
    """
    import subprocess

    p = subprocess.Popen(cmd, shell=True,
                         stdout=subprocess.PIPE, stderr=subprocess.PIPE)

    out, err = p.communicate()

    if p.returncode != 0:
        raise ValueError((
            'Message: Command execution ended with error\n'
            f'Command was: {cmd}\n'
            f'Output: {out.decode("utf-8")}\n'
            f'Error: {err.decode("utf-8")}'
        ))

    else:
        return out.decode('utf-8')
```

(Já explicados: *Import* execmd dentro do código rst_to_grs)

Import packages (relacionados com GRASS GIS) shp_to_rst dentro do código match_cellsize_and_clip

```
    Import packages related with GRASS GIS
    """
    from glass.it.rst   import rst_to_grs, grs_to_rst, grs_to_mask
    from glass.wenv.grs import rst_to_region
    from glass.it.shp   import shp_to_grs
    from glass.dtr.torst import grsshp_to_grsrst as shp_to_rst
```

"*Feature class* para *raster*".

"*Cellcize* vai ser ignorado se *rst_template* for definido".

"API'S acedidos: - GDAL; - PyGRASS; - Grass".

"Usa-se o GRASS: - Iniciar sessão; - Importar dados; - Converter; - Exportar".

```
def shp_to_rst(shp, inSource, cellsize, nodata, outRaster, epsg=None,
               rst_template=None, snapRst=None, api='gdal'):
    """
    Feature Class to Raster

    cellsize will be ignored if rst_template is defined

    * API's Available:
    - gdal;
    - pygrass;
    - grass;
    """

    if api == 'gdal':
        from osgeo         import gdal, ogr
        from glass.prop import drv_name

        if not epsg:
            from glass.prop.prj import get_shp_sref

            srs = get_shp_sref(shp).ExportToWkt()
        else:
            from glass.prop.prj import epsg_to_wkt
            srs = epsg_to_wkt(epsg)

        # Get Extent
        dtShp = ogr.GetDriverByName(
            drv_name(shp)).Open(shp, 0)

        lyr = dtShp.GetLayer()

        if not rst_template:
            if not snapRst:
                x_min, x_max, y_min, y_max = lyr.GetExtent()
                x_res = int((x_max - x_min) / cellsize)
                y_res = int((y_max - y_min) / cellsize)

            else:
                from glass.prop.rst import adjust_ext_to_snap

                x_min, y_max, y_res, x_res, cellsize = adjust_ext_to_snap(
                    shp, snapRst
                )

        else:
            from glass.rd.rst import rst_to_array

            img_temp = gdal.Open(rst_template)
            geo_transform = img_temp.GetGeoTransform()

            y_res, x_res = rst_to_array(rst_template).shape

        # Create output
        dtRst = gdal.GetDriverByName(drv_name(outRaster)).Create(
            outRaster, x_res, y_res, gdal.GDT_Byte
        )

        if not rst_template:
            dtRst.SetGeoTransform((x_min, cellsize, 0, y_max, 0, -cellsize))

        else:
            dtRst.SetGeoTransform(geo_transform)

        dtRst.SetProjection(str(srs))

        bnd = dtRst.GetRasterBand(1)
        bnd.SetNoDataValue(nodata)

        gdal.RasterizeLayer(dtRst, [1], lyr, burn_values=[1])

        del lyr
        dtShp.Destroy()

    elif api == 'grass' or api == 'pygrass':
        """
        Use GRASS GIS
        - Start Session
        - Import data
        - Convert
        - Export
        """

        import os
        from glass.pys.oss    import fprop
        from glass.wenv.grs import run_grass
        from glass.prop.prj import get_epsg

        # Create GRASS GIS Session
        ws = os.path.dirname(outRaster)
        loc = fprop(outRaster, 'fn')
        epsg = get_epsg(shp)

        gbase = run_grass(ws, location=loc, srs=epsg)

        import grass.script.setup as gsetup

        gsetup.init(gbase, ws, loc, 'PERMANENT')

        # Import Packages
        from glass.it.shp   import shp_to_grs
        from glass.it.rst   import grs_to_rst
        from glass.wenv.grs import shp_to_region
```

(Parte final do *script* def shp_to_rst ())

```
164
165         # Shape to GRASS GIS
166         gshp = shp_to_grs(shp, fprop(shp, 'fn'), asCMD=True)
167
168         # Set Region
169         shp_to_region(gshp, cellsize)
170
171         # Convert
172         grst = grsshp_to_grsrst(gshp, inSource, gshp+'__rst', api="grass")
173
174         # Export
175         grs_to_rst(grst, outRaster, as_cmd=True)
176
177     else:
178         raise ValueError('API {} is not available'.format(api))
179
180     return outRaster
```

Métodos e Ferramentas:

```
79          if not epsg:
80              from glass.prop.prj import get_shp_sref
81
82              srs = get_shp_sref(shp).ExportToWkt()
```

- *ExportTowkt*: método utilizado para converter o *SpatialReference* (SRS) no formato de texto conhecido WKT (*Well Know Text*). O WKT é uma maneira prática de entender e aceder aos fundamentos da representação vetorial com a linguagem WKT. Esta linguagem não é de programação, mas sim uma linguagem de marcação, ou *markup*.

#Obter a extensão

```
87          # Get Extent
88          dtShp = ogr.GetDriverByName(
89              drv_name(shp)).Open(shp, 0)
90
91          lyr = dtShp.GetLayer()
92
93          if not rst_template:
94              if not snapRst:
95                  x_min, x_max, y_min, y_max = lyr.GetExtent()
96                  x_res = int((x_max - x_min) / cellsize)
97                  y_res = int((y_max - y_min) / cellsize)
```

- *GetDriverbyName*: utilizado para procurar o driver com base no nome curto. Parâmetros: *pszName* – o nome curto com *GTiff*, que está a ser procurado. Devoluções: identificação do nome ou *Null* se nenhuma correspondência for encontrada.

- *gdal.Open*: função utilizada para abrir um *dataset*.

- *Getlayer*: utilizado para selecionar as camadas individuais de um *dataset*.

- *GetExtent*: serve para procurar a extensão da camada em questão. Dependendo dos *drivers*, a extensão devolvida pode ou não ter em consideração o filtro espacial. Neste sentido é mais seguro chamar a função *GetExtent ()* sem definir um filtro espacial.

```
106         else:
107             from glass.rd.rst import rst_to_array
108 
109             img_temp = gdal.Open(rst_template)
110             geo_transform = img_temp.GetGeoTransform()
111 
112             y_res, x_res = rst_to_array(rst_template).shape
113 
```

- *GetgeoTransform*: utilizado para obter os coeficientes de transformação. Obtém os coeficientes para transformar a área *raster* pixel/linha (P, L) e a área de coordenadas de projeção (Xp, Yp).

- *shape*: as matrizes *numpy* têm um atributo chamado *shape* que geralmente é utilizado para obter a área de uma matriz, mas também devolve uma tupla que pode ser usada para alterar as dimensões da matriz.

#Criação do *output*

```
114         # Create output
115         dtRst = gdal.GetDriverByName(drv_name(outRaster)).Create(
116             outRaster, x_res, y_res, gdal.GDT_Byte
117         )
118 
119         if not rst_template:
120             dtRst.SetGeoTransform((x_min, cellsize, 0, y_max, 0, -cellsize))
121 
122         else:
123             dtRst.SetGeoTransform(geo_transform)
124 
125         dtRst.SetProjection(str(srs))
126 
127         bnd = dtRst.GetRasterBand(1)
128         bnd.SetNoDataValue(nodata)
129 
130         gdal.RasterizeLayer(dtRst, [1], lyr, burn_values=[1])
131 
132         del lyr
133         dtShp.Destroy()
```

- *Create*: o método *create* permite gravar explicitamente todos os metadados e dados *raster*. É importante e oferece suporte à criação de novos ficheiros.

- *gdal.GDT_Byte*: o *pData* é o formato de memória no qual os dados são lidos ou gravados. O seu tipo real deve ser passado função do *eBufType*, como *GDT_Float32* ou *GDT_Byte*.

- *SetGeoTransform*: define os parâmetros de transformação para a fonte de dados GDAL.

- SetProjection: define a projeção da fonte de dados.

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *SetNoDataValue*: define o valor *nodata* como duplo, *uint16* e *int16*. Se o tipo de dados natural do atributo/matriz não for um dos três referidos, a conversão de tipo será em função do tipo definido por *GetDataType ()*.

- *RasterizeLayer*: serve para converter um polígono para um *raster type*.

- *Destroy*: serve para destruir objeto e neste caso, é utilizado para eliminar a *shapefile*.

#Inicio/ criação de uma sessão GRASS GIS

```
149         # Create GRASS GIS Session
150         ws = os.path.dirname(outRaster)
151         loc = fprop(outRaster, 'fn')
152         epsg = get_epsg(shp)
153 
154         gbase = run_grass(ws, location=loc, srs=epsg)
155 
156         import grass.script.setup as gsetup
157 
158         gsetup.init(gbase, ws, loc, 'PERMANENT')
```

- *os.path.dirname*: método em *python* utilizado para obter o nome especifico da pasta do caminho.

- *import.grass.script.setup as gsetup*: importa algumas partes convenientes do GRASS GIS a partir da API *python*.

- *gsetup.init*: importante para iniciar a sessão do GRASS GIS, é esta linha que faz com que a sessão se inicie.

#Exportar

```
174         # Export
175         grs_to_rst(grst, outRaster, as_cmd=True)
176 
177     else:
178         raise ValueError('API {} is not available'.format(api))
179 
180     return outRaster
```

- *format*: método que formata o valor detalhado e insere-o dentro do espaço reservado da *string*. O método devolve a *string* formatada.

Descrição:

Código que implementa dentro de si todos os pequenos códigos/ *scripts* descritos anteriormente com apenas uma função. Neste código, esse conjunto de códigos que implementam apenas uma função, vão fazer com que tudo corra para um resultado maior. O principal objetivo deste *script* é realizar a conversão de um dado *sapefile* para um formato *raster*, e para isso são corridos em conjuntos todos os *scripts*. O código acaba por ser muito mais complexo, no entanto se subdividir em partes é possível retirar dele toda a informação necessária para entendê-lo. Para começar, é necessário chegar à extensão da área do *shapefile* para puder transferir essa informação para o *raster* que será criado. Depois de criar e direcionar um caminho para o *output* do *raster* final, deve-se definir o sistema de projeção do ficheiro final. Iniciar uma sessão no GRASS GIS para a implementação de ferramentas como, importação da *shapefile* para a sessão do GRASS, definir a região e a extensão dessa *shapefile*, converter dentro da sessão GRASS a *shapefile* para *raster* e por fim realizar o *export* do *raster* final para fora da sessão GRASS.

Imports dentro do shp_to_rst:

```
75      if api == 'gdal':
76          from osgeo        import gdal, ogr
77          from glass.prop import drv_name
78
79          if not epsg:
80              from glass.prop.prj import get_shp_sref
81
82              srs = get_shp_sref(shp).ExportToWkt()
83          else:
84              from glass.prop.prj import epsg_to_wkt
85              srs = epsg_to_wkt(epsg)
86
87          # Get Extent
88          dtShp = ogr.GetDriverByName(
89              drv_name(shp)).Open(shp, 0)
90
91          lyr = dtShp.GetLayer()
92
93          if not rst_template:
94              if not snapRst:
95                  x_min, x_max, y_min, y_max = lyr.GetExtent()
96                  x_res = int((x_max - x_min) / cellsize)
97                  y_res = int((y_max - y_min) / cellsize)
98
99              else:
100                 from glass.prop.rst import adjust_ext_to_snap
101
102                 x_min, y_max, y_res, x_res, cellsize = adjust_ext_to_snap(
103                     shp, snapRst
104                 )
105
106         else:
107             from glass.rd.rst import rst_to_array
```

```
    elif api == 'grass' or api == 'pygrass':
        """
        Use GRASS GIS
        - Start Session
        - Import data
        - Convert
        - Export
        """

        import os
        from glass.pys.oss    import fprop
        from glass.wenv.grs import run_grass
        from glass.prop.prj import get_epsg
```

Imports packages relacionados com a sessão GRASS GIS:

```
        # Import Packages
        from glass.it.shp   import shp_to_grs
        from glass.it.rst   import grs_to_rst
        from glass.wenv.grs import shp_to_region
```

Import drv_name dentro do código shp_to_rst

```
    if api == 'gdal':
        from osgeo        import gdal, ogr
        from glass.prop import drv_name
```

"Devolve o formato de um ficheiro"

```
def drv_name(_file):
    """
    Return the driver for a given file format
    """

    import os

    drv = {
        # Vector files
        '.gml'    : 'GML',
        '.shp'    : 'ESRI Shapefile',
        '.json'   : 'GeoJSON',
        '.kml'    : 'KML',
        '.osm'    : 'OSM',
        '.dbf'    : 'ESRI Shapefile',
        '.vct'    : 'Idrisi',
        '.nc'     : 'netCDF',
        '.vrt'    : 'VRT',
        '.mem'    : 'MEMORY',
        '.sqlite' : 'SQLite',
        '.gdb'    : 'FileGDB',
        # Raster files
        '.tif'    : 'GTiff',
        '.ecw'    : 'ECW',
        '.mpr'    : 'ILWIS',
        '.mpl'    : 'ILWIS',
        '.jpg'    : 'JPEG',
        '.nc'     : 'netCDF',
        '.png'    : 'PNG',
        '.vrt'    : 'VRT',
        '.asc'    : 'AAIGrid',
        '.img'    : 'HFA',
        # Vector or Raster
        '.gpkg'   : 'GPKG',
        '.xlsx'   : 'XLSX'
    }
    return str(drv[os.path.splitext(_file)[1]])
```

Métodos e Ferramentas:

- *Import* OS: o módulo OS em *python* fornece funções para interagir com o sistema operacional. OS, vem nos módulos padrão do *python*. Este módulo fornece uma maneira flexível de usar a funcionalidade dependendo do sistema operacional. Os módulos "*OS*" e "*os.path*" incluem muitas funções para interagir com o sistema de ficheiros e pastas.

- *os.path.splitext*: o método *OS.path* é um submódulo do módulo OS em *python* utilizado para manipular nomes dos caminhos. *Os.path.splitext()* é o método usado em *python* para dividir o nome do caminho em um par *root* e *ext*. Aqui, *ext* significa extensão e possui a parte da extensão do caminho especificado, enquanto *root* é tudo, exceto a parte *ext*.

Descrição:

Pequeno código, mas importante para dar a conhecer ao *script* todos as extensões existentes dos diferentes formatos de dados vetoriais, dos *rasters* e dos vetoriais/ *rasters*. Este código devolve todos as extensões dos ficheiros em forma de *string*.

Import get_shp_sref dentro do código shp_to_rst

```
79          if not epsg:
80              from glass.prop.prj import get_shp_sref
```

"Retira sistema de referência de um objeto de uma *feature class*/ *layer*"

```
37  def get_shp_sref(shp):
38      """
39      Get Spatial Reference Object from Feature Class/Lyr
40      """
41
42      from osgeo        import ogr
43      from glass.prop import drv_name
44
45      if type(shp) == ogr.Layer:
46          lyr = shp
47
48          c = 0
49
50      else:
51          data = ogr.GetDriverByName(
52              drv_name(shp)).Open(shp)
53
54          lyr = data.GetLayer()
55          c = 1
56
57      spref = lyr.GetSpatialRef()
58
59      if c:
60          del lyr
61          data.Destroy()
62
63      return spref
```

Métodos e Ferramentas:

- *ogr.Layer*: método utilizado para chamar o ficheiro e aplicar alterações.

- *GetDriverbyName*: utilizado para procurar o driver com base no nome curto. Parâmetros: *pszName* – o nome curto com *GTiff*, que está a ser procurado. Devoluções: identificação do nome ou *Null* se não for encontrada nenhuma correspondência.

- *gdal.Open*: função utilizada para abrir um *dataset*.

- *Getlayer*: utilizado para selecionar as camadas individuais de um *dataset*.

- *GetSpatialRef*: vai buscar o sistema de referência espacial para esta camada. O objeto devolvido pertence ao *ogrLayer* e não deve ser alterado.

- *Destroy*: serve para destruir objetos.

Descrição:

Código extremamente importante para adquirir o sistema de referência ou a projeção de um determinado ficheiro. O sistema de referência é retirado do ficheiro, guardado e exportado para puder ser utilizado e aplicado a diferentes dados.

Imports dentro do get_shp_sref:

```
37 def get_shp_sref(shp):
38     """
39     Get Spatial Reference Object from Feature Class/Lyr
40     """
41 
42     from osgeo        import ogr
43     from glass.prop import drv_name
```

Import drv_name dentro do código get_shp_sref

```
50 def drv_name(_file):
51     """
52     Return the driver for a given file format
53     """
54 
55     import os
56 
57     drv = {
58         # Vector files
59         '.gml'    : 'GML',
60         '.shp'    : 'ESRI Shapefile',
61         '.json'   : 'GeoJSON',
62         '.kml'    : 'KML',
63         '.osm'    : 'OSM',
64         '.dbf'    : 'ESRI Shapefile',
65         '.vct'    : 'Idrisi',
66         '.nc'     : 'netCDF',
67         '.vrt'    : 'VRT',
68         '.mem'    : 'MEMORY',
69         '.sqlite' : 'SQLite',
70         '.gdb'    : 'FileGDB',
71         # Raster files
72         '.tif'    : 'GTiff',
73         '.ecw'    : 'ECW',
74         '.mpr'    : 'ILWIS',
75         '.mpl'    : 'ILWIS',
76         '.jpg'    : 'JPEG',
77         '.nc'     : 'netCDF',
78         '.png'    : 'PNG',
79         '.vrt'    : 'VRT',
80         '.asc'    : 'AAIGrid',
81         '.img'    : 'HFA',
82         # Vector or Raster
83         '.gpkg'   : 'GPKG',
84         '.xlsx'   : 'XLSX'
85     }
86     return str(drv[os.path.splitext(_file)[1]])
```

(Já explicados: *Import* drv_name dentro do código shp_to_rst)

Import epsg_to_wkt dentro do código shp_to_rst

```
83         else:
84             from glass.prop.prj import epsg_to_wkt
85             srs = epsg_to_wkt(epsg)
```

```
23 def epsg_to_wkt(epsg):
24     s = osr.SpatialReference()
25     s.ImportFromEPSG(epsg)
26 
27     return s.ExportToWkt()
```

Métodos e Ferramentas:

- *osr.SpatialReference*: serve para indicar que se vai chamar o sistema de referência.

- *ImportFromEPSG*: inicia o SRS com base no código CRS geográfico, projetado ou vertical do EPSG. Este método inicia a referência espacial com base no código EPSG CRS passado no conjunto de dados PROJ.

- *ExportToWkt*: converto o código SRS para o formato WKT.

Descrição:

Código importante por convocar o código do sistema de referência retirado do ficheiro e por convertê-lo para o formato WKT, necessário para a leitura do *script*.

Import adjust_ext_to_snap dentro do código shp_to_rst

```
99              else:
100                 from glass.prop.rst import adjust_ext_to_snap
101
```

"Ajusta a extensão de saída de um *raster* para igualar à área de outro *raster*"

```
369 def adjust_ext_to_snap(outExt, snapRst):
370     """
371     Adjust extent for a output raster to snap with other raster
372     """
373
374     from glass.prop     import is_shp, is_rst
375     from glass.prop.rst import rst_ext, get_cellsize
376     from glass.gobj     import new_pnt, create_polygon
377
378     # Check if outExt is a raster or not
379     isRst = is_rst(outExt)
380
381     if isRst:
382         shpAExt = rst_ext(outExt)
383
384     else:
385         isShp = is_shp(outExt)
386
387         if isShp:
388             from glass.prop.feat import get_ext
389
390             shpAExt = get_ext(outExt)
391
392         else:
393             raise ValueError((
394                 "outExt value should be a path to a SHP or to a Raster file"
395             ))
396
397     # Check if snapRst is a raster
398     isRst = is_rst(snapRst)
399
400     if not isRst:
401         raise ValueError((
402             "snapRst should be a path to a raster file"
403         ))
404
```

(Continuação do código anterior)

```
    # Get snapRst Extent
    snapRstExt = rst_ext(snapRst)

    # Get cellsize
    csize = get_cellsize(snapRst)

    # Find extent point of outExt inside the two extents
    # This will be used as pseudo origin

    snapRstPnt = [
        new_pnt(snapRstExt[0], snapRstExt[3]),
        new_pnt(snapRstExt[1], snapRstExt[3]),
        new_pnt(snapRstExt[1], snapRstExt[2]),
        new_pnt(snapRstExt[0], snapRstExt[2]),
        new_pnt(snapRstExt[0], snapRstExt[3]),
    ]

    poly_snap_rst = create_polygon(snapRstPnt)

    outExtPnt = {
        'top_left'     : new_pnt(shpAExt[0], shpAExt[3]),
        'top_right'    : new_pnt(shpAExt[1], shpAExt[3]),
        'bottom_right' : new_pnt(shpAExt[1], shpAExt[2]),
        'bottom_left'  : new_pnt(shpAExt[0], shpAExt[2])
    }

    out_rst_pseudo = {}
    for pnt in outExtPnt:
        out_rst_pseudo[pnt] = outExtPnt[pnt].Intersects(poly_snap_rst)

    pseudoOrigin = outExtPnt['top_left'] if out_rst_pseudo['top_left'] else \
        outExtPnt['bottom_left'] if out_rst_pseudo['bottom_left'] else \
        outExtPnt['top_right'] if out_rst_pseudo['top_right'] else \
        outExtPnt['bottom_right'] if out_rst_pseudo['bottom_right'] else None
    if not pseudoOrigin:
        raise ValueError((
            'Extents doesn\'t have overlapping areas'
        ))

    pseudoOriginName = 'top_left' if out_rst_pseudo['top_left'] else \
        'bottom_left' if out_rst_pseudo['bottom_left'] else \
        'top_right' if out_rst_pseudo['top_right'] else \
        'bottom_right' if out_rst_pseudo['bottom_right'] else None

    # Get out Raster Shape
    n_col = int((shpAExt[1] - shpAExt[0]) / csize)
    n_row = int((shpAExt[3] - shpAExt[2]) / csize)

    # Get Output Raster real origin/top left
    yName, xName = pseudoOriginName.split('_')

    if xName == 'left':
        # Obtain left of output Raster
        left_out_rst = snapRstExt[0] + (
            csize * int((shpAExt[0] - snapRstExt[0]) / csize))

    else:
        # obtain right of output Raster
        right_out_rst = snapRstExt[1] - (
            csize * int((snapRstExt[1] - shpAExt[1]) / csize))

        # Use right to obtain left coordinate
        left_out_rst = right_out_rst - (n_col * csize)

    if yName == 'top':
        # Obtain top of output Raster
        top_out_rst = snapRstExt[3] - (
            csize * int((snapRstExt[3] - shpAExt[3]) / csize))

    else:
        # obtain bottom of output raster
        bot_out_rst = snapRstExt[2] + (
            csize * int((shpAExt[2] - snapRstExt[2]) / csize))

        # use bottom to find the top of the output raster
        top_out_rst = bot_out_rst + (n_row * csize)

    return left_out_rst, top_out_rst, n_row, n_col, csize
```

Métodos e Ferramentas:

```
    out_rst_pseudo = {}
    for pnt in outExtPnt:
        out_rst_pseudo[pnt] = outExtPnt[pnt].Intersects(poly_snap_rst)
```

- *Intersects*: determina se duas geometrias cruzam.

Obtém a extensão real do *raster output*/ canto superior esquerdo

```
    # Get Output Raster real origin/top left
    yName, xName = pseudoOriginName.split('_')
```

- *split*: método que divide uma *string* em uma lista. Pode especificar o separador, como um separador padrão ou até mesmo um espaço em branco.

Descrição:

Código muito extenso, com a capacidade de desenvolver pontos muito importantes para o *script* final. Este código tem a tarefa de confirmar se o *output* que vai ser carregado no código é um ficheiro *raster* ou não e confirma se o *snap raster* é um ficheiro desse mesmo formato. Além disso, obtém a extensão do *snap raster*, adquire o *cellsize* do pixel, encontra o ponto dentro das duas extensões e usa-os como uma *pseudo-origem*. Obtém o formato do *raster* proveniente da *shape*, regista o valor verdadeiro do ponto do canto superior esquerdo do *raster output*, assim como o valor verdadeiro do ponto do canto superior direito. Utiliza o ponto superior esquerdo para obter as coordenadas, regista os valores da parte superior (topo) e do fundo do *raster output* e certifica-se de que o *snap raster* fica bem posicionado em relação a outro *raster*.

Imports dentro do adjust_ext_to_snap:

```
def adjust_ext_to_snap(outExt, snapRst):
    """
    Adjust extent for a output raster to snap with other raster
    """

    from glass.prop     import is_shp, is_rst
    from glass.prop.rst import rst_ext, get_cellsize
    from glass.gobj     import new_pnt, create_polygon

    # Check if outExt is a raster or not
    isRst = is_rst(outExt)

    if isRst:
        shpAExt = rst_ext(outExt)

    else:
        isShp = is_shp(outExt)

        if isShp:
            from glass.prop.feat import get_ext
```

Import is_shp dentro do código adjust_ext_to_snap

```
374     from glass.prop      import is_shp, is_rst
375     from glass.prop.rst import rst_ext, get_cellsize
376     from glass.gobj      import new_pnt, create_polygon
```

```
33 def is_shp(_file):
34     from glass.pys.oss import fprop
35 
36     lst = vector_formats()
37 
38     file_ext = fprop(_file, 'ff')
39 
40     if file_ext not in lst:
41         return None
42     else:
43         return True
```

Descrição:

Pequeno código que ajuda a identificar os ficheiros com um formato de dados vetorial, neste caso *shapefiles*. Se for uma *shapefile*, o *script* devolve o valor como verdadeiro, enquanto se não for devolve o valor de *None*.

Imports dentro do is_shp:

```
33 def is_shp(_file):
34     from glass.pys.oss import fprop
35 
36     lst = vector_formats()
37 
38     file_ext = fprop(_file, 'ff')
39 
40     if file_ext not in lst:
41         return None
42     else:
43         return True
```

Import fprop dentro do código is_shp

```
def fprop(__file, prop, forceLower=None, fs_unit=None):
    """
    Return some property of file

    prop options:
    * filename or fn - return filename
    """

    from glass.pys import obj_to_lst

    prop = obj_to_lst(prop)

    result = {}

    if 'filename' in prop or 'fn' in prop:
        fn, ff = os.path.splitext(os.path.basename(__file))

        result['filename'] = fn if not forceLower else fn.lower()

        if 'fileformat' in prop or 'fn' in prop:
            result['fileformat'] = ff

    elif 'fileformat' in prop or 'ff' in prop:
        result['fileformat'] = os.path.splitext(__file)[1]

    if 'filesize' in prop or 'fs' in prop:
        fs_unit = 'MB' if not fs_unit else fs_unit

        fs = os.path.getsize(__file)

        if fs_unit == 'MB':
            fs  = (fs / 1024.0) / 1024

        elif fs_unit == 'KB':
            fs = fs / 1024.0
```

(Já explicados: *Import* fprop dentro do código is_rst)

Import is_rst dentro do código adjust_ext_to_snap

```
        from glass.prop     import is_shp, is_rst
        from glass.prop.rst import rst_ext, get_cellsize
        from glass.gobj     import new_pnt, create_polygon
```

```
def is_rst(_file):
    from glass.pys.oss import fprop

    rst_lst = raster_formats()

    file_ext = fprop(_file, 'ff')

    if file_ext not in rst_lst:
        return None
    else:
        return True
```

Descrição:

Pequeno código que ajuda a identificar os ficheiros com um formato de dados matricial, neste caso *rasters*. Se for um *raster*, o Script devolve o valor como verdadeiro, enquanto se não for devolve o valor de *None*.

Imports dentro do is_rst:

```
20 def is_rst(_file):
21     from glass.pys.oss import fprop
22 
23     rst_lst = raster_formats()
24 
25     file_ext = fprop(_file, 'ff')
26 
27     if file_ext not in rst_lst:
28         return None
29     else:
30         return True
```

Import fprop dentro do código is_rst

```
25 def fprop(__file, prop, forceLower=None, fs_unit=None):
26     """
27     Return some property of file
28 
29     prop options:
30     * filename or fn - return filename
31     """
32 
33     from glass.pys import obj_to_lst
34 
35     prop = obj_to_lst(prop)
36 
37     result = {}
38 
39     if 'filename' in prop or 'fn' in prop:
40         fn, ff = os.path.splitext(os.path.basename(__file))
41 
42         result['filename'] = fn if not forceLower else fn.lower()
43 
44         if 'fileformat' in prop or 'fn' in prop:
45             result['fileformat'] = ff
46 
47     elif 'fileformat' in prop or 'ff' in prop:
48         result['fileformat'] = os.path.splitext(__file)[1]
49 
50     if 'filesize' in prop or 'fs' in prop:
51         fs_unit = 'MB' if not fs_unit else fs_unit
52 
53         fs = os.path.getsize(__file)
54 
55         if fs_unit == 'MB':
56             fs  = (fs / 1024.0) / 1024
57 
58         elif fs_unit == 'KB':
59             fs = fs / 1024.0
60 
```

(Já explicados: *Import* fprop dentro do código is_rst)

***Import* rst_ext dentro do código adjust_ext_to_snap**

```
374     from glass.prop     import is_shp, is_rst
375     from glass.prop.rst import rst_ext, get_cellsize
376     from glass.gobj     import new_pnt, create_polygon
```

"Devolve um *array* com a extensão do *raster dataset*"

"Ordem do *Array* = Xmin (*left*), Xmax (*right*), Ymin (*bottom*), Ymax (*top*)"

```
32  def rst_ext(rst):
33      """
34      Return a array with the extent of one raster dataset
35
36      array order = Xmin (left), XMax (right), YMin (bottom), YMax (top)
37      """
38
39      from osgeo import gdal
40
41      img = gdal.Open(rst)
42
43      lnhs = int(img.RasterYSize)
44      cols = int(img.RasterXSize)
45
46      left, cellx, z, top, c, celly = img.GetGeoTransform()
47
48      right  = left + (cols * cellx)
49      bottom = top  - (lnhs * abs(celly))
50
51      extent = [left, right, bottom, top]
52
53      return extent
```

Métodos e Ferramentas:

- *gdal.Open*: função utilizada para abrir um *dataset*.

- *RasterYSize*: utilizado para obter a altura da imagem *raster*.

- *RasterXSize*: utilizado para obter a largura da imagem *raster*.

- *GetgeoTransform*: utilizado para obter os coeficientes de transformação. Obtém os coeficientes para transformar a área *raster* pixel/linha (P, L) e a área de coordenadas de projeção (Xp, Yp).

Descrição:

Pequeno código que implementa ferramentas para adquirir o tamanho do *raster* num array, tendo em consideração Xmin (esquerdo), Xmax (esquerdo), Ymin (fundo) e Ymax (topo). A extensão do *raster* adquirido vai ter em consideração a recolha das coordenadas importantes, para a definição da sua área.

***Import* get_cellsize dentro do código adjust_ext_to_snap**

```
374     from glass.prop     import is_shp, is_rst
375     from glass.prop.rst import rst_ext, get_cellsize
376     from glass.gobj     import new_pnt, create_polygon
```

"Devolve o *cellsize* de um ou mais *rasters*"

"Em caso de grupos de *raster*, o resultado será:

D = {

'path_to_raster1': cellsize_raster_1,

'path_to_raster2': cellsize_raster_2,

'path_to_raster3': cellsize_raster_3,

…,

'path_to_rastern': cellsize_raster_n,

}"

"API's disponíveis: - GDAL; - PyGRASS"

```
def get_cellsize(rst, xy=False, bnd=None, gisApi='gdal'):
    """
    Return cellsize of one or more Raster Datasets

    In the case of groups, the result will be:
    d = {
        'path_to_raster1': cellsize_raster_1,
        'path_to_raster2': cellsize_raster_2,
        'path_to_raster3': cellsize_raster_3,
        ...,
        'path_to_rastern': cellsize_raster_n,
    }

    API'S Available:
    * gdal;
    * pygrass
    """

    import os

    if gisApi == 'gdal':
        from osgeo             import gdal
        from glass.prop.img import get_cell_size

        if type(rst) != list:
            if os.path.exists(rst) and os.path.isdir(rst):
                from glass.pys.oss import lst_ff
                from glass.prop  import raster_formats

                rsts = lst_ff(rst, file_format=raster_formats())

            elif os.path.exists(rst) and os.path.isfile(rst):
                rsts = [rst]
            else:
                raise ValueError((
                    'Invalid object rst. Please insert a path to a raster, '
                    'a path to a directory with rasters or a list with '
                    'rasters path.'
                ))

        else:
            rsts = rst

        cs = {}
        for r in rsts:
            imgsrc = gdal.Open(r)

            cs[r] = get_cell_size(
                imgsrc) if xy else get_cell_size(imgsrc)[0]

        return cs[rsts[0]] if len(rsts) == 1 else cs

    elif gisApi == 'qgis':
        from qgis.core import QgsRasterLayer

        rasterLyr = QgsRasterLayer(rst, "lyr")
        x = rasterLyr.rasterUnitsPerPixelX()

        if xy:
            y = rasterLyr.rasterUnitsPerPixelY()

            return x, y
        else:
            return x

    elif gisApi == 'pygrass':
        import grass.script as grass

        dic = grass.raster.raster info(rst)

        return dic['nsres']

    else:
        raise ValueError('The api {} is not available'.format(gisApi))
```

Métodos e Ferramentas:

```
        if type(rst) != list:
            if os.path.exists(rst) and os.path.isdir(rst):
                from glass.pys.oss import lst_ff
                from glass.prop  import raster_formats

                rsts = lst_ff(rst, file_format=raster_formats())

            elif os.path.exists(rst) and os.path.isfile(rst):
                rsts = [rst]
```

- *os.path.exists*: este método em *python* é usado para verificar se o caminho especificado existe. Este método também pode ser usado para verificar se o caminho fornecido de refere a uma pasta aberto ou não.

- *os.path.isdir*: método em *python* utilizado para verificar se o caminho especificado para uma pasta existe ou não. Este método segue um *link* representativo, ou seja, se o caminho indicado for um *link* para uma pasta que exista, o método devolve *true*.

```
        for r in rsts:
            imgsrc = gdal.Open(r)

            cs[r] = get_cell_size(
                imgsrc) if xy else get_cell_size(imgsrc)[0]

        return cs[rsts[0]] if len(rsts) == 1 else cs

    elif gisApi == 'qgis':
        from qgis.core import QgsRasterLayer

        rasterLyr = QgsRasterLayer(rst, "lyr")
        x = rasterLyr.rasterUnitsPerPixelX()

        if xy:
            y = rasterLyr.rasterUnitsPerPixelY()
```

- *gdal.Open*: função utilizada para abrir um *dataset*.

- *Qgis.Core*: biblioteca CORE que contém todas as funcionalidades básicas do GIS.

- *rasterUnitsPerPixelX*: devolve o número de unidades *raster* por cada pixel *raster* no eixo X.

- *rasterUnitsPerPixelY*: devolve o número de unidades *raster* por cada pixel *raster* no eixo Y.

```
    elif gisApi == 'pygrass':
        import grass.script as grass

        dic = grass.raster.raster_info(rst)

        return dic['nsres']

    else:
        raise ValueError('The api {} is not available'.format(gisApi))
```

- *grass.script*: importa o pacote na guia *python* na GUI do GRASS.

- *grass.raster.raster_info*: devolve informações básicas sobre a informação de um *raster*.

- *format*: método que formata os valores especificos e insere-os dentro do espaço reservado da *string*. Este método devolve a *string* formatada.

Descrição:

Script muito complexo e com o acesso fundamental a ferramentas QGIS e GRASS para devolver o *cellsize* de um *raster*. Para além de devolver o *cellsize* de um *raster*, consegue devolver de vários *rasters* em conjunto.

Imports dentro do get_cellsize:

```
98      if gisApi == 'gdal':
99          from osgeo            import gdal
100         from glass.prop.img import get_cell_size
101
102         if type(rst) != list:
103             if os.path.exists(rst) and os.path.isdir(rst):
104                 from glass.pys.oss import lst_ff
105                 from glass.prop  import raster_formats
```

***Import* get_cell_size dentro do código get_cellsize**

```
98      if gisApi == 'gdal':
99          from osgeo            import gdal
100         from glass.prop.img import get_cell_size
```

"Devolve o *cellsize*"

```
15  def get_cell_size(img):
16      """Return Cellsize"""
17
18      (tlx, x, xr, tly, yr, y) = img.GetGeoTransform()
19
20      return x, y
```

Métodos e Ferramentas:

- *GetgeoTransform*: utilizado para obter os coeficientes de transformação. Obtém os coeficientes para transformar a área *raster* pixel/linha (P, L) e a área de coordenadas de projeção (Xp, Yp).

Descrição:

Pequeno *script* que interpreta o *cellsize* de qualquer célula de um *raster* através do método *GetGeoTransform*. Este método ajuda a devolver a distância entre x e y de cada célula e assim perceber o seu tamanho auxiliado das coordenadas de projeção.

Import lst_ff dentro do código get_cellsize

```
102         if type(rst) != list:
103             if os.path.exists(rst) and os.path.isdir(rst):
104                 from glass.pys.oss import lst_ff
105                 from glass.prop  import raster_formats
```

```
76  def lst_ff(w, file_format=None, filename=None, fnpart=None, rfilename=None):
77      """
78      List the abs path of all files with a specific extension on a folder
79      """
80
81      from glass.pys import obj_to_lst
82
83      filename = None if filename and fnpart else filename
84
85      # Prepare file format list
86      if file_format:
87          formats = obj_to_lst(file_format)
88
89          for f in range(len(formats)):
90              if formats[f][0] != '.':
91                  formats[f] = '.' + formats[f]
92
93      # List files
94      r = []
95      for (d, _d_, f) in os.walk(w):
96          r.extend(f)
97          break
98
99      # Filter files by format or not
100     if not file_format:
101         if not rfilename:
102             t = [os.path.join(w, i) for i in r]
103         else:
104             t = [i for i in r]
105
```

(Já explicados: *Import* lst_ff – *from* glass.pys.oss *import* lst_ff)

Import raster_formats dentro do código get_cellsize

```
102         if type(rst) != list:
103             if os.path.exists(rst) and os.path.isdir(rst):
104                 from glass.pys.oss import lst_ff
105                 from glass.prop  import raster_formats
```

```
13  def raster_formats():
14      return [
15          '.tiff', '.tif', '.img', '.nc', 'ecw', '.jpg',
16          '.png', '.vrt', '.jp2', '.asc'
17      ]
```

Descrição:

Código que dá a conhecer ao *script* principal todos os formatos de *raster* existentes, para que quando seja chamada um *raster*, seja conhecido como este formato independentemente da tua extensão (*.tiff*, *.img*, *.tif*, *.jp2*, etc.).

Import new_pnt dentro do código adjust_ext_to_snap

```
374     from glass.prop     import is_shp, is_rst
375     from glass.prop.rst import rst_ext, get_cellsize
376     from glass.gobj     import new_pnt, create_polygon
```

"Devolve a OGR de um objeto da geometria ponto"

```
7   def new_pnt(x, y):
8       """
9       Return a OGR Point geometry object
10      """
11
12      pnt = ogr.Geometry(ogr.wkbPoint)
13      pnt.AddPoint(float(x), float(y))
14
15      return pnt
```

Métodos e Ferramentas:

- *ogr.Geometry*: construtor de cópia.

- *ogr.wkbPoint*: em conjunto com *ogr.Geometry* são utilizados para construir um ponto.

- *AddPoint*: importante para criação de pontos e posterior criação de polígonos ou linhas.

Descrição:

Pequeno código que tem a principal função de criar um ponto em um polígono e assegurar a devolução da OGR de um objeto a partir desse ponto.

Import create_polygon dentro do código adjust_ext_to_snap

```
374     from glass.prop     import is_shp, is_rst
375     from glass.prop.rst import rst_ext, get_cellsize
376     from glass.gobj     import new_pnt, create_polygon
```

"Devolve a OGR de um objeto de geometria poligonal"

"Opções API: - OGR; - Shapely"

```
def create_polygon(points, api='ogr'):
    """
    Return a OGR Polygon geometry object

    api options:
    * ogr;
    * shapely;
    """

    if api == 'shapely':
        from shapely.geometry import Polygon

        polygon = Polygon(points)

        return polygon

    else:
        ring = ogr.Geometry(ogr.wkbLinearRing)

        for pnt in points:
            if type(pnt) == tuple or type(pnt) == list:
                ring.AddPoint(pnt[0], pnt[1])
            else:
                ring.AddPoint(pnt.GetX(), pnt.GetY())

        polygon = ogr.Geometry(ogr.wkbPolygon)
        polygon.AddGeometry(ring)

    return polygon
```

Métodos e Ferramentas:

```
    if api == 'shapely':
        from shapely.geometry import Polygon

        polygon = Polygon(points)

        return polygon
```

- *shapely.geometry*: classes de geometria *shapely*, como *shapely.Point* ou *shapely.Polygon*, são os tipos de dados centrais de *shapely*. Cada classe da geometria abrange a *shapely.Geometry* classe base, considerado como um container do objeto de geometria GEOS subjacente, para fornecer atributos e comportamentos específicos do tipo de geometria. O *geometry* objeto analisa a geometria GEOS subjacente e permite que o *python* liberte memória quando não for mais utilizado. Objetos de geometria são inalteráveis. Isso significa que, depois de construídas, não podem ser alteradas no local. Cada operação *shapely* resulta na devolução de um objeto novo.

- *import Polygon*: um tipo de geometria que representa uma área delimitada por um anel linear.

```
    else:
        ring = ogr.Geometry(ogr.wkbLinearRing)

        for pnt in points:
            if type(pnt) == tuple or type(pnt) == list:
                ring.AddPoint(pnt[0], pnt[1])
            else:
                ring.AddPoint(pnt.GetX(), pnt.GetY())

        polygon = ogr.Geometry(ogr.wkbPolygon)
        polygon.AddGeometry(ring)

    return polygon
```

- *ogr.Geometry*: construtor de cópia.

- *ogr.wkbLinearRing*: em conjunto com *ogr.Geometry* tem a função de construir uma *linestring* ou um polygon.

- *AddPoint*: importante para criação de pontos e posterior criação de polígonos ou linhas.

- *GetX*: serve para obter o valor da coordenada X.

- *GetY*: serve para obter o valor da coordenada Y.

- *ogr.wkbPolygon*: em conjunto com *ogr.Geometry* têm a função de contruir um polígono.

- *AddGeometry*: adiciona uma geometria ao container. Algumas subclasses de OGR *Geometry Collection* são restritos aos tipos de geometria podendo ser adicionados e devolver um erro. A geometria passada é duplicada para fazer uma cópia interna.

Descrição:

Código importante para a criação de um polígono, pois inclui no seu *script* ferramentas que realizam a criação de pontos para a posterior formação de polígonos. Para além disso devolve a OGR de uma forma geométrica do tipo polígono.

Import **rst_to_array dentro do código shp_to_rst**

```
106         else:
107             from glass.rd.rst import rst_to_array
```

"Converte uma imagem *raster* para *numpy array*"

"Se o achatamento for verdadeiro, o resultado será uma *shape* de (1, 1)"

"Se sem dados for igual a verdadeiro, o resultado terá valores sem dados"

```
6  def rst_to_array(r, flatten=False, with_nodata=True):
7      """
8      Convert Raster image to numpy array
9
10     If flatten equal a True, the output will have a shape of (1, 1).
11
12     If with_nodata equal a True, the output will have the nodata values
13     """
14
15     from osgeo         import gdal
16     from glass.rd.rsrc import imgsrc_to_num
17
18     img = gdal.Open(r)
19
20     return imgsrc_to_num(img, flatten=flatten, with_nodata=with_nodata)
```

Métodos e Ferramentas:

- *gdal.Open*: função utilizada para abrir um *dataset*.

Descrição:

Pequeno *script* que realiza a conversão de um *raster* para um *array*. A conversão é feita independentemente dos resultados que se podem obter, mas o *array* é criado.

Imports dentro do rst_to_array:

```
6  def rst_to_array(r, flatten=False, with_nodata=True):
7      """
8      Convert Raster image to numpy array
9
10     If flatten equal a True, the output will have a shape of (1, 1).
11
12     If with_nodata equal a True, the output will have the nodata values
13     """
14
15     from osgeo         import gdal
16     from glass.rd.rsrc import imgsrc_to_num
17
18     img = gdal.Open(r)
19
20     return imgsrc_to_num(img, flatten=flatten, with_nodata=with_nodata)
```

***Import* imgsrc_to_num dentro do código rst_to_array**

```
15      from osgeo          import gdal
16      from glass.rd.rsrc import imgsrc_to_num
```

"Converte *raster* para *numpy array*"

```
5   import numpy as np
6
7   def imgsrc_to_num(img, flatten=None, with_nodata=True):
8       """
9       Convert Raster Source to Numpy Array
10      """
11
12      if not flatten and with_nodata:
13          return img.ReadAsArray()
14      elif flatten and with_nodata:
15          return img.ReadAsArray().flatten()
16      elif flatten and not with_nodata:
17          bnd = img.GetRasterBand(1)
18          no_val = bnd.GetNoDataValue()
19          values = img.ReadAsArray().flatten()
20
21          return np.delete(values, np.where(values==no_val), None)
22      else:
23          bnd = img.GetRasterBand(1)
24          no_val = bnd.GetNoDataValue()
25          values = img.ReadAsArray()
26
27          return np.delete(values, np.where(values==no_val), None)
```

Métodos e Ferramentas:

- *ReadsAsArray*: utilizado em *python* para ler *arrays*.

- *flatten*: método que devolve uma cópia do *array* recolhida numa dimensão.

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *GetNoDataValue*: vai buscar o valor sem dados para esta banda. Se não houver nenhum valor de dados fora do intervalo, um valor fora do intervalo é devolvido. O valor sem dados para uma banda, geralmente é um valor especial usado para destacar pixéis que não são dados válidos.

- *numpy.Delete*: devolve um novo *array* com *sub-arrays* ao longo de um eixo excluído.

- *numpy.Where*: devolve elementos X ou Y dependendo da condição imposta.

Descrição:

Código que realiza a conversão de um *raster* para um *array*, no entanto este *array* acaba por ter a dimensão referente à banda (*GetRasterBand*), mas sem qualquer tipo de valores associados. Importante para obter *arrays* com determinadas extensões, tem em consideração as bandas dos *rasters* chamadas para o código.

Import fprop dentro do código shp_to_rst

```
        import os
        from glass.pys.oss    import fprop
        from glass.wenv.grs import run_grass
        from glass.prop.prj import get_epsg
```

```
def fprop(__file, prop, forceLower=None, fs_unit=None):
    """
    Return some property of file

    prop options:
    * filename or fn - return filename
    """

    from glass.pys import obj_to_lst

    prop = obj_to_lst(prop)

    result = {}

    if 'filename' in prop or 'fn' in prop:
        fn, ff = os.path.splitext(os.path.basename(__file))

        result['filename'] = fn if not forceLower else fn.lower()

        if 'fileformat' in prop or 'fn' in prop:
            result['fileformat'] = ff

    elif 'fileformat' in prop or 'ff' in prop:
        result['fileformat'] = os.path.splitext(__file)[1]

    if 'filesize' in prop or 'fs' in prop:
        fs_unit = 'MB' if not fs_unit else fs_unit

        fs = os.path.getsize(__file)

        if fs_unit == 'MB':
            fs  = (fs / 1024.0) / 1024

        elif fs_unit == 'KB':
            fs = fs / 1024.0

        result['filesize'] = fs

    if len(prop) == 1:
        if prop[0] == 'fn':
            return result['filename']
        elif prop[0] == 'ff':
            return result['fileformat']
        elif prop[0] == 'fs':
            return result['filesize']
        else:
            return result[prop[0]]
    else:
        return result
```

(Já explicados: *Import* fprop dentro do código is_rst)

Import run_grass dentro do código shp_to_rst

```
144         import os
145         from glass.pys.oss    import fprop
146         from glass.wenv.grs import run_grass
147         from glass.prop.prj import get_epsg
```

```
198 def run_grass(workspace, grassBIN=GRASS_BIN, location=None, srs=None):
199     """
200     Generic method that could be used to put GRASS GIS running in any Os
201 
202     To work on MSWindows, Path to GRASS Must be in the PATH Environment
203     Variables
204     """
205 
206     from glass.pys.oss import os_name
207 
208     __os = os_name()
209 
210     if location:
211         from glass.pys.oss import lst_fld
212 
213         # Check if location exists
214         flds = lst_fld(workspace, name=True)
215 
216         if location in flds:
217             base = start_grass_linux_existLocation(
218                 workspace, grassBin=grassBIN
219             ) if __os == 'Linux' else start_grass_win_exisLocation(
220                 workspace, grassBin=grassBIN)
221 
222         else:
223             base = start_grass_linux_newLocation(
224                 workspace, location, srs=srs, grassBin=grassBIN
225             ) if __os == 'Linux' else start_grass_win_newLocation(
226                 workspace, location, srs=srs, grassBin=grassBIN
227             ) if __os == 'Windows' else None
228 
229     else:
230         base = start_grass_linux_existLocation(
231             workspace, grassBin=grassBIN
232         ) if __os == 'Linux' else start_grass_win_exisLocation(
233             workspace, grassBin=grassBIN
234         )
235 
236     if not base:
237         raise ValueError((
238             'Could not identify operating system'
239         ))
240 
241     return base
242 
```

(Já explicados: *Import* run_grass dentro do código match_cellsize_and_clip)

***Import* get_epsg dentro do código shp_to_rst**

```
144         import os
145         from glass.pys.oss    import fprop
146         from glass.wenv.grs import run_grass
147         from glass.prop.prj import get_epsg
```

"Obtém o código EPSG de qualquer ficheiro SIG"

```
173 def get_epsg(inFile):
174     """
175     Get EPSG of any GIS File
176     """
177 
178     from glass.prop import is_rst, is_shp
179 
180     if is_rst(inFile):
181         return get_rst_epsg(inFile)
182     else:
183         if is_shp(inFile):
184             return get_shp_epsg(inFile)
185         else:
186             return None
```

Descrição:

Pequeno *script* que recolhe o código EPSG de qualquer ficheiro SIG, e é por isso que nele foram implementados os *imports* is_rst e is_shp.

Imports dentro do get_epsg:

```
173 def get_epsg(inFile):
174     """
175     Get EPSG of any GIS File
176     """
177 
178     from glass.prop import is_rst, is_shp
179 
180     if is_rst(inFile):
181         return get_rst_epsg(inFile)
182     else:
183         if is_shp(inFile):
184             return get_shp_epsg(inFile)
185         else:
186             return None
```

Import is_rst e is_shp dentro do código get_epsg

```
def is_rst(_file):
    from glass.pys.oss import fprop

    rst_lst = raster_formats()

    file_ext = fprop(_file, 'ff')

    if file_ext not in rst_lst:
        return None
    else:
        return True


def is_shp(_file):
    from glass.pys.oss import fprop

    lst = vector_formats()

    file_ext = fprop(_file, 'ff')

    if file_ext not in lst:
        return None
    else:
        return True
```

(Já explicados: *Import* is_rst dentro do código match_cellsize_and_clip)

(Já explicados: *Import* is_shp dentro do código adjust_ext_to_snap)

Import packages shp_to_grs dentro do código shp_to_rst

```
        # Import Packages
        from glass.it.shp   import shp_to_grs
        from glass.it.rst   import grs_to_rst
        from glass.wenv.grs import shp_to_region
```

```
def shp_to_grs(ilyr, olyr, filterByReg=None, asCMD=None):
    """
    Add Shape to GRASS GIS
    """

    if not asCMD:
        from grass.pygrass.modules import Module

        f = 'o' if not filterByReg else 'ro'

        m = Module(
            "v.in.ogr", input=ilyr, output=olyr, flags=f,
            overwrite=True, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        f = " -r" if filterByReg else ""

        rcmd = execmd((
            f"v.in.ogr input={ilyr} output={olyr} -o{f} "
            "--overwrite --quiet"
        ))

    return olyr
```

(Já explicados: *Import packages* (relacionados com GRASS GIS) shp_to_grs dentro do código match_cellsize_and_clip)

Import packages grs_to_rst dentro do código shp_to_rst

```
        # Import Packages
        from glass.it.shp   import shp_to_grs
        from glass.it.rst   import grs_to_rst
        from glass.wenv.grs import shp_to_region
```

```
def grs_to_rst(grsRst, rst, as_cmd=None, allBands=None, is_int=None):
    """
    GRASS Raster to Raster
    """

    from glass.prop  import grs_rst_drv
    from glass.pys.oss import fprop

    rstDrv = grs_rst_drv()
    rstExt = fprop(rst, 'ff')

    predictor = "2" if is_int else "3"

    if not as_cmd:
        from grass.pygrass.modules import Module

        m = Module(
            "r.out.gdal", input=grsRst, output=rst,
            format=rstDrv[rstExt], flags='c' if not allBands else '',
            createopt="INTERLEAVE=PIXEL,TFW=YES" if allBands else \
                f'COMPRESS=LZW,PREDICTOR={predictor},BIGTIFF=YES',
            overwrite=True, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        if not allBands:
            opt = f"createopt=\"COMPRESS=LZW,PREDICTOR={predictor},BIGTIFF=YES\""
        else:
            opt = "createopt=\"INTERLEAVE=PIXEL,TFW=YES\""

        rcmd = execmd((
            f"r.out.gdal input={grsRst} output={rst} "
            f"format={rstDrv[rstExt]} {opt} -c --overwrite --quiet"
        ))

    return rst
```

(Já explicados: *Import packages* (relacionados com GRASS GIS) grst_to_rst dentro do código match_cellsize_and_clip)

***Import packages* shp_to_region dentro do código shp_to_rst**

```
160         # Import Packages
161         from glass.it.shp   import shp_to_grs
162         from glass.it.rst   import grs_to_rst
163         from glass.wenv.grs import shp_to_region
```

"*Feature Class* para a região"

```
258 def shp_to_region(shp, cellsize=None):
259     """
260     Feature Class to region
261     """
262 
263     from grass.pygrass.modules import Module
264 
265     r = Module("g.region", vector=shp, res=cellsize, run_=False, quiet=True)
266 
267     r()
```

Métodos e Ferramentas:

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*g.region*": define o limite para a região geográfica.

Descrição:

Pequeno código que cria as definições de limite da região geográfica que vai ser trabalhada e para além disso georreferencia *features classes* para essa área limite definida.

Anexo D2 – Programa 2: *Remove clouds from Sentinel-2 bands* (código existente).

Bloco de código, em *pyhton*, preparado para realizar a eliminação das células com valores de nuvens, sombras e outras intervenções que afetassem as informações espetrais das várias bandas, seguido do seu estudo de bibliotecas e funções implementadas, uma vez que o presente código foi disponibilizado pela orientação.

Remove clouds from sentinel-2 bands

In [1]:
```
folder = '/home/francisco/create_grid/stl_2/step2_2020'

bands = [
    'b02', 'b03', 'b04', 'b05', 'b06', 'b07',
    'b08', 'b8a', 'b09', 'b11', 'b12'
]

scl = 'scl'

ff = '.tif'

ofolder = '/home/francisco/create_grid/stl_2/step3_2020'

clouds_file = '/home/francisco/create_grid/stl_2/clound_file/clouds_2020.tif'
```

In [2]:
```
import os
from glass.pys.oss  import lst_ff, fprop
from glass.pys.tm   import now_as_str
from glass.wenv.grs import run_grass
from glass.rst.rcls import rcls_rules
```

In [3]:
```
# List bands
imgs = lst_ff(folder, file_format=ff, fnpart=bands)

# List scl
scls = lst_ff(folder, file_format=ff, fnpart=[scl])
```

In [4]:
```
# Create GRASS GIS Session

ws, loc = ofolder, f"loc_{now_as_str()}"

grsb = run_grass(ws, location=loc, srs=imgs[0])

import grass.script.setup as gsetup

gsetup.init(grsb, ws, loc, 'PERMANENT')
```

Out[4]:
```
'/tmp/tmp3xmlueq9'
```

In [5]:
```
from glass.it.rst   import rst_to_grs, grs_to_rst
from glass.rst.rcls import rcls_rst, set_null, null_to_value
from glass.rst.alg  import rstcalc
```

In [6]:
```
cldrules = rcls_rules({
    0 : 'NULL',
    1 : 'NULL',
    2 : 1,
    3 : 1,
    4 : 'NULL',
    5 : 'NULL',
    6 : 'NULL',
    7 : 'NULL',
    8 : 1,
    9 : 1,
    10 : 1,
    11 : 'NULL'
}, os.path.join(ws, loc, 'clouds_rules.txt'))

rscl = []
for s in scls:
    # Add SCL to GRASS GIS
    _s = rst_to_grs(s, fprop(s, 'fn'))

    # Reclassify all files to get only things related to clouds
    rs = rcls_rst(_s, cldrules, f'rcls_{_s}', api='grass')
    _rs = rstcalc(rs, f'cp_{_s}', api='grass')

    # Clouds to NoData
    null_to_value(_rs, 0, as_cmd=True)
    set_null(_rs, 1, ascmd=True)

    rscl.append(_rs)

# One file with all clouds as nodata
cloud_rst = rstcalc(" + ".join(rscl), 'clouds', api='grass')

# Export clouds
rcld = grs_to_rst(cloud_rst, clouds_file, is_int=True)
```

In [7]:
```
for i in imgs:
    # Import all bands to GRASS GIS
    _i = rst_to_grs(i, fprop(i, 'fn'))

    # Sum to clouds file
    i_noclouds = rstcalc(f"{_i} + {cloud_rst}", f'nocld_{_i}', api='grass')

    # Export file
    grs_to_rst(i_noclouds, os.path.join(
        ofolder, os.path.basename(i)
    ), is_int=True)
```

Análise e estudo do Programa 2: *Remove clouds from Sentinel-2 bands*

Remove clouds from sentinel-2 bands

[1]:
```
folder = '/home/francisco/lisboa/paco2'

bands = [
    'b02', 'b03', 'b04', 'b05', 'b06', 'b07',
    'b08', 'b8a', 'b09', 'b11', 'b12'
]

scl = 'scl'

ff = '.tif'

ofolder = '/home/francisco/lisboa/paco3'

clouds_file = '/home/francisco/lisboa/clouds_2022.tif'
```

Caminho utilizado para chegar aos resultados do Programa 1

O número das bandas onde serão eliminadas as células de nuvens

Caminho para os *outputs* das bandas após aplicação das máscaras

Criação do ficheiro referente às nuvens e aos pixéis com informações indesejadas

[2]:
```
import os
from glass.pys.oss  import lst_ff, fprop
from glass.pys.tm   import now_as_str
from glass.wenv.grs import run_grass
from glass.rst.rcls import rcls_rules
```

[3]:
```
# List bands
imgs = lst_ff(folder, file_format=ff, fnpart=bands)

# List scl
scls = lst_ff(folder, file_format=ff, fnpart=[scl])
```

[4]:
```
# Create GRASS GIS Session

ws, loc = ofolder, f"loc_{now_as_str()}"

grsb = run_grass(ws, location=loc, srs=imgs[0])

import grass.script.setup as gsetup

gsetup.init(grsb, ws, loc, 'PERMANENT')
```

[4]:
```
'/tmp/tmphx7ezfsz'
```

[5]:
```
from glass.it.rst   import rst_to_grs, grs_to_rst
from glass.rst.rcls import rcls_rst, set_null, null_to_value
from glass.rst.alg  import rstcalc
```

[6]:
```
cldrules = rcls_rules({
    0 : 'NULL',
    1 : 'NULL',
    2 : 1,
    3 : 1,
    4 : 'NULL',
    5 : 'NULL',
    6 : 'NULL',
    7 : 'NULL',
    8 : 1,
    9 : 1,
    10 : 1,
    11 : 'NULL'
}, os.path.join(ws, loc, 'clouds_rules.txt'))

rscl = []
for s in scls:
    # Add SCL to GRASS GIS
    _s = rst_to_grs(s, fprop(s, 'fn'))

    # Reclassify all files to get only things related to clouds
    rs = rcls_rst(_s, cldrules, f'rcls_{_s}', api='grass')
    _rs = rstcalc(rs, f'cp_{_s}', api='grass')

    # Clouds to NoData
    null_to_value(_rs, 0, as_cmd=True)
    set_null(_rs, 1, ascmd=True)

    rscl.append(_rs)

# One file with all clouds as nodata
cloud_rst = rstcalc(" + ".join(rscl), 'clouds', api='grass')

# Export clouds
rcld = grs_to_rst(cloud_rst, clouds_file, is_int=True)
```

(Parte final do Programa 2)

```
[7]: for i in imgs:
         # Import all bands to GRASS GIS
         _i = rst_to_grs(i, fprop(i, 'fn'))

         # Sum to clouds file
         i_noclouds = rstcalc(f"{_i} + {cloud_rst}", f'nocld_{_i}', api='grass')

         # Export file
         grs_to_rst(i_noclouds, os.path.join(
             ofolder, os.path.basename(i)
         ), is_int=True)
```

Programa 2: Extremamente importante para a eliminar células de nuvens presentes nas diferentes bandas da série temporal das imagens Sentinel-2. Para além de eliminar as nuvens das imagens, também exclui todos os pixéis que se consideram como sombras, células incompletas e sem informação. Passo pertinente para a preparação das imagens para que possam ser posteriormente classificadas através dos dados de treino.

Imports:

```
[2]: import os
     from glass.pys.oss  import lst_ff, fprop
     from glass.pys.tm   import now_as_str
     from glass.wenv.grs import run_grass
     from glass.rst.rcls import rcls_rules
```

```
[5]: from glass.it.rst   import rst_to_grs, grs_to_rst
     from glass.rst.rcls import rcls_rst, set_null, null_to_value
     from glass.rst.alg  import rstcalc
```

- *import* OS: o módulo OS em *Python* fornece funções para interagir com o sistema operacional. O sistema operacional vem sob os módulos padrão do *Python*. Este módulo fornece uma maneira flexível de usar a funcionalidade dependendo do sistema operacional. Os módulos "OS" e "os.path" incluem muitas funções para interagir com o sistema de pastas.

- *from glass.pys.oss import lst_ff, fprop* (…): são diferentes *scripts* pré-criados e importados de outras pastas, introduzidos no Programa 2 através do caminho de pastas. Todos os *imports* seguintes serão explicados nas secções seguintes.

Módulos e ferramentas:

#Criação de uma sessão GRASS GIS

```
[4]: # Create GRASS GIS Session

     ws, loc = ofolder, f"loc_{now_as_str()}"

     grsb = run_grass(ws, location=loc, srs=imgs[0])

     import grass.script.setup as gsetup

     gsetup.init(grsb, ws, loc, 'PERMANENT')
```

- *import.grass.script.setup as gsetup*: importa algumas partes convenientes do GRASS GIS a partir do Pyhton API.

- *gsetup.init*: importante para iniciar a sessão do GRASS GIS, é esta linha que faz com que a sessão se inicie.

```
[6]: cldrules = rcls_rules({
         0 : 'NULL',
         1 : 'NULL',
         2 : 1,
         3 : 1,
         4 : 'NULL',
         5 : 'NULL',
         6 : 'NULL',
         7 : 'NULL',
         8 : 1,
         9 : 1,
         10 : 1,
         11 : 'NULL'
     }, os.path.join(ws, loc, 'clouds_rules.txt'))
```

- *os.path.join*: O módulo *os.path* é um submódulo do módulo OS em *python* usado para manipular nomes de caminhos. O método *os.path.join()* em *python* une um ou mais componentes dos caminhos de forma inteligente. Este método concatena vários componentes dos caminhos com exatamente um separador de pasta (‘/’) após cada parte não vazia, exceto o último elemento do caminho. Exemplo: ‘/home/glass/francisco/’. Utilizado para encaminhar o ficheiro SCL que contém a informação relativa aos pixéis com valores de nuvens e outras interferências atmosféricas.

#Nuvens para valores *noData*

#Um ficheiro em que todas os pixéis de nuvens são considerados como *noData*

#*Export* do ficheiro de nuvens

```
    # Clouds to NoData
    null_to_value(_rs, 0, as_cmd=True)
    set_null(_rs, 1, ascmd=True)

    rscl.append(_rs)

# One file with all clouds as nodata
cloud_rst = rstcalc(" + ".join(rscl), 'clouds', api='grass')

# Export clouds
rcld = grs_to_rst(cloud_rst, clouds_file, is_int=True)
```

- *append*: método que acrescenta um elemento a uma lista final.

- *join*: função utilizado para unir todos os elementos num iterável novo.

#Importar as bandas das imagens para a sessão GRASS GIS

#Soma do ficheiro de nuvens

#*Export* do ficheiro de nuvens

```
[7]: for i in imgs:
        # Import all bands to GRASS GIS
        _i = rst_to_grs(i, fprop(i, 'fn'))

        # Sum to clouds file
        i_noclouds = rstcalc(f"{_i} + {cloud_rst}", f'nocld_{_i}', api='grass')

        # Export file
        grs_to_rst(i_noclouds, os.path.join(
            ofolder, os.path.basename(i)
        ), is_int=True)
```

- *os.path.join*: o módulo os.path é um submódulo do módulo OS em python usado para manipulação comum de nomes de caminho. O método os.path.join() em python une um ou mais componentes de caminho de forma inteligente. Este método concatena vários componentes de caminho com exatamente um separador de diretório (‘/’) após cada parte não vazia, exceto o último componente do caminho. Exemplo: ‘/home/glass/francisco/’.

- *os.path.basename*: é utilizado em *python* para obter o nome base do caminho especifico. Este método utiliza internamente o método *os.path.split()* para dividir o caminho especifico em duas partes, da frente e de trás. O método *os.path.basename()* devolve a parte final depois de dividir o caminho específico nas duas partes.

Principais *Imports* implementados no Programa 2 e descrição dos mesmos

Nas figuras seguintes são apresentados os diferentes *Imports* materializados no Programa 2 com funções e objetivos extremamente importantes para o desempenho do código. Como já foi mencionado anteriormente, a maioria dos *imports* são *scripst*/códigos, desenvolvidos pela orientação, encontrados no repositório GLASS. É preciso notar que a maioria das funções implementadas no Programa 2, foram explicadas e definidas anteriormente no Programa 1 e, portanto, nas secções que se seguem serão descritas apenas funções que ainda não detiveram de um esclarecimento aprofundado e oportuno.

```
[2]: import os
     from glass.pys.oss  import lst_ff, fprop
     from glass.pys.tm   import now_as_str
     from glass.wenv.grs import run_grass
     from glass.rst.rcls import rcls_rules
```

```
[5]: from glass.it.rst   import rst_to_grs, grs_to_rst
     from glass.rst.rcls import rcls_rst, set_null, null_to_value
     from glass.rst.alg  import rstcalc
```

(Já explicados: *Import OS*, lst_ff, fprop, now_as_str, run_grass, rst_to_grs e grs_to_rst incluídos no Programa 1)

***Import* rcls_rules – *from* glass.rst.rcls *import* rcls_rules**

```
[2]: import os
     from glass.pys.oss  import lst_ff, fprop
     from glass.pys.tm   import now_as_str
     from glass.wenv.grs import run_grass
     from glass.rst.rcls import rcls_rules
```

"*Reclassify* em ambiente GRASS GIS"

"Descrição das regras de reclassificação num ficheiro"

```
"""
Reclassify in GRASS GIS
"""

def rcls_rules(dic, out_rules):
    """
    Write reclassify rules file
    """

    if os.path.splitext(out_rules)[1] != '.txt':
        out_rules = os.path.splitext(out_rules)[0] + '.txt'

    with open(out_rules, 'w') as txt:
        for k in dic:
            if type(k) == tuple:
                thru = [
                    f'{str(k[i-1])} thru {str(k[i])}'
                    for i in range(1, len(k), 2)
                ]

                txt.write(f'{"  ".join(thru)}  = {str(dic[k])}\n')

            else:
                txt.write(f'{str(k)}  = {str(dic[k])}\n')

    return out_rules
```

Métodos e Ferramentas:

- *os.path.splitext*: o método *OS.path* é um submódulo do módulo *OS* em *python* utilizado para manipular nomes dos caminhos. *Os.path.splitext()* é o método usado em *python* para dividir o nome do caminho em um par *root* e *ext*. Aqui, *ext* significa extensão e possui a parte da extensão do caminho especificado, enquanto *root* é tudo, exceto a parte *ext*.

- *write*: escreve e substitui qualquer conteúdo existente.

- *join*: função utilizado para unir todos os elementos num iterável novo.

Descrição:

Código relevante que discrimina as regras para reclassificação para determinar os valores de pixéis que devem ser mantidos nas bandas das imagens Sentinel-2 e os valores dos pixéis que devem ser eliminados das imagens. Estas regras atribuem um valor único para os pixéis com lacunas e um outro valor único para todos os pixéis importantes para a posterior classificação.

***Import* rcls_rst – *from* glass.rst.rcls *import* rcls_rst**

```
[5]: from glass.it.rst   import rst_to_grs, grs_to_rst
     from glass.rst.rcls import rcls_rst, set_null, null_to_value
     from glass.rst.alg  import rstcalc
```

"Reclassificação do *raster* (pontos categóricos e *floating*)"

"*if api* == GDAL

rclsRules = {

1 : 90,

2 : 100,

…

}

Or

rclsRules = {

(0, 8) : 1

(8, 16) : 2

'*' : '*NoData*'

}

Elif api == GRASS:

rclsRules deve ser um caminho para um ficheiro do tipo texto"

```
Reclassify Raster files
"""

import os

def rcls_rst(inrst, rclsRules, outrst, api='gdal', maintain_ext=True):
    """
    Reclassify a raster (categorical and floating points)

    if api == 'gdal
    rclsRules = {
        1 : 99,
        2 : 100
        ...
    }

    or

    rclsRules = {
        (0, 8) : 1
        (8, 16) : 2
        '*'     : 'NoData'
    }

    elif api == grass:
    rclsRules should be a path to a text file
    """

    if api == 'gdal':
        import numpy        as np
        from osgeo          import gdal
        from glass.wt.rst   import obj_to_rst
        from glass.rd.rsrc  import imgsrc_to_num
        from glass.prop.img import get_nd
        if not os.path.exists(inrst):
            raise ValueError(f'File {inrst} does not exist!')

        # Open Raster
        img = gdal.Open(inrst)

        # Raster to Array
        rst_num = imgsrc_to_num(img)

        nodataVal = get_nd(img)

        rcls_num = np.full(rst_num.shape, 255, dtype=np.uint8)

        # Change values
        for k in rclsRules:
            if rclsRules[k] == 'NoData':
                continue

            if type(k) == str:
                continue

            elif type(k) == tuple:
                q = (rst_num > k[0]) & (rst_num <= k[1])

            else:
                q = rst_num == k

            np.place(rcls_num, q, rclsRules[k])

        if '*' in rclsRules and rclsRules['*'] != 'NoData':
            np.place(rcls_num, rcls_num == 255, rclsRules['*'])

        if 'NoData' in rclsRules and rclsRules['NoData'] != 'NoData':
            np.place(rcls_num, rst_num == nodataVal, rclsRules['NoData'])

        if not maintain_ext:
            from glass.rst.rshp import rshp_to_data
```

(Parte final do *script* rcls_rst)

```
            left, cellx, z, top, c, celly = img.GetGeoTransform()

            clip_rcls, n_left, n_top = rshp_to_data(rcls_num, 255, left, cellx, top, celly)

            return obj_to_rst(clip_rcls, outrst, img, noData=255, geotrans=(
                n_left, cellx, z, n_top, c, celly
            ))
        else:
            return obj_to_rst(rcls_num, outrst, img, noData=255)

    elif api == "pygrass":
        from grass.pygrass.modules import Module

        r = Module(
            'r.reclass', input=inrst, output=outrst, rules=rclsRules,
            overwrite=True, run_=False, quiet=True
        )

        r()

    elif api == "grass":
        from glass.pys import execmd

        rcmd = execmd((
            f"r.reclass input={inrst} output={outrst} "
            f"rules={rclsRules} --overwrite --quiet"
        ))

    else:
        raise ValueError(f"API {api} is not available")

    return outrst
```

Métodos e Ferramentas:

```
    if api == 'gdal':
        import numpy        as np
        from osgeo          import gdal
        from glass.wt.rst   import obj_to_rst
        from glass.rd.rsrc  import imgsrc_to_num
        from glass.prop.img import get_nd

        if not os.path.exists(inrst):
            raise ValueError(f'File {inrst} does not exist!')
```

- *import Numpy*: *NumPy*, significa *Numerical Python*, e é conhecida como uma biblioteca de computação científica construída na linguagem de programação *python*. *Array* é o tipo de dados mais comum de se trabalhar a partir desta biblioteca.

- *from osgeo import gdal:* GDAL é uma biblioteca C++ para mais de 200 formatos de dados geospaciais *raster* e vetoriais. É lançado sob uma licença *open source pela Open Source Geospacial Foudation.*

- *os.path.exists*: este método em *python* é usado para verificar se o caminho especificado existe. Este método também pode ser utilizado para verificar se o caminho indicado se relaciona com uma pasta aberto ou não.

#Abrir Raster

#Raster para Array

```
        # Open Raster
        img = gdal.Open(inrst)

        # Raster to Array
        rst_num = imgsrc_to_num(img)

        nodataVal = get_nd(img)

        rcls_num = np.full(rst_num.shape, 255, dtype=np.uint8)
```

- *gdal.Open*: função utilizada para abrir um *dataset*.

- *np.full*: devolve um *array* com o formato e o tipo definidos pelo utilizador, preenchido com *fill_value*.

- *shape*: os *arrays numpy* têm um atributo chamado *shape* que geralmente é utilizado para obter a área de um *array*, mas também devolve uma tupla que pode ser usada para alterar as dimensões do *array*.

- *uint8*: tipo de dado do *array* considerado como inteiro de 8 *bits* sem sinal.

#Alteração dos valores

```
        # Change values
        for k in rclsRules:
            if rclsRules[k] == 'NoData':
                continue

            if type(k) == str:
                continue

            elif type(k) == tuple:
                q = (rst_num > k[0]) & (rst_num <= k[1])

            else:
                q = rst_num == k

            np.place(rcls_num, q, rclsRules[k])
```

- *np.place*: altera os elementos de um *array* com base nos valores definidos e atribuídos como entrada.

```
        if not maintain_ext:
            from glass.rst.rshp import rshp_to_data

            left, cellx, z, top, c, celly = img.GetGeoTransform()

            clip_rcls, n_left, n_top = rshp_to_data(rcls_num, 255, left, cellx, top, celly)
```

- *GetgeoTransform*: utilizado para obter os coeficientes de transformação. Obtém os coeficientes para transformar a área *raster* pixel/linha (P, L) e a área de coordenadas de projeção (Xp, Yp).

```
85      elif api == "pygrass":
86          from grass.pygrass.modules import Module
87
88          r = Module(
89              'r.reclass', input=inrst, output=outrst, rules=rclsRules,
90              overwrite=True, run_=False, quiet=True
91          )
92
93          r()
94
95      elif api == "grass":
96          from glass.pys import execmd
97
98          rcmd = execmd((
99              f"r.reclass input={inrst} output={outrst} "
100             f"rules={rclsRules} --overwrite --quiet"
```

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*r.reclass*": reclassifica um *raster* com base nos valores das classes. Cria um *raster* com valores baseados na reclassificação das calasses, a partir de um *raster* já existente.

Descrição:

Script importante que realiza a reclassificação de *rasters*, classificando os pixéis com anomalias com um determinado valor e os restantes pixéis com um valor distinto, de maneira a realizar a posterior eliminação de todas as células perturbadas pelos efeitos atmosféricos.

Imports dentro do rcls_rst:

```
30      if api == 'gdal':
31          import numpy        as np
32          from osgeo          import gdal
33          from glass.wt.rst   import obj_to_rst
34          from glass.rd.rsrc  import imgsrc_to_num
35          from glass.prop.img import get_nd

72          if not maintain_ext:
73              from glass.rst.rshp import rshp_to_data

95      elif api == "grass":
96          from glass.pys import execmd
```

Import obj_to_rst dentro do código rcls_rst

```
30      if api == 'gdal':
31          import numpy         as np
32          from osgeo           import gdal
33          from glass.wt.rst    import obj_to_rst
34          from glass.rd.rsrc   import imgsrc_to_num
35          from glass.prop.img import get_nd
```

"*Array* para *Raster*"

```
9   def obj_to_rst(inArray, outRst, template, noData=None, geotrans=None):
10      """
11      Send Array to Raster
12      """
13
14      from osgeo            import gdal, osr, gdal_array
15      from glass.prop     import drv_name
16      from glass.prop.rst import compress_option
17
18      if type(template).__name__ == 'Dataset':
19          img_template = template
20      else:
21          img_template  = gdal.Open(template)
22
23      geo_transform = img_template.GetGeoTransform() if not geotrans else \
24          geotrans
25      rows, cols    = inArray.shape
26      drv_n         = drv_name(outRst)
27      driver        = gdal.GetDriverByName(drv_n)
28
29      c_opt = compress_option(drv_n)
30      if c_opt:
31          out       = driver.Create(
32              outRst, cols, rows, 1,
33              gdal_array.NumericTypeCodeToGDALTypeCode(inArray.dtype),
34              options=[c_opt]
35          )
36      else:
37          out       = driver.Create(
38              outRst, cols, rows, 1,
39              gdal_array.NumericTypeCodeToGDALTypeCode(inArray.dtype)
40          )
41      out.SetGeoTransform(geo_transform)
42      outBand       = out.GetRasterBand(1)
43
44      if noData or noData==0:
45          outBand.SetNoDataValue(noData)
46
47      outBand.WriteArray(inArray)
48
49      proj = osr.SpatialReference(wkt=img_template.GetProjection())
50
51      if proj:
52          out.SetProjection(img_template.GetProjection())
53
54      outBand.FlushCache()
55
56      return outRst
```

Métodos e Ferramentas:

```
18      if type(template).__name__ == 'Dataset':
19          img_template = template
20      else:
21          img_template  = gdal.Open(template)
```

- *gdal.Open*: função utilizada para abrir um *dataset*.

```
    geo_transform = img_template.GetGeoTransform() if not geotrans else \
        geotrans
    rows, cols    = inArray.shape
    drv_n         = drv_name(outRst)
    driver        = gdal.GetDriverByName(drv_n)
```

- *GetgeoTransform*: utilizado para obter os coeficientes de transformação. Obtém os coeficientes para transformar a área *raster* pixel/linha (P, L) e a área de coordenadas de projeção (Xp, Yp).

- *shape*: os *arrays numpy* têm um atributo chamado *shape* que geralmente é utilizado para obter a área de um *array*, mas também devolve uma tupla que pode ser usada para alterar as dimensões do *array*.

- *GetDriverbyName*: utilizado para procurar o driver com base no nome curto. Parâmetros: *pszName* – o nome curto com *GTiff*, que está a ser procurado. Devoluções: identificação do nome ou *Null* se não for encontrada nenhuma correspondência.

```
    c_opt = compress_option(drv_n)
    if c_opt:
        out       = driver.Create(
            outRst, cols, rows, 1,
            gdal_array.NumericTypeCodeToGDALTypeCode(inArray.dtype),
            options=[c_opt]
```

- *Create*: o método *create* permite gravar explicitamente todos os metadados e dados *raster*. É importante e oferece suporte à criação de novos ficheiros.

- *NumericTypeCodeToGDALTypeCode*: traduz um valor do tipo "numérico" num tipo de código GDAL.

- *dtype*: descreve o tipo de dados de um objeto. Define como devem ser interpretados os *bytes* do tamanho de um *array*. Esta função define as características dos dados como: 1.Tipo de dado (*integer*, *float*, *python object*, etc.); 2. Tamanho do dado (quantos *bytes* são agrupados); 3. Ordem dos *bytes* dos dados (*little-endian or big-endian*); 4. Se o tipo de dados for estruturado, os dados, podem subdividir-se nos diferentes "campos", os quais podem ser acedidos, quanto aos dados presentes em cada "campo" e o seu peso de memória; 5. Se o tipo de dados for um *sub-array*, tem-se em consideração a sua forma e tipo de dado.

```
    out.SetGeoTransform(geo_transform)
    outBand       = out.GetRasterBand(1)

    if noData or noData==0:
        outBand.SetNoDataValue(noData)

    outBand.WriteArray(inArray)

    proj = osr.SpatialReference(wkt=img_template.GetProjection())
```

- *SetGeoTransform*: define os parâmetros de transformação para a fonte de dados GDAL.

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *SetNoDataValue*: define o valor *nodata* como duplo, *uint16* e *int16*. Se o tipo de dados natural do atributo/matriz não for um dos três referidos, a conversão de tipo será em função do tipo definido por *GetDataType ()*.

- *WriteArray*: função utilizada para escrever e colocar informações no *array* determinadas pelo utilizados.

- *osr.SpatialReference*: serve para indicar que se vai chamar o sistema de referência.

- *GetProjection*: serve para procurar a projeção do ficheiro definido.

```
51      if proj:
52          out.SetProjection(img_template.GetProjection())
53
54      outBand.FlushCache()
```

- *FlushCache*: Descarrega todos os dados armazenados no *disk cache*. Qualquer dado *raster* ou outro guardado através da biblioteca GDAL será gravado no disco.

Descrição:

Código que faz a conversão de um objeto do tipo *array* para formato *raster*. Este processo realiza apenas a conversão destes dois formatos, no entanto tem em atenção ao sistema de referência que o mantém do *array* para o *raster*. A conversão implica a transferência dos dados e dos campos dos *arrays* para *raster*, onde essas informações serão mantidas.

Imports dentro do obj_to_rst:

```
14      from osgeo           import gdal, osr, gdal_array
15      from glass.prop     import drv_name
16      from glass.prop.rst import compress_option
```

Import drv_name dentro do código obj_to_rst

"Devolve o formato de um ficheiro"

```
    # Get snapRst Extent
    snapRstExt = rst_ext(snapRst)

    # Get cellsize
    csize = get_cellsize(snapRst)

    # Find extent point of outExt inside the two extents
    # This will be used as pseudo origin

    snapRstPnt = [
        new_pnt(snapRstExt[0], snapRstExt[3]),
        new_pnt(snapRstExt[1], snapRstExt[3]),
        new_pnt(snapRstExt[1], snapRstExt[2]),
        new_pnt(snapRstExt[0], snapRstExt[2]),
        new_pnt(snapRstExt[0], snapRstExt[3]),
    ]

    poly_snap_rst = create_polygon(snapRstPnt)

    outExtPnt = {
        'top_left'     : new_pnt(shpAExt[0], shpAExt[3]),
        'top_right'    : new_pnt(shpAExt[1], shpAExt[3]),
        'bottom_right' : new_pnt(shpAExt[1], shpAExt[2]),
        'bottom_left'  : new_pnt(shpAExt[0], shpAExt[2])
    }

    out_rst_pseudo = {}
    for pnt in outExtPnt:
        out_rst_pseudo[pnt] = outExtPnt[pnt].Intersects(poly_snap_rst)

    pseudoOrigin = outExtPnt['top_left'] if out_rst_pseudo['top_left'] else \
        outExtPnt['bottom_left'] if out_rst_pseudo['bottom_left'] else \
        outExtPnt['top_right'] if out_rst_pseudo['top_right'] else \
        outExtPnt['bottom_right'] if out_rst_pseudo['bottom_right'] else None
```

(Já explicados: *Import* drv_name incluído no Programa 1)

Import compress_option dentro do código obj_to_rst

```
14    from osgeo           import gdal, osr, gdal_array
15    from glass.prop     import drv_name
16    from glass.prop.rst import compress_option
```

"Devolve uma compactação opcional para qualquer driver GDAL"

```
5  def compress_option(drv):
6      """
7      Return compress option for some gdal driver
8      """
9
10     if drv == 'GTiff':
11         return 'COMPRESS=LZW'
12     else:
13         return None
```

Descrição:

Pequeno código que realiza a simples compactação de um ficheiro *.tif* para reduzir o seu tamanho e peso de memória.

Import imgsrc_to_num dentro do código rcls_rst

```
30     if api == 'gdal':
31         import numpy        as np
32         from osgeo          import gdal
33         from glass.wt.rst   import obj_to_rst
34         from glass.rd.rsrc  import imgsrc_to_num
35         from glass.prop.img import get_nd
```

"Converte uma fonte dados *raster* para *array numpy*"

```
5  import numpy as np
6
7  def imgsrc_to_num(img, flatten=None, with_nodata=True):
8      """
9      Convert Raster Source to Numpy Array
10     """
11
12     if not flatten and with_nodata:
13         return img.ReadAsArray()
14     elif flatten and with_nodata:
15         return img.ReadAsArray().flatten()
16     elif flatten and not with_nodata:
17         bnd = img.GetRasterBand(1)
18         no_val = bnd.GetNoDataValue()
19         values = img.ReadAsArray().flatten()
20
21         return np.delete(values, np.where(values==no_val), None)
22     else:
23         bnd = img.GetRasterBand(1)
24         no_val = bnd.GetNoDataValue()
25         values = img.ReadAsArray()
26
27         return np.delete(values, np.where(values==no_val), None)
```

(Já explicados: *Import* imgsrc_to_num incluído no Programa 1)

***Import* get_nd dentro do código rcls_rst**

```
30      if api == 'gdal':
31          import numpy        as np
32          from osgeo          import gdal
33          from glass.wt.rst   import obj_to_rst
34          from glass.rd.rsrc  import imgsrc_to_num
35          from glass.prop.img import get_nd
```

"Devolve valor *NoData*"

```
5   def get_nd(img):
6       """
7       Return NoData Value
8       """
9
10      band = img.GetRasterBand(1)
11
12      return band.GetNoDataValue()
```

Métodos e Ferramentas:

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *GetNoDataValue*: vai buscar o valor sem dados para esta banda. Se não houver nenhum valor de dados fora do intervalo, um valor fora do intervalo é devolvido. O valor sem dados para uma banda, geralmente é um valor especial usado para destacar pixéis que não são dados válidos.

Descrição:

Script muito curto que vai buscar o valor *NoData* para um objeto (banda), importante para destacar pixéis que não contém informação válida para a posterior análise de imagens de satélite (Sentinel-2, Landsat-8, etc.).

***Import* rshp_to_data dentro do código rcls_rst**

```
72          if not maintain_ext:
73              from glass.rst.rshp import rshp_to_data
```

"Escolhe um *array raster* e elimina todas as linhas e as colunas atribuindo valor de *NoData*"

"Devolve um novo *array* e um novo topo e ladoesquerdo"

```
6   def rshp_to_data(in_arr, ndval, left, cellx, top, celly):
7       """
8       Pick Raster array and delete all rows and cols with only
9       NoData Values
10
11      Returns new array and new top and left
12      """
13
14      import numpy as np
15
16      X = in_arr.reshape((in_arr.shape[0] * in_arr.shape[1]))
17      I = np.arange(X.shape[0])
18
19      np.place(I, X == ndval, -1)
20      I = I[I != -1]
21
22      # Get Minimum and Maximum Row Indexes
23      # considering the rows with cells with data
24      i_row = I / in_arr.shape[0]
25      i_row = i_row.astype(np.uint32)
26      min_row, max_row = i_row.min(), i_row.max()
27
28      # Get Minimum and Maximum Column Indexes
29      i_col = I - (i_row * in_arr.shape[1])
30      min_col, max_col = i_col.min(), i_col.max()
31
32      # Get Clipped array
33      clipnum = in_arr[min_row:max_row+1, min_col:max_col+1]
34
35      # Get new left and new top
36      new_left = left + (cellx * min_col)
37      new_top  = top  + (celly * min_row)
38
39      return clipnum, new_left, new_top
```

Métodos e Ferramentas:

```
14      import numpy as np
15
16      X = in_arr.reshape((in_arr.shape[0] * in_arr.shape[1]))
17      I = np.arange(X.shape[0])
18
19      np.place(I, X == ndval, -1)
20      I = I[I != -1]
```

- *import Numpy*: *NumPy*, significa *Numerical Python*, e é conhecida como uma biblioteca de computação científica construída na linguagem de programação *python*. *Array* é o tipo de dados mais comum de se trabalhar a partir desta biblioteca.

- *reshape*: devolve uma nova forma ao *array* sem alterar os seus dados.

- *shape*: os *arrays numpy* têm um atributo chamado *shape* que geralmente é utilizado para obter a área de um *array*, mas também devolve uma tupla que pode ser usada para alterar as dimensões do *array*.

- *np.arenge*: devolve valores uniformes dentro de um determinado intervalo. O Início do intervalo inclui sempre um valor padrão, o valor é 0.

- *np.place*: altera os elementos de um *array* com base nos valores definidos e atribuídos como entrada.

#Adquire os valores das linhas máximos e mínimos

#Considera as linhas com as células dos dados

```
    # Get Minimum and Maximum Row Indexes
    # considering the rows with cells with data
    i_row = I / in_arr.shape[0]
    i_row = i_row.astype(np.uint32)
    min_row, max_row = i_row.min(), i_row.max()

    # Get Minimum and Maximum Column Indexes
    i_col = I - (i_row * in_arr.shape[1])
    min_col, max_col = i_col.min(), i_col.max()
```

- *astype*: copia o *array* convertido para um tipo específico.

- *np.uint32*: tipo de dado *array* considerado como de 32 bits inteiro.

- *min*: devolve o valor mínimo ao longo de um determinado eixo.

- *max*: devolve o valor máximo ao longo de um determinado eixo.

Descrição:

Código que elemina os valores do *array* de um *raster* para valores *NoData* e ainda atribui novos valores e informações sobre a parte do topo e do lado esquerdo desse *array*. Para além de eliminar a informação do *array*, aproveita os valores dos seus intervalos para atribuir a informação pretendida ao *array* de saída.

Import execmd dentro do código rcls_rst

```
    elif api == "grass":
        from glass.pys import execmd
```

```
def execmd(cmd):
    """
    Execute a command and provide information about the results
    """
    import subprocess

    p = subprocess.Popen(cmd, shell=True,
                         stdout=subprocess.PIPE, stderr=subprocess.PIPE)

    out, err = p.communicate()

    if p.returncode != 0:
        raise ValueError((
            'Message: Command execution ended with error\n'
            f'Command was: {cmd}\n'
            f'Output: {out.decode("utf-8")}\n'
            f'Error: {err.decode("utf-8")}'
        ))

    else:
        return out.decode('utf-8')
```

(Já explicados: *Import* execmd incluído no Programa 1)

Import set_null – *from* glass.rst.rcls *import* set_null

```
[5]: from glass.it.rst   import rst_to_grs, grs_to_rst
     from glass.rst.rcls import rcls_rst, set_null, null_to_value
     from glass.rst.alg  import rstcalc
```

"Atribuição de valores nulos para alguns valores do *raster*"

```
def set_null(rst, value, ascmd=None):
    """
    Null in Raster to Some value
    """

    if not ascmd:
        from grass.pygrass.modules import Module

        m = Module(
            'r.null', map=rst, setnull=value, run_=False, quiet=True
        )

        m()

    else:
        from glass.pys import execmd

        rcmd = execmd(f"r.null map={rst} setnull={value} --quiet")
```

Métodos e Ferramentas:

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*r.null*": atribuição de valores NULOS num *raster*.

Descrição:

Pequeno código que atribui valores NULOS a determinados pixéis dos *raster*.

Imports dentro do set_null:

```
206     else:
207         from glass.pys import execmd
208 
209         rcmd = execmd(f"r.null map={rst} setnull={value} --quiet")
```

Import execmd dentro do código set_null

```
95     elif api == "grass":
96         from glass.pys import execmd
```

```
27 def execmd(cmd):
28     """
29     Execute a command and provide information about the results
30     """
31     import subprocess
32 
33     p = subprocess.Popen(cmd, shell=True,
34                          stdout=subprocess.PIPE, stderr=subprocess.PIPE)
35 
36     out, err = p.communicate()
37 
38     if p.returncode != 0:
39         raise ValueError((
40             'Message: Command execution ended with error\n'
41             f'Command was: {cmd}\n'
42             f'Output: {out.decode("utf-8")}\n'
43             f'Error: {err.decode("utf-8")}'
44         ))
45 
46     else:
47         return out.decode('utf-8')
```

(Já explicados: *Import* execmd incluído no Programa 1)

***Import* null_to_value – *from* glass.rst.rcls *import* null_to_value**

```
[5]: from glass.it.rst   import rst_to_grs, grs_to_rst
     from glass.rst.rcls import rcls_rst, set_null, null_to_value
     from glass.rst.alg  import rstcalc
```

"Aplica um valor numérico às células com valores NULO"

```
def null_to_value(rst, value, as_cmd=None):
    """
    Give a numeric value to the NULL cells
    """

    if not as_cmd:
        from grass.pygrass.modules import Module

        m = Module(
            'r.null', map=rst, null=value, run_=False, quiet=True
        )
        m()

    else:
        from glass.pys import execmd

        rcmd = execmd(f"r.null map={rst} null={value} --quiet")
```

Métodos e Ferramentas:

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*r.null*": atribuição de valores NULOS num *raster*.

Descrição:

Pequeno código que vai dar uma legenda única (valor numérico) às células com valores NULOS. É importante para a posterior máscara binária, onde os valores dos pixéis NULOS terão uma atribuição do seu valor original.

Imports dentro do null_to_value:

```
    else:
        from glass.pys import execmd

        rcmd = execmd(f"r.null map={rst} null={value} --quiet")
```

Import execmd dentro do código null_to_value

```
225     else:
226         from glass.pys import execmd
227 
228         rcmd = execmd(f"r.null map={rst} null={value} --quiet")
```

```
27  def execmd(cmd):
28      """
29      Execute a command and provide information about the results
30      """
31      import subprocess
32  
33      p = subprocess.Popen(cmd, shell=True,
34                           stdout=subprocess.PIPE, stderr=subprocess.PIPE)
35  
36      out, err = p.communicate()
37  
38      if p.returncode != 0:
39          raise ValueError((
40              'Message: Command execution ended with error\n'
41              f'Command was: {cmd}\n'
42              f'Output: {out.decode("utf-8")}\n'
43              f'Error: {err.decode("utf-8")}'
44          ))
45  
46      else:
47          return out.decode('utf-8')
```

(Já explicados: *Import* execmd incluído no Programa 1)

Import rstcalc – *from* glass.rst.alg *import* rstcalc

```
[5]: from glass.it.rst   import rst_to_grs, grs_to_rst
     from glass.rst.rcls import rcls_rst, set_null, null_to_value
     from glass.rst.alg  import rstcalc
```

"*Raster Calculater*"

```
def rstcalc(expression, output, api='saga', grids=None):
    """
    Basic Raster Calculator
    """

    if api == 'saga':
        # Using SAGA GIS

        import os
        from glass.pys      import execmd
        from glass.pys.oss  import fprop
        from glass.it.rst import saga_to_tif

        SAGA_RASTER = os.path.join(
            os.path.dirname(output),
            "sag_{}.sgrd".format(fprop(output, 'fn'))
        )

        cmd = (
            "saga_cmd grid_calculus 1 -FORMULA \"{}\" -GRIDS \"{}\" "
            "-RESULT {} -RESAMPLING 0"
        ).format(
            expression, ";".join(grids), SAGA_RASTER
        )

        outcmd = execmd(cmd)

        # Convert to tiff
        saga_to_tif(SAGA_RASTER, output)

    elif api == 'pygrass':
        from grass.pygrass.modules import Module
        rc = Module(
            'r.mapcalc',
            f'{output} = {expression}',
            overwrite=True, run_=False, quiet=True
        )

        rc()

    elif api == 'grass':
        from glass.pys  import execmd

        rcmd = execmd((
            f"r.mapcalc \"{output} = {expression}\" "
            "--overwrite --quiet"
        ))

    else:
        raise ValueError(f"{api} is not available!")

    return output
```

Métodos e Ferramentas:

#Utilizando o SAGA GIS

```
        # Using SAGA GIS

        import os
        from glass.pys      import execmd
        from glass.pys.oss  import fprop
        from glass.it.rst import saga_to_tif

        SAGA_RASTER = os.path.join(
            os.path.dirname(output),
            "sag_{}.sgrd".format(fprop(output, 'fn'))
        )
```

- *import* OS: o módulo OS em *Python* fornece funções para interagir com o sistema operacional. O sistema operacional vem sob os módulos padrão do *Python*. Este módulo fornece uma maneira flexível de usar a funcionalidade dependendo do sistema

operacional. Os módulos "OS" e "os.path" incluem muitas funções para interagir com o sistema de pastas.

- *os.path.join*: o módulo os.path é um submódulo do módulo OS em python usado para manipulação comum de nomes de caminho. O método os.path.join() em python une um ou mais componentes de caminho de forma inteligente. Este método concatena vários componentes de caminho com exatamente um separador de diretório ('/') após cada parte não vazia, exceto o último componente do caminho. Exemplo: '/home/glass/francisco/'.

- *os.path.dirname*: método em *python* utilizado para obter o nome especifico da pasta do caminho.

- *format*: método que formata o valor detalhado e insere-o dentro do espaço reservado da *string*. O método devolve a *string* formatada. SGRD é o formato de dados de grelhas binárias SAGA.

```
        cmd = (
            "saga_cmd grid_calculus 1 -FORMULA \"{}\" -GRIDS \"{}\" "
            "-RESULT {} -RESAMPLING 0"
        ).format(
            expression, ";".join(grids), SAGA_RASTER
        )
```

- *join*: função utilizado para unir todos os elementos num iterável novo.

#Converção para Tiff

```
        # Convert to tiff
        saga_to_tif(SAGA_RASTER, output)

    elif api == 'pygrass':
        from grass.pygrass.modules import Module

        rc = Module(
            'r.mapcalc',
            f'{output} = {expression}',
            overwrite=True, run_=False, quiet=True
        )

        rc()
```

- *Import Module*: importa para o *script* todos os módulos e ferramentas do *software* GRASS GIS.

- "*r.mapcal*": Calculadora de *rasters*.

Descrição:

Código que utiliza o SAGA GIS para a transformação do *raster* em um formato de dados de grelhas binárias para a discriminação dos dois níveis de pixéis. Posteriormente aplica a fórmula e realiza a conversão do formato do *raster* SGRD para *Tiff*. Esta tarefa também pode ser feita através da importação dos módulos do GRASS GIS para aceder à calculadora de

rasters para a atribuição dos valores aos pixéis do *raster* de saída. A execução do exercício pode seguir um caminho com acesso ao SAGA GIS ou pelo caminho do GRASS GIS.

Imports dentro do rstcalc:

```
48      if api == 'saga':
49          # Using SAGA GIS
50
51          import os
52          from glass.pys       import execmd
53          from glass.pys.oss   import fprop
54          from glass.it.rst import saga_to_tif

84      elif api == 'grass':
85          from glass.pys  import execmd
86
```

***Import* execmd dentro do código rstcalc**

```
48      if api == 'saga':
49          # Using SAGA GIS
50
51          import os
52          from glass.pys       import execmd
53          from glass.pys.oss   import fprop
54          from glass.it.rst import saga_to_tif

84      elif api == 'grass':
85          from glass.pys  import execmd
86
```

```
27  def execmd(cmd):
28      """
29      Execute a command and provide information about the results
30      """
31      import subprocess
32
33      p = subprocess.Popen(cmd, shell=True,
34                           stdout=subprocess.PIPE, stderr=subprocess.PIPE)
35
36      out, err = p.communicate()
37
38      if p.returncode != 0:
39          raise ValueError((
40              'Message: Command execution ended with error\n'
41              f'Command was: {cmd}\n'
42              f'Output: {out.decode("utf-8")}\n'
43              f'Error: {err.decode("utf-8")}'
44          ))
45
46      else:
47          return out.decode('utf-8')
```

(Já explicados: *Import* execmd incluído no Programa 1)

Import fprop dentro do código rstcalc

```
    if api == 'saga':
        # Using SAGA GIS

        import os
        from glass.pys      import execmd
        from glass.pys.oss  import fprop
        from glass.it.rst import saga_to_tif
```

```
def fprop(__file, prop, forceLower=None, fs_unit=None):
    """
    Return some property of file

    prop options:
    * filename or fn - return filename
    """

    from glass.pys import obj_to_lst

    prop = obj_to_lst(prop)

    result = {}

    if 'filename' in prop or 'fn' in prop:
        fn, ff = os.path.splitext(os.path.basename(__file))

        result['filename'] = fn if not forceLower else fn.lower()

        if 'fileformat' in prop or 'fn' in prop:
            result['fileformat'] = ff

    elif 'fileformat' in prop or 'ff' in prop:
        result['fileformat'] = os.path.splitext(__file)[1]

    if 'filesize' in prop or 'fs' in prop:
        fs_unit = 'MB' if not fs_unit else fs_unit

        fs = os.path.getsize(__file)

        if fs_unit == 'MB':
            fs  = (fs / 1024.0) / 1024

        elif fs_unit == 'KB':
            fs = fs / 1024.0

        result['filesize'] = fs

    if len(prop) == 1:
        if prop[0] == 'fn':
            return result['filename']
        elif prop[0] == 'ff':
            return result['fileformat']
        elif prop[0] == 'fs':
            return result['filesize']
        else:
            return result[prop[0]]
    else:
        return result

```

(Já explicados: *Import* fprop incluído no Programa 1)

Import saga_to_tif dentro do código rstcalc

```
48      if api == 'saga':
49          # Using SAGA GIS
50
51          import os
52          from glass.pys      import execmd
53          from glass.pys.oss  import fprop
54          from glass.it.rst import saga_to_tif
```

"Formato SAGA GIS para GeoTIFF"

```
365 def saga_to_tif(inFile, outFile):
366     """
367     SAGA GIS format to GeoTIFF
368     """
369
370     from glass.pys     import execmd
371     from glass.pys.oss import fprop
372
373     # Check if outFile is a GeoTiff
374     if fprop(outFile, 'ff') != '.tif':
375         raise ValueError(
376             'Outfile should have GeoTiff format'
377         )
378
379     cmd = (
380         "saga_cmd io_gdal 2 -GRIDS {} "
381         "-FILE {}"
382     ).format(inFile, outFile)
383
384     outcmd = execmd(cmd)
385
386     return outFile
```

Métodos e Ferramentas:

- *format*: método que formata o valor detalhado e insere-o dentro do espaço reservado da *string*. O método devolve a *string* formatada. SGRD é o formato de dados de grelhas binárias SAGA.

Descrição:

Este código realiza a simples conversão do formato de dados SAGA GIS para formato de dados GeoTIFF. Para isso o código certifica-se de que o ficheiro de saída vai ter a extensão *.tif* e indica o caminho para a sua pasta.

Imports dentro do saga_to_tif:

```
365 def saga_to_tif(inFile, outFile):
366     """
367     SAGA GIS format to GeoTIFF
368     """
369
370     from glass.pys     import execmd
371     from glass.pys.oss import fprop
```

Import execmd dentro do código saga_to_tif

```
365 def saga_to_tif(inFile, outFile):
366     """
367     SAGA GIS format to GeoTIFF
368     """
369 
370     from glass.pys     import execmd
371     from glass.pys.oss import fprop
```

```
27 def execmd(cmd):
28     """
29     Execute a command and provide information about the results
30     """
31     import subprocess
32 
33     p = subprocess.Popen(cmd, shell=True,
34                          stdout=subprocess.PIPE, stderr=subprocess.PIPE)
35 
36     out, err = p.communicate()
37 
38     if p.returncode != 0:
39         raise ValueError((
40             'Message: Command execution ended with error\n'
41             f'Command was: {cmd}\n'
42             f'Output: {out.decode("utf-8")}\n'
43             f'Error: {err.decode("utf-8")}'
44         ))
45 
46     else:
47         return out.decode('utf-8')
```

(Já explicados: *Import* execmd incluído no Programa 1)

Import fprop dentro do código saga_to_tif

```
365 def saga_to_tif(inFile, outFile):
366     """
367     SAGA GIS format to GeoTIFF
368     """
369 
370     from glass.pys     import execmd
371     from glass.pys.oss import fprop
```

```
25 def fprop(__file, prop, forceLower=None, fs_unit=None):
26     """
27     Return some property of file
28 
29     prop options:
30     * filename or fn - return filename
31     """
32 
33     from glass.pys import obj_to_lst
34 
35     prop = obj_to_lst(prop)
36 
37     result = {}
38 
39     if 'filename' in prop or 'fn' in prop:
40         fn, ff = os.path.splitext(os.path.basename(__file))
41 
42         result['filename'] = fn if not forceLower else fn.lower()
43 
44         if 'fileformat' in prop or 'fn' in prop:
45             result['fileformat'] = ff
46 
47     elif 'fileformat' in prop or 'ff' in prop:
48         result['fileformat'] = os.path.splitext(__file)[1]
49 
50     if 'filesize' in prop or 'fs' in prop:
51         fs_unit = 'MB' if not fs_unit else fs_unit
52 
53         fs = os.path.getsize(__file)
54 
55         if fs_unit == 'MB':
56             fs  = (fs / 1024.0) / 1024
57 
58         elif fs_unit == 'KB':
59             fs = fs / 1024.0
60 
61         result['filesize'] = fs
62 
63     if len(prop) == 1:
64         if prop[0] == 'fn':
65             return result['filename']
66         elif prop[0] == 'ff':
67             return result['fileformat']
68         elif prop[0] == 'fs':
69             return result['filesize']
70         else:
71             return result[prop[0]]
72     else:
73         return result
74 
```

(Já explicados: *Import* fprop incluído no Programa 1)

Anexo D3 – Programa 3: *Atmosferic and topografhic Landsat-8 images correction* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para realizar a correções atmosféricas e topográficas das imagens Landsat-8, com recurso a funções do repositório GLASS e de ferramentas do *software* GRASS GIS.

Atmosferic and topografhic Landsat-8 images correction

In [ ]:
```
path = '/home/francisco/create_grid/lds_8/2020_09_15_terceira_lc8'
extes = '/home/francisco/create_grid/lds_8/2020_01_03_terceira_lc8/extensao/extensao.TIF'
mtl = '/home/francisco/create_grid/lds_8/2020_09_15_terceira_lc8/LC08_L1TP_216033_20200915_20200923_02_T1_MTL.txt'
mde = '/home/francisco/create_grid/modelo/step1_mde_nearest_lds8/dem_nearest_hyd_lds8_2.tif'
output= '/home/francisco/create_grid/lds_8/step1_atm_corr_20200915'
```

In [ ]:
```
import os

from glass.pys.oss import mkdir, fprop, lst_ff
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str
from glass.rst.alg  import rstcalc
```

In [ ]:
```
# Check if outfolder exists
out_folder = os.path.dirname(output)
if not os.path.exists(out_folder):
    mkdir(out_folder, overwrite=None)

"""
Start GRASS GIS Session
"""

loc = f'loc_{now_as_str()}'
grsb = run_grass(
    out_folder, grassBIN='grass78', location=loc,
    srs= extes
    )

import grass.script.setup as gsetup

gsetup.init(grsb, out_folder, loc, 'PERMANENT')
```

In [ ]:
```
from glass.it.rst import rst_to_grs, grs_to_rst
from glass.wenv.grs import rst_to_region
from grass.pygrass.modules import Module
```

In [ ]:
```
#set region
gregion = rst_to_grs(extes, fprop(extes, 'fn'))
rst_to_region(gregion)
```

In [ ]:
```
#Serching in the path of lds8 bands all files with '.tif' format
bands = lst_ff(path, file_format='.TIF')

#Ordinating bands in fuction of your band value
ord_bands = sorted(bands, key=lambda x: int(x.split('_B')[-1].split('.')[0]))
```

In [ ]:
```
final_dict = {}
for i, ord_bands in enumerate(ord_bands, start=1):
    key = f"band.{i}"
    final_dict[key] = ord_bands
```

In [ ]:
```
for key in final_dict:
    value = final_dict[key]
    grsRst = key  # Use the key as the output name
    final_dict[key] = []  # Transform the value into an empty list
    final_dict[key].append(rst_to_grs(value, grsRst))
```

In [ ]:
```
#Aplication i.landsat.toar tool
rc = Module('i.landsat.toar', input='band.', output='toar.',
            metfile=mtl, sensor='oli8', overwrite=True, run_=True, quiet=True)
```

In [ ]:
```
#Input DEM in Grass Session
dem = rst_to_grs(mde, fprop(mde, 'fn'))

#date_lds8_bands:20200103
#zenith- 90 - 25.12328562 = 64.87671438
#azimuth- 158.34677522

#date_lds8_bands:20200915
#zenith- 90 - 49.91752450 = 40.0824755
#azimuth- 149.75932387

#1 i.topo.corr first the terrain lighting model from the DEM
rc = Module('i.topo.corr', flags='i', output='dem.illu',
            basemap= dem, zenith='40.0824755', azimuth='149.75932387',
            overwrite=True, run_=True, quiet=True)

#out_top = []
#2 i.topo.corr second the topographic correction
rc = Module('i.topo.corr', input='toar.1,toar.2,toar.3,toar.4,toar.5,toar.6,toar.7,toar.8,toar.9,toar.10,toar.11',
            output='topo.', basemap='dem.illu', zenith='40.0824755', method='minnaert',
            overwrite=True, run_=True, quiet=True)
```

In [ ]:
```
#r.mapcalc all bands *10 000 to equal information at sentinel-2 bands and convert values to integers
rast_bands = ['topo..toar.1', 'topo..toar.2', 'topo..toar.3', 'topo..toar.4',
              'topo..toar.5', 'topo..toar.6', 'topo..toar.7', 'topo..toar.8',
              'topo..toar.9', 'topo..toar.10', 'topo..toar.11']
final_bands = []

for r in rast_bands:
    form = f'int(round({r} * 10000))'
    outband = f'band_{r.split(".")[-1]}'  # Extract the band number from the r string

    rc = Module('r.mapcalc', expression=f'{outband} = {form}', overwrite=True, run_=True, quiet=True)
    final_bands.append(outband)

print(final_bands)
```

In [ ]:
```
#Export all band to path
for f in final_bands:
    inpt_raster = f
    grs_to_rst(inpt_raster, os.path.join(output, inpt_raster + '.tif'), is_int=True)
```

Anexo D4 – Programa 4: *Resampling Landsat-8 images to a 10 meter spatial resolution* (código existente e adaptado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para realizar a reamostragem do tamanho do pixel das imagens Landsat-8 para 10 metros e recorte pelo limite da área de estudo, adaptado do Programa 1 disponibilizado pela orientação. É apresentado ainda as funções atualizadas que permitiram a execução do código.

Resampling Landsat-8 images to a 10 meter spatial resolution

In [1]:
```
imgs_lds ='/home/francisco/create_grid/script5_terceira/'
outfolder ='/home/francisco/create_grid/script5_terceira/step3_ld8'

refrst = '/home/francisco/create_grid/lds_8/bnd_ref/b02_20200220.tif'
```

In [2]:
```
import os

from glass.pys.oss  import lst_ff, mkdir
from glass.pys.tm   import now_as_str
from terceira.new_scripts.ldst8_process.get_ibands_lds8.uzip  import unzip_img
from glass.rst.rmp  import match_cellsize_and_clip
```

In [3]:
```
imgs = lst_ff(imgs_lds, file_format='.zip')
```

In [4]:
```
# Image Resampling

for img in imgs:
    # Create folder for temp files
    zfolder = mkdir(os.path.join(outfolder, now_as_str()))

    # Unzip images
    bands = unzip_img(img, zfolder)
    # Match cellsize and clip
    bands = match_cellsize_and_clip(
        [bands[b] for b in bands],
        bands["B1"], outfolder,
        isint=True, clipgeo=refrst,
        ws=zfolder
    )
```

Blocos de código, em *python*, das funções atualizadas para implementação do Programa 4.

```python
def unzip_img(zip_file, out_folder):
    """
    Unzip Landsat-8 Image
    """

    import os
    from zipfile        import ZipFile
    from terceira.new_scripts.ldst8_process.get_ibands_lds8.lds8 import get_l8bands
    from glass.pys.oss  import copy_file

    intbands = get_l8bands()


    with ZipFile(zip_file, 'r') as zipo:
        zipff = zipo.namelist()

        rbands = {}
        for b in intbands:
            for f in zipff:
                if b in f:
                    ob = os.path.join(out_folder, os.path.basename(f))

                    zipo.extract(f, out_folder, pwd=None)
                    copy_file(
                        os.path.join(out_folder, f), ob
                    )
                    rbands[b] = ob

                    break

    return rbands
```

```python
def get_l8bands():
    return [
        'B1', 'B2', 'B3', 'B4',
        'B5', 'B6', 'B7', 'B8',
        'B9', 'B10', 'B11'
    ]

def get_lwl8bands():
    return [
        'b1', 'b2', 'b3', 'b4',
        'b5', 'b06', 'b7', 'b8',
        'b9', 'b10', 'b11', 'b12'
    ]
```

Anexo D5 – Programa 5: *Projection and resampling MDE to a 10 meter spatial resolution* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para realizar a projeção de sistema de coordenadas, reamostragem do tamanho do pixel, suavização das elevações e recorte pelo limite da área de estudo do MDE.

Projection and resampling MDE to a 10 meter spatial resolution

In [ ]:
```
img_mde ='/home/francisco/create_grid/modelo/mde/eu_dem_v11_E10N20.TIF'
out ='/home/francisco/create_grid/modelo/step1_mde_nearest/dem_nearest_hyd_stl2.tif'

refrst = '/home/francisco/create_grid/lds_8/band_ref_buff/bff_lmt_20km.tif'
reg = '/home/francisco/create_grid/lds_8/bnd_ref/b02_20200220.tif'

meth_int = 'nearest' #or 'bicubic'
```

In [ ]:
```
import os

from glass.pys.oss import mkdir, fprop
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str
```

In [ ]:
```
# Check if outfolder exists
out_folder = os.path.dirname(out)
if not os.path.exists(out_folder):
    mkdir(out_folder, overwrite=None)

"""
Start GRASS GIS Session
"""

loc = f'loc_{now_as_str()}'
grsb = run_grass(
    out_folder, grassBIN='grass78', location=loc,
    srs=refrst
    )

import grass.script.setup as gsetup

gsetup.init(grsb, out_folder, loc, 'PERMANENT')
```

In [ ]:
```
from glass.wenv.grs import rst_to_region
from glass.it.rst import grs_to_rst, grs_to_mask, rst_to_grs
from grass.pygrass.modules import Module
```

In [ ]:
```
#set region
rstref = rst_to_grs(refrst, fprop(refrst, 'fn'))
rst_to_region(rstref)
```

In [ ]:
```
#import mde(r.import)
inpt = fprop(out, 'fn')
m = Module(
    "r.import", input= img_mde, output= inpt,
    extent= 'region', resolution= 'region',
    overwrite=True, run_=True, quiet=True
)
```

In [ ]:
```
#projection to srs of s2 image
proj = fprop(out, 'fn')
m = Module(
    "r.proj", input= inpt, output= proj,
    location= loc, mapset='PERMANENT',
    resolution= '10', #method= 'bicubic',
    overwrite=True, run_=True, quiet=True
)
```

In [ ]:
```
#resamp.interp to s2 image resolution
res = fprop(out, 'fn')
m = Module(
    "r.resamp.interp", input= proj, output= res,
    method= meth_int,
    overwrite=True, run_=True, quiet=True
)
```

In [ ]:
```
#r.hydrodem to soften mde values

hyd = fprop(out, 'fn')
m = Module(
    "r.hydrodem", input= res, output= hyd,
    mod= 4, size= 4,
    overwrite=True, run_=True, quiet=True
)
```

In [ ]:
```
#r.in.gdal b2
btwo = rst_to_grs(reg, fprop(reg, 'fn'))
```

In [ ]:
```
#g.region to b2 extension
rst_to_region(btwo)
```

In [ ]:
```
#set mask
grs_to_mask(btwo)
```

In [ ]:
```
grs_to_rst(hyd, out)#, is_int=True)
```

Anexo D6 – Programa 6: *Split area* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para pegar numa banda do Sentinel-2 e realizar a divisão em blocos mais pequenos com dimensões de 500x500 pixéis.

Split area

In [ ]:
```
img = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/file_rst/b02_20200220.tif'

rowspx, colspx = 500, 500

out = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_rsts'

bname = 'terceira'
```

In [ ]:
```
import os
from osgeo import gdal, ogr, osr
import numpy as np
from shapely.geometry import Polygon
import geopandas as gp
```

In [ ]:
```
src = gdal.Open(img)
```

In [ ]:
```
proj = osr.SpatialReference(wkt=src.GetProjection())

epsg = int(str(proj.GetAttrValue('AUTHORITY', 1)))
```

In [ ]:
```
left, cellx, _, top, __, celly = src.GetGeoTransform()
```

In [ ]:
```
ncols = src.RasterXSize
nrows = src.RasterYSize
```

In [ ]:
```
right = left + (ncols * cellx)
bottom = top + (nrows * celly)
```

In [ ]:
```
npartescols = ncols / colspx
npartescols = int(npartescols) if npartescols == int(npartescols) else int(npartescols) + 1

npartesrows = nrows / rowspx
npartesrows = int(npartesrows) if npartesrows == int(npartesrows) else int(npartesrows) + 1
```

In [ ]:

```
min_x = left
max_y = top
for r in range(npartesrows):
    min_y = max_y + (celly * rowspx)

    if min_y < bottom:
        min_y = bottom

    for c in range(npartescols):
        max_x = min_x + (cellx * colspx)

        if max_x > right:
            max_x = right

        # Create bounding box polygon
        points = [
            # Top left
            (min_x, max_y),
            # Top Right
            (max_x, max_y),
            # Bootom Right
            (max_x, min_y),
            # Bottom left
            (min_x, min_y),
            # Top Left
            (min_x, max_y)
        ]

        polygon = Polygon(points)

        gdf = gp.GeoDataFrame([[1]], columns=['pid'], crs=f"EPSG:{str(epsg)}", geometry=[polygon])

        gdf.to_file(os.path.join(out, f'{bname}_{str(r)}_{str(c)}.shp'))

        # Create Raster
        _rows = (max_y - min_y) / abs(celly)
        _cols = (max_x - min_x) / cellx
```

```
        _rows = int(_rows) if _rows == int(_rows) else int(_rows) + 1
        _cols = int(_cols) if _cols == int(_cols) else int(_cols) + 1

        rstval = np.full((_rows, _cols), 1)

        rstpath = os.path.join(out, f'{bname}_{str(r)}_{str(c)}.tif')

        nimg = gdal.GetDriverByName('GTiff').Create(
            rstpath, _cols, _rows, 1, gdal.GDT_Byte
        )

        nimg.SetGeoTransform((min_x, cellx, 0, max_y, 0, celly))

        band = nimg.GetRasterBand(1)

        band.WriteArray(rstval)

        rstSrs = osr.SpatialReference()
        rstSrs.ImportFromEPSG(epsg)
        nimg.SetProjection(rstSrs.ExportToWkt())

        band.FlushCache()

        min_x = min_x + (cellx * colspx)

    max_y = max_y + (celly * rowspx)
    min_x = left
```

Anexo D7 – Programa 7: *Make grid* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para converter os pequenos blocos *raster* em grelhas vetoriais.

Make grid

In [ ]:
```
import multiprocessing as mp
from glass.pys.oss import lst_ff


import os
from glass.pys.oss import mkdir, fprop
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str

from glass.it.shp import grs_to_shp
from glass.it.rst import rst_to_grs
```

In [ ]:
```
refs = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_rsts'

out_grid = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_shp'

bname = 'terceira_fish'
nproc = 4
```

In [ ]:
```
rsts = lst_ff(refs, file_format='.tif')
```

In [ ]:
```
dimlist = len(rsts) / nproc

dimlist = int(dimlist) if dimlist == int(dimlist) else int(dimlist) + 1
```

In [ ]:
```
process_data = []
for e in range(0, nproc*dimlist, dimlist):
    process_data.append(rsts[e:e+dimlist])
```

In [ ]:

```
def create_fishnet(th, idx, raster, out_shp):
    #print(f'Process {str(th)} - {raster}')

    # Check if outfolder exists
    out_folder = os.path.dirname(out_shp)
    if not os.path.exists(out_folder):
        mkdir(out_folder, overwrite=None)

    """
    Start GRASS GIS Session
    """

    loc = f'loc_{now_as_str()}_{str(th)}_{str(idx)}'
    #loc = f'loc_{now_as_str()}'
    grsb = run_grass(
        out_folder, grassBIN='grass78', location=loc,
        srs=raster
    )

    import grass.script.setup as gsetup

    gsetup.init(grsb, out_folder, loc, 'PERMANENT')

    from grass.pygrass.modules import Module

    bloc = rst_to_grs(raster, fprop(raster, 'fn'))

    grsgrid = fprop(out_shp, 'fn')

    m = Module('v.mkgrid', map=grsgrid, overwrite=True, run_=True, quiet=True)

    out_grid = grs_to_shp(grsgrid, out_shp, 'area')
```

```
def run_multiproc(th, rasters, out_grid, name):
    for _r in range(len(rasters)):
        create_fishnet(
            th, _r, rasters[_r],
            os.path.join(out_grid, f'{name}_{str(th)}_{str(_r)}.shp'))

thrds = []

for i in range(len(process_data)):
    processo = mp.Process(
        target=run_multiproc, name=f'th-{str(i+1)}',
        args=(i+1, process_data[i], out_grid, bname)
    )

    thrds.append(processo)

for p in thrds:
    p.start()

for p in thrds:
    p.join()

print('terminou')
```

Anexo D8 – Programa 8: *Intersection of grids with* OSM *classes* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para intersetar as diferentes grelhas vetoriais com os dados recolhidos do OSM, criar uma coluna na tabela de atributos de cada ficheiro produzido da interseção e ainda calcular a área ocupada de cada classe nas células das grelhas.

Intersection of grids with OSM classes

In [ ]:
```
import multiprocessing as mp
from glass.pys.oss import lst_ff


import os
from glass.pys.oss import mkdir, fprop
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str

from glass.it.shp import grs_to_shp
from glass.it.rst import rst_to_grs
from glass.it.shp import shp_to_grs
```

In [ ]:
```
refs = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_shp'

lulc = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_lulc'

out_grid = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/int_shps_lulc'

bname = 'terceira_intrs'
nproc = 4


#File 4_2 does not include data in the attribute table, because the grid area does not intersect with the LULC classes
```

In [ ]:
```
rsts = lst_ff(refs, file_format='.shp')
classe = lst_ff(lulc, file_format='.shp')
```

In [ ]:
```
intrsect = []
for r in rsts:
    re = fprop(r, 'fn')

    refsplit = re.split('_')

    idx, idy = refsplit[-2], refsplit[-1]

    for l in classe:
        le = fprop(l, 'fn')

        dsplit = le.split('_')

        _idx, _idy = dsplit[-2], dsplit[-1]

        intrsect.append({
            'fish' : r, 'lc' : l,
            'out' : os.path.join(out_grid, f'{bname}_{str(idx)}_{str(idy)}.shp')
        })
        break
```

In [ ]:
```
dimlist = len(intrsect) / nproc

dimlist = int(dimlist) if dimlist == int(dimlist) else int(dimlist) + 1
```

In [ ]:
```
process_data = []
for e in range(0, nproc*dimlist, dimlist):
    process_data.append(intrsect[e:e+dimlist])
```

In [ ]:

```
def create_fishnet(th, raster, classes, out_shp):

    # Check if outfolder exists
    out_folder = os.path.dirname(out_shp)
    if not os.path.exists(out_folder):
        mkdir(out_folder, overwrite=None)

    """
    Start GRASS GIS Session
    """

    loc = f'loc_{now_as_str()}_{str(th)}_{str(idx)}'
    grsb = run_grass(
        out_folder, grassBIN='grass78', location=loc,
        srs=raster
    )

    import grass.script.setup as gsetup

    gsetup.init(grsb, out_folder, loc, 'PERMANENT')

    from grass.pygrass.modules import Module

    bloc = shp_to_grs(raster, fprop(raster, 'fn'))
    cat = shp_to_grs(classes, fprop(classes, 'fn'))

    intsct = fprop(out_shp, 'fn')
    m = Module("v.overlay", ainput=bloc, binput=cat,
               operator='and', output=intsct,
               overwrite=True, run_=True, quiet=True
              )


    col = fprop(intsct, 'fn')
    m = Module("v.db.addcolumn", map=col, columns='area_ocp double precision',
               run_=True, quiet=True
              )
```

```
    size = fprop(col, 'fn')
    m = Module("v.to.db", map=size, option='area', columns='area_ocp', units='meters',
               overwrite=True, run_=True, quiet=True
              )

    out_grid = grs_to_shp(size, out_shp, 'area')

def run_multiproc(_th, _intrsect):
    for i in _intrsect:
        create_fishnet(_th, i['fish'], i['lc'], i['out'])

thrds = []

for i in range(len(process_data)):
    processo = mp.Process(
        target=run_multiproc, name=f'th-{str(i+1)}',
        args=(i+1, process_data[i])
    )

    thrds.append(processo)

for p in thrds:
    p.start()

for p in thrds:
    p.join()

print('terminou')
```

Anexo D9 – Programa 9: *Dissolve with* GDAL/OGR (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para a agregação da informação comum da tabela de atributos depois da interseção realizada entre as grelhas e camada vetorial da informação OSM.

Dissolve with GDAL/OGR

In [ ]:
```
import multiprocessing as mp


import os
from glass.pys.oss import lst_ff
from glass.pys import execmd
from glass.pys.oss import fprop
from glass.prop import drv_name
```

In [ ]:
```
refs = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/int_shps_lulc'

out_grid = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/gdal_terceira'


bname = 'terceira_gdal'
nproc = 4
#Since the Grass Gis tool doesn't perform Dissolve in the same way as the ArcGis tool (as was intended), it was necessary to use the Gdal library for this purpose.
```

In [ ]:
```
rsts = lst_ff(refs, file_format='.shp')

idcol = 'a_cat'
area_col = 'area_ocp'
```

In [ ]:
```
sorted_rsts = sorted(rsts)
```

In [ ]:
```
dimlist = len(sorted_rsts) / nproc

dimlist = int(dimlist) if dimlist == int(dimlist) else int(dimlist) + 1
```

In [ ]:
```
process_data = []
for e in range(0, nproc*dimlist, dimlist):
    process_data.append(sorted_rsts[e:e+dimlist])
```

In [ ]:
```
def create_fishnet(th, idx, raster, out_shp):

    shpname = fprop(raster, 'fn')
    drv = drv_name(out_shp)

    query = (
        f"SELECT {idcol}, ST_Union(geometry) AS GEOMETRY, "
        f"COUNT({idcol}) AS cclasse, MAX({area_col}) AS maxarea "
        f"FROM {shpname} "
        f"GROUP BY {idcol};"
    )

    cmd = (
        f'ogr2ogr -f "{drv}" {out_shp} {raster} '
        f'-dialect sqlite -sql "{query};"'
    )

    rcmd = execmd(cmd)
```

```
def run_multiproc(th, rasters, out_grid, name):
    for _r in range(len(rasters)):
        create_fishnet(
            th, _r, rasters[_r],
            os.path.join(out_grid, f'{name}_{str(th)}_{str(_r)}.shp'))

thrds = []

for i in range(len(process_data)):
    processo = mp.Process(
        target=run_multiproc, name=f'th-{str(i+1)}',
        args=(i+1, process_data[i], out_grid, bname)
    )

    thrds.append(processo)

for p in thrds:
    p.start()

for p in thrds:
    p.join()

print('terminou')
```

Anexo D10 – Programa 10: *Selection and extraction of attributes with 1 class and total occupied area* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para selecionar e extrair informações da tabela de atributos, onde para cada atributo associa-se apenas uma classe de ocupação do solo com uma área total de ocupação da célula da grelha.

Selection and extraction of attributes with 1 class and total occupied area

In [ ]:
```
import multiprocessing as mp
from glass.pys.oss import lst_ff


import os
from glass.pys.oss import mkdir, fprop
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str

from glass.it.shp import grs_to_shp
from glass.it.rst import rst_to_grs
from glass.it.shp import shp_to_grs
```

In [ ]:
```
refs = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_shp_23'

lulc = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/gdal_terceira'

out_grid = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/join_terceira'


bname = 'terceira_join'
nproc = 4
```

In [ ]:
```
rsts = lst_ff(refs, file_format='.shp')
diss = lst_ff(lulc, file_format='.shp')
```

In [ ]:
```
data = []
for ref in rsts:
    refname = fprop(ref, 'fn')

    refsplit = refname.split('_')

    idx, idy = refsplit[-2], refsplit[-1]
```

```
    for shp in diss:
        dname = fprop(shp, 'fn')

        dsplit = dname.split('_')

        _idx, _idy = dsplit[-2], dsplit[-1]

        if idx == _idx and idy == _idy:
            data.append({
                'REF' : ref, 'DISS': shp,
                'OUT' : os.path.join(out_grid, f'{bname}_{str(idx)}_{str(idy)}.shp')
            })
            break
```

In [ ]:
```
dimlist = len(dimlist = len(data) / nproc

dimlist = int(dimlist) if dimlist == int(dimlist) else int(dimlist) + 1
```

In [ ]:
```
process_data = []
for e in range(0, nproc*dimlist, dimlist):
    process_data.append(data[e:e+dimlist])
```

In [ ]:
```
def do_join(th, ref_shp, diss_shp, out_shp):

    # Check if outfolder exists
    out_folder = os.path.dirname(out_shp)
    if not os.path.exists(out_folder):
        mkdir(out_folder, overwrite=None)

    """
    Start GRASS GIS Session
    """

    loc = f'loc_{now_as_str()}_{str(th)}_{str(idx)}'

    grsb = run_grass(
        out_folder, grassBIN='grass78', location=loc,
        srs=ref_shp
    )

    import grass.script.setup as gsetup
```

```
    gsetup.init(grsb, out_folder, loc, 'PERMANENT')

    from grass.pygrass.modules import Module

    bloc = shp_to_grs(ref_shp, fprop(ref_shp, 'fn'))
    dss = shp_to_grs(diss_shp, fprop(diss_shp, 'fn'))

    m = Module(
        "v.db.join", map=bloc, column='cat', other_table=dss,
        other_column='a_cat', subset_columns='cclasse, maxarea',
        run_=True, quiet=True
    )

    extrat = fprop(out_shp, 'fn')
    m = Module(
        "v.extract", input=dss, where='cclasse=1 and maxarea=100',
        output=extrat, overwrite=True, run_=True, quiet=True
    )

    out_grid = grs_to_shp(extrat, out_shp, 'area')

def run_multiproc(_th, _data):
    for d in _data:
        do_join(_th, d['REF'], d['DISS'], d['OUT'])

thrds = []

for i in range(len(process_data)):
    processo = mp.Process(
        target=run_multiproc, name=f'th-{str(i+1)}',
        args=(i+1, process_data[i])
    )

    thrds.append(processo)

for p in thrds:
    p.start()

for p in thrds:
    p.join()

print('terminou')
```

Anexo DII – Programa II: *Updating class legend values* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para realizar a segunda interseção entre os ficheiros obtidos até à presente etapa e a camada vetorial das classes OSM, de forma a alcançar informações sobre o valor da legenda e descrição de cada classe, eliminar campos das tabelas de atributos irrelevantes, criar um campo do tipo *double precision* e atualizar a respetiva coluna com os valores da legenda de cada classe.

Updating class legend values

In [ ]:
```
import multiprocessing as mp
from glass.pys.oss import lst_ff


import os
from glass.pys.oss import mkdir, fprop
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str

from glass.it.shp import grs_to_shp
from glass.it.rst import rst_to_grs
from glass.it.shp import shp_to_grs
```

In [ ]:
```
refs = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/join_terceira'

lulc = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_lulc'

out_grid = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/update_terceira'

bname = 'terceira_update'
nproc = 4
```

In [ ]:
```
rsts = lst_ff(refs, file_format='.shp')
classe = lst_ff(lulc, file_format='.shp')
```

In [ ]:
```
intrsect = []
for r in rsts:
    re = fprop(r, 'fn')

    refsplit = re.split('_')

    idx, idy = refsplit[-2], refsplit[-1]

    for l in classe:
        le = fprop(l, 'fn')
```

```
        dsplit = le.split('_')

        _idx, _idy = dsplit[-2], dsplit[-1]

        intrsect.append({
            'fish' : r, 'lc' : l,
            'out' : os.path.join(out_grid, f'{bname}_{str(idx)}_{str(idy)}.shp')
        })
        break
```

In [ ]:
```
dimlist = len(intrsect) / nproc

dimlist = int(dimlist) if dimlist == int(dimlist) else int(dimlist) + 1
```

In [ ]:
```
process_data = []
for e in range(0, nproc*dimlist, dimlist):
    process_data.append(intrsect[e:e+dimlist])
```

In [ ]:
```
def create_fishnet(th, raster, classes, out_shp):

    # Check if outfolder exists
    out_folder = os.path.dirname(out_shp)
    if not os.path.exists(out_folder):
        mkdir(out_folder, overwrite=None)

    """
    Start GRASS GIS Session
    """

    loc = f'loc_{now_as_str()}_{str(th)}_{str(idx)}'
    grsb = run_grass(
        out_folder, grassBIN='grass78', location=loc,
        srs=raster
    )

    import grass.script.setup as gsetup

    gsetup.init(grsb, out_folder, loc, 'PERMANENT')

    from grass.pygrass.modules import Module
```

```
    bloc = shp_to_grs(raster, fprop(raster, 'fn'))
    cat = shp_to_grs(classes, fprop(classes, 'fn'))

    intsct = fprop(out_shp, 'fn')
    m = Module("v.overlay", ainput=bloc, binput=cat,
               operator='and', output=intsct,
               overwrite=True, run_=True, quiet=True
              )


    intsct = fprop(out_shp, 'fn')
    m = Module("v.db.dropcolumn", map=intsct,
               columns=['a_cat', 'a_cat_', 'a_cat__1', 'a_a_cat', 'a_cls', 'b_cat', 'b_cat_'],
               run_=True, quiet=True
              )


    intsct = fprop(out_shp, 'fn')
    m = Module("v.db.addcolumn", map=intsct,
               columns='cls double precision',
               run_=True, quiet=True
              )

    intsct = fprop(out_shp, 'fn')
    m = Module("v.db.update", map=intsct,
               layer=1, column='cls',
               query_column='b_lulc',
               run_=True, quiet=True
              )


    out_grid = grs_to_shp(intsct, out_shp, 'area')

def run_multiproc(_th, _intrsect):
    for i in _intrsect:
        create_fishnet(_th, i['fish'], i['lc'], i['out'])

thrds = []
```

```
for i in range(len(process_data)):
    processo = mp.Process(
        target=run_multiproc, name=f'th-{str(i+1)}',
        args=(i+1, process_data[i])
    )

    thrds.append(processo)

for p in thrds:
    p.start()

for p in thrds:
    p.join()

print('terminou')
```

Anexo D12 – Programa 12: *Vector grids to raster* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para converter o conjunto de ficheiros vetoriais para *raster*.

Vector grids to raster

In [ ]:
```
import multiprocessing as mp
from glass.pys.oss import lst_ff


import os
from glass.pys.oss import mkdir, fprop
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str

from glass.it.shp import shp_to_grs
from glass.it.rst import grs_to_rst
```

In [ ]:
```
refs = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/update_terceira'

file = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/file_rst/b02_20200220.tif'

out_grid = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_vect_to_rst'

bname = 'terceira_vectorst'
nproc = 4
```

In [ ]:
```
rsts = lst_ff(refs, file_format='.shp')
```

In [ ]:
```
intrsect = []
for r in rsts:
    re = fprop(r, 'fn')

    refsplit = re.split('_')

    idx, idy = refsplit[-2], refsplit[-1]

    intrsect.append({
        'fish' : r,
        'out' : os.path.join(out_grid, f'{bname}_{str(idx)}_{str(idy)}.tif')
    })
```

In [ ]:
```
dimlist = len(intrsect) / nproc

dimlist = int(dimlist) if dimlist == int(dimlist) else int(dimlist) + 1
```

In [ ]:
```
process_data = []
for e in range(0, nproc*dimlist, dimlist):
    process_data.append(intrsect[e:e+dimlist])
```

In [ ]:
```
def create_fishnet(th, raster, out_shp):

    # Check if outfolder exists
    out_folder = os.path.dirname(out_shp)
    if not os.path.exists(out_folder):
        mkdir(out_folder, overwrite=None)

    """
    Start GRASS GIS Session
    """

    loc = f'loc_{now_as_str()}_{str(th)}_{str(idx)}'
    grsb = run_grass(
        out_folder, grassBIN='grass78', location=loc,
        srs=file
    )

    import grass.script.setup as gsetup

    gsetup.init(grsb, out_folder, loc, 'PERMANENT')

    from grass.pygrass.modules import Module

    bloc = shp_to_grs(raster, fprop(raster, 'fn'))

    convert = fprop(out_shp, 'fn')
    m = Module(
        "v.to.rast", input=bloc,
        output=convert,
        use='attr', attribute_column='cls',
        overwrite=True, run_=True, quiet=True
    )
```

```
    out_grid = grs_to_rst(convert, out_shp)

def run_multiproc(_th, _intrsect):
    for i in _intrsect:
        create_fishnet(_th, i['fish'], i['out'])

thrds = []

for i in range(len(process_data)):
    processo = mp.Process(
        target=run_multiproc, name=f'th-{str(i+1)}',
        args=(i+1, process_data[i])
    )

    thrds.append(processo)

for p in thrds:
    p.start()

for p in thrds:
    p.join()

print('terminou')
```

Anexo D13 – Programa 13: *Aggregation of rasters* (código programado no âmbito deste trabalho).

Bloco de código, em *pyhton*, preparado para agregar o conjunto de ficheiros matriciais em apenas um único dado *raster* final.

Aggregation of rasters

In [ ]:
```
import os

from glass.pys.oss import mkdir, fprop, lst_ff
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str
from glass.it.rst import rst_to_grs, grs_to_rst
```

In [ ]:
```
rsts = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/terceira_vect_to_rst'
exte = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/file_rst/b02_20200220.tif'
out = '/home/francisco/create_grid/multiprocess/multiprocess_terceira/patch_rsts/patch_rast.tif'
```

In [ ]:
```
rast = lst_ff(rsts, file_format= '.tif')
```

In [ ]:
```
# Check if outfolder exists
out_folder = os.path.dirname(out)
if not os.path.exists(out_folder):
    mkdir(out_folder, overwrite=None)

"""
Start GRASS GIS Session
"""

loc = f'loc_{now_as_str()}'
grsb = run_grass(
    out_folder, grassBIN='grass78', location=loc,
    srs=exte
)

import grass.script.setup as gsetup

gsetup.init(grsb, out_folder, loc, 'PERMANENT')
```

In [ ]:
```
from grass.pygrass.modules import Module

rasters = []
for r in rast:
    rasters.append(rst_to_grs(r, fprop(r, 'fn')))

out_rast = fprop(out, 'fn')
m = Module(
    "r.patch", input= rasters, output= out_rast,
    overwrite=True, run_=True, quiet=True
)
```

In [ ]:
```
grs_to_rst(out_rast, out)
```

Anexo D14 – Programas 14, 15, 16, 17 e 18: *Example code for calculating the radiometric index* (código existente e adaptado no âmbito deste trabalho)

Exemplo de bloco de código, em *pyhton*, preparado para calcular os diferentes índices radiométricos, para além dos utilizados como referência (NDVI, NDWI e NDBI), os novos índices implementados (MSAVI e NBR).

Example code for calculating the radiometric index

In [1]:
```
import numpy as np
import os
from osgeo import gdal, gdal_array
from osgeo import osr
```

In [2]:
```
red = '/home/francisco/create_grid/mascaras_terceira/bands_stl2/b04_20201025.tif'
nir = '/home/francisco/create_grid/mascaras_terceira/bands_stl2/b08_20201025.tif'
out = '/home/francisco/create_grid/mascaras_terceira/ndvi/ndvi_20201025.tif'
```

In [3]:
```
"""
GDAL Drivers Name
"""

def drv_name(_file):
    """
    Return the driver for a given file format
    """

    drv = {
        # Vector files
        '.gml'    : 'GML',
        '.shp'    : 'ESRI Shapefile',
        '.json'   : 'GeoJSON',
        '.kml'    : 'KML',
        '.osm'    : 'OSM',
        '.dbf'    : 'ESRI Shapefile',
        '.vct'    : 'Idrisi',
        '.nc'     : 'netCDF',
        '.vrt'    : 'VRT',
        '.mem'    : 'MEMORY',
        '.sqlite' : 'SQLite',
        '.gdb'    : 'FileGDB',
        # Raster files
        '.tif'    : 'GTiff',
        '.ecw'    : 'ECW',
        '.mpr'    : 'ILWIS',
        '.mpl'    : 'ILWIS',
```

```
        '.jpg'    : 'JPEG',
        '.nc'     : 'netCDF',
        '.png'    : 'PNG',
        '.vrt'    : 'VRT',
        '.asc'    : 'AAIGrid',
        '.img'    : 'HFA',
        # Vector or Raster
        '.gpkg'   : 'GPKG'
    }

    return str(drv[os.path.splitext(_file)[1]])
```

In [4]:

```
"""
Send Array to Raster
"""

def obj_to_rst(inArray, outRst, template, noData=None, geotrans=None):
    if type(template).__name__ == 'Dataset':
        img_template = template
    else:
        img_template  = gdal.Open(template)

    geo_transform = img_template.GetGeoTransform() if not geotrans else \
        geotrans
    rows, cols    = inArray.shape
    drv_n         = drv_name(outRst)
    driver        = gdal.GetDriverByName(drv_n)

    out       = driver.Create(
        outRst, cols, rows, 1,
        gdal_array.NumericTypeCodeToGDALTypeCode(inArray.dtype)
    )

    out.SetGeoTransform(geo_transform)
    outBand       = out.GetRasterBand(1)

    if noData or noData == 0:
        outBand.SetNoDataValue(noData)

    outBand.WriteArray(inArray)

    proj = osr.SpatialReference(wkt=img_template.GetProjection())
```

```
    if proj:
        out.SetProjection(img_template.GetProjection())

    outBand.FlushCache()

    return outRst
```

In [5]:
```
"""
Apply Normalized Difference NIR/Red Normalized Difference
Vegetation Index, Calibrated NDVI - CDVI

https://www.indexdatabase.de/db/i-single.php?id=58

EXPRESSION: (nir - red) / (nir + red)
"""

def ndvi(nir, red, outRst):

    # Open Images
    src_nir = gdal.Open(nir, gdal.GA_ReadOnly)
    src_red = gdal.Open(red, gdal.GA_ReadOnly)

    # To Array
    num_nir = src_nir.GetRasterBand(1).ReadAsArray().astype(float)
    num_red = src_red.GetRasterBand(1).ReadAsArray().astype(float)

    # Do Calculation
    ndvi = (num_nir - num_red) / (num_nir + num_red)

    # Place NoData Value
    nirNdVal = src_nir.GetRasterBand(1).GetNoDataValue()
    redNdVal = src_red.GetRasterBand(1).GetNoDataValue()

    ndNdvi = np.amin(ndvi) - 1

    np.place(ndvi, num_nir==nirNdVal, ndNdvi)
    np.place(ndvi, num_red==redNdVal, ndNdvi)

    # Export Result
    return obj_to_rst(ndvi, outRst, nir, noData=ndNdvi)
```

In [6]:
```
ndvi(nir, red, out)
```

Anexo D15 – Programa 19: *Final code for data filtering* (código programado no âmbito deste trabalho)

Exemplo de bloco de código, em *pyhton*, preparado para definir regras de limiares de corte para cada classe, índice radiométrico e data de imagem para implementar processos de filtragem de dados através de máscaras de binárias.

Final code for data filtering

In [ ]:

```
lulc_rst = '/home/francisco/create_grid/mascaras/fidx/lulc_ref.tif'

idx_rst = {
    "20220504" : {
        "ndvi" : '/home/francisco/create_grid/ndvi/ndvi_20220504.tif',
        "ndbi" : '/home/francisco/create_grid/ndbi/ndbi_20220504.tif',
        "ndwi" : '/home/francisco/create_grid/ndwi/ndwi_20220504.tif'
    },
    "20220812" : {
        "ndvi" : '/home/francisco/create_grid/ndvi/ndvi_20220812.tif',
        "ndbi" : '/home/francisco/create_grid/ndbi/ndbi_20220812.tif',
        "ndwi" : '/home/francisco/create_grid/ndwi/ndwi_20220812.tif'
    },
    "20221001" : {
        "ndvi" : '/home/francisco/create_grid/ndvi/ndvi_20221001.tif',
        "ndbi" : '/home/francisco/create_grid/ndbi/ndbi_20221001.tif',
        "ndwi" : '/home/francisco/create_grid/ndwi/ndwi_20221001.tif'
    }
}

idx_rules = {
    1 : {
        "ndbi" : {'operator' : '>', 'value': 0, 'oneall' : 'one'},
        "ndvi" : {'operator' : '<', 'value': 0.3, 'oneall' : 'all'},
        "ndwi" : {'operator' : '<', 'value': 0, 'oneall' : 'all'}
    },
    2 : {
        "ndbi" : None,
        "ndvi" : {'operator' : '>', 'value': 0.3, 'oneall' : 'all'},
        "ndwi" : {'operator' : '<', 'value': 0, 'oneall' : 'all'}
    },
    3 : {
        "ndbi" : None,
        "ndvi" : {'operator' : '>', 'value': 0.3, 'oneall' : 'all'},
        "ndwi" : {'operator' : '<', 'value': 0, 'oneall' : 'all'}
    },
```

```
    4 : {
        "ndbi" : None,
        "ndvi" : {'operator' : '>', 'value': 0.3, 'oneall' : 'all'},
        "ndwi" : {'operator' : '<', 'value': 0, 'oneall' : 'all'}
    },
    5 : {
        "ndbi" : None,
        "ndvi" : {'operator' : '>', 'value': 0.3, 'oneall' : 'all'},
        "ndwi" : {'operator' : '<', 'value': 0, 'oneall' : 'all'}
    },
    6 : {
        "ndbi" : None,
        "ndvi" : {'operator' : '>', 'value': 0, 'oneall' : 'one'},
        "ndwi" : {'operator' : '<', 'value': 0, 'oneall' : 'one'}
    },
    7 : {
        "ndbi" : None,
        "ndvi" : {'operator' : '>', 'value': 0, 'oneall' : 'one'},
        "ndwi" : {'operator' : '<', 'value': 0, 'oneall' : 'one'}
    },
    8 : {
        "ndbi" : None,
        "ndvi" : {'operator' : '<', 'value': 0.3, 'oneall' : 'one'},
        "ndwi" : {'operator' : '>', 'value': 0, 'oneall' : 'all'}
    }
}

fraster= '/home/francisco/create_grid/mascaras/fidx/lulc_filter.tif'
```

In [ ]:

```
import os
from osgeo import gdal
import numpy as np
from glass.rd.rst import rst_to_array
from glass.prop.rst import get_nodata
from glass.pys.oss import lst_ff, fprop
from glass.wenv.grs import run_grass
from glass.pys.tm import now_as_str
```

In [ ]:
```
# Check if outfolder exists
out_folder = os.path.dirname(fraster)
if not os.path.exists(out_folder):
    mkdir(out_folder, overwrite=None)

"""
Start GRASS GIS Session
"""

loc = f'loc_{now_as_str()}'
grsb = run_grass(
    out_folder, grassBIN='grass78', location=loc,
    srs=lulc_rst
    )

import grass.script.setup as gsetup

gsetup.init(grsb, out_folder, loc, 'PERMANENT')
```

In [ ]:
```
from glass.it.rst import rst_to_grs, grs_to_rst
from grass.pygrass.modules import Module
```

In [ ]:
```
# Identify existing classes in the raster with LULC classes

lulc_img = rst_to_array(lulc_rst)

ndval = get_nodata(lulc_rst)

lulc_class = [v for v in np.unique(lulc_img) if v != ndval]
```

In [ ]:
```
# Send all rasters to GRASS GIS

lulc_grs = rst_to_grs(lulc_rst, fprop(lulc_rst, 'fn'))

for d in idx_rst:
    for k in idx_rst[d]:
        idx_rst[d][k] = rst_to_grs(idx_rst[d][k], fprop(idx_rst[d][k], 'fn'))
```

In [ ]:
```
mask_by_class_day_idx = {}

for cls in lulc_class:
    mask_by_class_day_idx[cls] = {}
    for d in idx_rst:
        mask_by_class_day_idx[cls][d] = {}
        for k in idx_rst[d]:
            if not idx_rules[cls][k]: continue

            form = (
                f"if({lulc_grs} == {str(cls)} && "
                f"{idx_rst[d][k]} {idx_rules[cls][k]['operator']} "
                f"{idx_rules[cls][k]['value']}, 1, 0)"
            )

            outmask = f'mask_{cls}_{k}_{d}'
            rc = Module ('r.mapcalc', expression=f'{outmask} = {form}')

            mask_by_class_day_idx[cls][d][k] = outmask

            grs_to_rst(outmask, os.path.join(
                os.path.dirname(fraster),
                f'{outmask}.tif'
            ), is_int=True)
```

In [ ]:
```
class_mask= {}
for cls in mask_by_class_day_idx:
    class_mask[cls]= {}
    for day in mask_by_class_day_idx[cls]:
        for idx in mask_by_class_day_idx[cls][day]:
            if idx not in class_mask[cls]:
                class_mask[cls][idx]= []
            class_mask[cls][idx].append(mask_by_class_day_idx[cls][day][idx])
```

In [ ]:
```
for cls in class_mask:
    for idx in class_mask[cls]:
        ifr = []
        for r in class_mask[cls][idx]:
            ifr.append(f"{r} == 1")

        print(ifr)

        if idx_rules[cls][idx]['oneall'] == 'all':
            _if = " && ".join(ifr)
        else:
            _if = " || ".join(ifr)

        outmask = f'clsidxmask_{cls}_{idx}'
        exp = f"if({_if}, 1, 0)"
        print(exp)
        rc = Module ('r.mapcalc', expression=f'{outmask} = {exp}')

        class_mask[cls][idx] = outmask
```

In [ ]:
```
for cls in class_mask:
    ifr = []
    for idx in class_mask[cls]:
        ifr.append(f"{class_mask[cls][idx]} == 1")

    _if = " && ".join(ifr)

    outmask = f'clsmask_{cls}'
    exp = f"if({_if}, 1, 0)"
    print(exp)
    rc = Module ('r.mapcalc', expression=f'{outmask} = {exp}')

    class_mask[cls] = outmask

    grs_to_rst(outmask, os.path.join(
                os.path.dirname(fraster),
                f'{outmask}.tif'
            ), is_int=True)
```

In [ ]:
```
irf = []

for t in class_mask:
    exp = (
        f"if({class_mask[t]} ==1, "
        f"{str(t)}, "
        f"{'null()' if not len (irf) else irf[len(irf)-1]})"
    )

    irf.append(exp)

outmask = f'clsmask'

rc = Module ('r.mapcalc', expression=f'{outmask} = {exp}',
             overwrite=True, run_=True, quiet=True)

class_mask = outmask

grs_to_rst(outmask, os.path.join(
                os.path.dirname(fraster),
                f'{outmask}.tif'
            ), is_int=True)
```

Anexo D16 – Programa 20: *Random forest – Training model file* (código existente).

Bloco de código, em *pyhton*, preparado para realizar o treino do classificador *Random Forest* utilizando o ficheiro *raster* resultante do processo de filtragem dos dados, seguido do seu estudo de bibliotecas e funções implementadas, uma vez que o presente código foi disponibilizado pela orientação.

Random forest - Training model file

In [ ]:
```
import os
from glass.rd      import tbl_to_obj
from glass.rst.cls import train_to_mdl

reffile  = '/home/francisco/create_grid/script5_terceira/excel/excel_scrpt5.xlsx'
refsheet = 'mainsheet'


mdf = tbl_to_obj(reffile, sheet=refsheet)

for i, row in mdf.iterrows():
    cvar = tbl_to_obj(reffile, sheet=row.sheet)

    train_to_mdl(
        row.traindata,
        [os.path.join(row.train_folder, f) for f in cvar.trainvar.tolist()],
        row.trainfile,
        ntrees=row.ntrees
    )
```

Análise e estudo do Programa 20: *Random forest – Training model file*

```
import os
from glass.rd      import tbl_to_obj
from glass.rst.cls import train_to_mdl

reffile  = '/home/francisco/lisboa/excel/excel_scrpt5.xlsx'
refsheet = 'mainsheet'


mdf = tbl_to_obj(reffile, sheet=refsheet)

for i, row in mdf.iterrows():
    cvar = tbl_to_obj(reffile, sheet=row.sheet)

    train_to_mdl(
        row.traindata,
        [os.path.join(row.train_folder, f) for f in cvar.trainvar.tolist()],
        row.trainfile,
        ntrees=row.ntrees
    )
```

Ficheiro excel introduzido pelo utilizador que incorpora todas as informações necessárias para o treino do modelo:

- Ficheiro dos dados de treino filtrados;
- Bandas das imagens de satélite e MDE;
- Número de árvores do classificador.

Campo do ficheiro excel a ser identificado com todos os parâmetros

Programa 20: Extremamente importante para a criar o ficheiro modelo treinado com a informação das diferentes classes através da aplicação do classificador *random forest*. Para a correta execução do treino, este código necessita do ficheiro excel que integre o ficheiro dos dados de treino filtrados, as diferentes bandas das imagens e o MDE e a definição do número de árvores do classificador.

Imports:

```
import os
from glass.rd      import tbl_to_obj
from glass.rst.cls import train_to_mdl
```

- *import* OS: o módulo OS em *Python* fornece funções para interagir com o sistema operacional. O sistema operacional vem sob os módulos padrão do *Python*. Este módulo fornece uma maneira flexível de usar a funcionalidade dependendo do sistema operacional. Os módulos "OS" e "os.path" incluem muitas funções para interagir com o sistema de pastas.

- from glass.rd import tbl_to_obj (…): são diferentes *scripts* pré-criados e importados de outras pastas, introduzidos no Programa 20 através do caminho de pastas. Todos os *imports* seguintes serão explicados nas secções seguintes.

Módulos e ferramentas:

```
for i, row in mdf.iterrows():
    cvar = tbl_to_obj(reffile, sheet=row.sheet)

    train_to_mdl(
        row.traindata,
        [os.path.join(row.train_folder, f) for f in cvar.trainvar.tolist()],
        row.trainfile,
        ntrees=row.ntrees
    )
```

- *itrrows*: método que gera um objeto iterado do *DataFrame*, que permite iterar cada linha do *DataFrame*. Cada iteração produz um objeto de índice e um objeto de linha.

- *sheet*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel utilizada para identificar o nome das bandas das imagens de satélite e MDE selecionadas para o treino do classificador).

- *traindata*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel que indica o caminho para o ficheiro dos dados de treino filtrados).

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipulação comum de nomes de caminho. O método *os.path.join()* em *python* une um ou mais componentes de caminho de forma inteligente. Este método concatena vários componentes de caminho com exatamente um separador de diretório ('/') após cada parte não vazia, exceto o último componente do caminho. Exemplo: '/home/glass/francisco/'.

- *train_folder*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel que conduz à pasta dos ficheiros relativos às bandas das imagens de satélite e MDE).

- *trainvar*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel que identifica as bandas das imagens de satélite e MDE para o treino do classificador).

- *tolist*: método utilizado para converter elementos de dados para lista respeitando as mesmas informações, elementos ou valores.

- *ntrees*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel que identifica o número de árvores definidas para treino do classificador).

Principais *Imports* implementados no Programa 20 e descrição dos mesmos

Nas figuras seguintes são apresentados os diferentes *Imports* materializados no Programa 20 com funções e objetivos extremamente importantes para o desempenho do código. Como foi mencionado anteriormente, a maioria dos *imports* são *scripst*/códigos, desenvolvidos pela orientação, encontrados no repositório GLASS. É preciso notar que um grande número de funções implementadas no Programa 20, foram explicadas e definidas anteriormente nos Programa 1 e 2 e, portanto, nas secções que se seguem serão descritas apenas funções que ainda não detiveram de um esclarecimento aprofundado e oportuno.

```
import os
from glass.rd      import tbl_to_obj
from glass.rst.cls import train_to_mdl
```

(Já explicados: *Import OS* incluído no Programa 1)

Import tbl_to_objt – *from* glass.rd *import* tbl_to_objt

```
import os
from glass.rd      import tbl_to_obj
from glass.rst.cls import train_to_mdl
```

"Ficheiro de tabela para Pandas *DataFrame*"

"Opções de outputs: - df; - dict; - array;"

```
def tbl_to_obj(tblFile, sheet=None, useFirstColAsIndex=None,
               _delimiter=None, encoding_='utf8', output='df',
               fields=None, colsAsArray=None, csvheader=True):
    """
    Table File to Pandas DataFrame

    output Options:
    - df;
    - dict;
    - array;
    """

    import pandas as pd
    from glass.pys.oss import fprop

    fFormat = fprop(tblFile, 'ff')

    if fFormat == '.dbf':
        """
        Convert dBase to Pandas Dataframe
        """

        from simpledbf import Dbf5

        dbfObj = Dbf5(tblFile)

        tableDf = dbfObj.to_dataframe()

    elif fFormat == '.ods':
        """
        ODS file to Pandas Dataframe
        """
```

(Continuação do código tbl_to_objt)

```
        from pyexcel_ods import get_data

        if not sheet:
            raise ValueError("You must specify sheet name when converting ods files")
        data = get_data(tblFile)[sheet]

        tableDf = pandas.DataFrame(data[1:], columns=data[0])

    elif fFormat == '.xls' or fFormat == '.xlsx':
        """
        XLS to Pandas Dataframe
        """

        from glass.pys import obj_to_lst

        sheet = 0 if not sheet else sheet

        indexCol = 0 if useFirstColAsIndex else None

        tableDf = pd.read_excel(
            tblFile, sheet, index_col=indexCol,
            dtype='object',
            usecols=obj_to_lst(fields) if fields != "ALL" else None
        )

    elif fFormat == '.txt' or fFormat == '.csv':
        """
        Text file to Pandas Dataframe
        """

        if not _delimiter:
            raise ValueError(
                "You must specify _delimiter when converting txt files"
            )
        if csvheader:
            tableDf = pd.read_csv(
                tblFile, sep=_delimiter, low_memory=False,
                encoding=encoding_
            )
        else:
            tableDf = pd.read_csv(
                tblFile, sep=_delimiter, low_memory=False,
                encoding=encoding_, header=None
            )

    else:
        raise ValueError(f'{fFormat} is not a valid table format!')

    if fields:
        from glass.pys  import obj_to_lst

        fields = obj_to_lst(fields)
        if fields:
            delCols = []
            for fld in list(tableDf.columns.values):
                if fld not in fields:
                    delCols.append(fld)

            if delCols:
                tableDf.drop(delCols, axis=1, inplace=True)

    if output != 'df':
        if output == 'dict':
            orientation = "index" if not colsAsArray else "list"

        elif output == 'array':
            tableDf["FID"] = tableDf.index

            orientation = "records"

        tableDf = tableDf.to_dict(orient=orientation)

    return tableDf
```

Métodos e Ferramentas:

#Converte dBase para Pandas *DataFrame*

```
import pandas as pd
from glass.pys.oss import fprop

fFormat = fprop(tblFile, 'ff')

if fFormat == '.dbf':
    """
    Convert dBase to Pandas Dataframe
    """

    from simpledbf import Dbf5

    dbfObj = Dbf5(tblFile)

    tableDf = dbfObj.to_dataframe()
```

- *import pandas*: biblioteca utilizada para manipular e analisar dados, através da linguagem de programação em *python*. Detêm de um conjunto de módulos excelentes para iniciar a análise explorativa de dados, porque permite ler, alterar, agregar dados em poucos passos. É utilizado neste programa porque é a biblioteca mais indicada para trabalhar com tabelas.

- *import simpledbf*: pequeno módulo em *python* utilizado para converter dados de extensão *.dbf* para ficheiros CSV, *Pandas DataFrames*, tabelas SQL ou tabelas HDF5.

- *import Dbf5*: é o objeto que inicialmente lê apenas as informações de introdução do ficheiro, para que seja possível adquirir algumas informações das propriedades dos dados.

- *to_dataframe*: converte conjuntos de dados definidos para *Pandas DataFrame*.

#Converte ODS file para *Pandas DataFrame*

```
elif fFormat == '.ods':
    """
    ODS file to Pandas Dataframe
    """

    from pyexcel_ods import get_data

    if not sheet:
        raise ValueError("You must specify sheet name when converting ods files")
    data = get_data(tblFile)[sheet]

    tableDf = pandas.DataFrame(data[1:], columns=data[0])
```

- *import pyexcel_ods*: pequena biblioteca abstrata utilizada para ler, manipular e gravar dados no formato *.ods* utilizando a linguagem *python*.

- *import get_data*: é o objeto que permite procurar qualquer ficheiro ou dado e introduzir no código.

- *DataFrame*: dados representados em tabelas bidimensionais, de diferentes tamanhos e potencialmente heterogéneos. A estrutura de dados contém eixos descritos em linhas e

colunas. As operações aritméticas alinham-se nos nomes de linha e coluna. É considerada como a estrutura principal de dados do *Pandas*.

#Converte XLS para *Pandas DataFrame*

```
    elif fFormat == '.xls' or fFormat == '.xlsx':
        """
        XLS to Pandas Dataframe
        """

        from glass.pys import obj_to_lst

        sheet = 0 if not sheet else sheet

        indexCol = 0 if useFirstColAsIndex else None

        tableDf = pd.read_excel(
            tblFile, sheet, index_col=indexCol,
            dtype='object',
            usecols=obj_to_lst(fields) if fields != "ALL" else None
        )
```

- *read_excel*: tem o objetivo de ler um ficheiro excel e converter a informação para um *Pandas DataFrame*. Suporta extensões como *.xls*, *.xlsx*, *.xlsb*, *.odf*, *.ods* e formatos de ficheiros *.odt* lidas de um sistema local ou URL. Tem a opção de ler uma única folha ou uma lista de folhas.

#Ficheiro texto para *Pandas DataFrame*

```
    elif fFormat == '.txt' or fFormat == '.csv':
        """
        Text file to Pandas Dataframe
        """

        if not _delimiter:
            raise ValueError(
                "You must specify _delimiter when converting txt files"
            )

        if csvheader:
            tableDf = pd.read_csv(
                tblFile, sep=_delimiter, low_memory=False,
                encoding=encoding_
            )
        else:
            tableDf = pd.read_csv(
                tblFile, sep=_delimiter, low_memory=False,
                encoding=encoding_, header=None
            )
```

- *read_csv*: permite a leitura de ficheiros separados por vírgula (*.csv*). Também suporta a iteração ou separação de ficheiros em várias partes.

```
    else:
        raise ValueError(f'{fFormat} is not a valid table format!')

    if fields:
        from glass.pys  import obj_to_lst

        fields = obj_to_lst(fields)
        if fields:
            delCols = []
            for fld in list(tableDf.columns.values):
                if fld not in fields:
                    delCols.append(fld)

            if delCols:
                tableDf.drop(delCols, axis=1, inplace=True)
```

- *columns.values*: devolve uma representação *Numpy* do *DataFrame* (values) e ainda recupera os nomes das colunas (*columns*)

- *append*: método que acrescenta um elemento a uma lista final.

- *drop*: exclui nomes de linhas e colunas específicos. Elimina linhas ou colunas através da designação dos nomes e eixo correspondente ou diretamente, através do nome da linha ou coluna.

```
    if output != 'df':
        if output == 'dict':
            orientation = "index" if not colsAsArray else "list"

        elif output == 'array':
            tableDf["FID"] = tableDf.index

            orientation = "records"

        tableDf = tableDf.to_dict(orient=orientation)

    return tableDf
```

- index: sequência imutável utilizada para indexação e alinhamento. Objeto básico que guarda os nomes de eixos em todos os objetos *Pandas*.

- *to_dict*: Converte o *DataFrame* num dicionário.

Descrição:

Código muito extenso que utiliza a biblioteca *Pandas* e bibliotecas mais pequenas, como *simpledbf* e *pyexcel_ods* para executar a conversão de diferentes extensões para um *Pandas DataFrame*. O principal objetivo do código é converter tabelas excel dadas como entrada para diferentes saídas que posteriormente, a sua informação possa ser trabalhada e utilizada através da linguagem de programação *python*. Nesta situação é utilizado para converter um ficheiro excel para um *Pandas DataFrame* para que a sua informação possa ser utilizada ao longo do Programa 20.

Imports dentro do código tbl_to_objt:

```
17      import pandas as pd
18      from glass.pys.oss import fprop
```

```
46      elif fFormat == '.xls' or fFormat == '.xlsx':
47          """
48          XLS to Pandas Dataframe
49          """
50
51          from glass.pys import obj_to_lst
52
53          sheet = 0 if not sheet else sheet
```

```
84      else:
85          raise ValueError(f'{fFormat} is not a valid table format!')
86
87      if fields:
88          from glass.pys  import obj_to_lst
```

Import fprop dentro do código tbl_to_objt

```
17      import pandas as pd
18      from glass.pys.oss import fprop
```

```
25  def fprop(__file, prop, forceLower=None, fs_unit=None):
26      """
27      Return some property of file
28
29      prop options:
30      * filename or fn - return filename
31      """
32
33      from glass.pys import obj_to_lst
34
35      prop = obj_to_lst(prop)
36
37      result = {}
38
39      if 'filename' in prop or 'fn' in prop:
40          fn, ff = os.path.splitext(os.path.basename(__file))
41
42          result['filename'] = fn if not forceLower else fn.lower()
43
44          if 'fileformat' in prop or 'fn' in prop:
45              result['fileformat'] = ff
46
47      elif 'fileformat' in prop or 'ff' in prop:
48          result['fileformat'] = os.path.splitext(__file)[1]
49
50      if 'filesize' in prop or 'fs' in prop:
51          fs_unit = 'MB' if not fs_unit else fs_unit
52
53          fs = os.path.getsize(__file)
54
55          if fs_unit == 'MB':
56              fs  = (fs / 1024.0) / 1024
57
58          elif fs_unit == 'KB':
59              fs = fs / 1024.0
60
```

(Já explicados: *Import* fprop incluído no Programa 1)

Import obj_to_lst dentro do código tbl_to_objt

```
46      elif fFormat == '.xls' or fFormat == '.xlsx':
47          """
48          XLS to Pandas Dataframe
49          """
50
51          from glass.pys import obj_to_lst
52
53          sheet = 0 if not sheet else sheet
84      else:
85          raise ValueError(f'{fFormat} is not a valid table format!')
86
87      if fields:
88          from glass.pys  import obj_to_lst
```

```
17  def obj_to_lst(obj):
18      """
19      A method uses a list but the user gives other type of object
20
21      This method will see if the object is not a list and convert it to a list
22      """
23
24      return obj if type(obj) == list else [obj] if obj != None else None
```

(Já explicados: *Import* obj_to_lst incluído no Programa 1)

Import train_to_mdl – *from* glass.rst.cls *import* train_to_mdl

```
import os
from glass.rd      import tbl_to_obj
from glass.rst.cls import train_to_mdl
```

"Treina o modelo para classificação e guarda esse modelo num ficheiro"

"Classificadores disponíveis: *Random Forest*"

```python
def train_to_mdl(train_rst, imgs, outmdl, ntrees=1000, fileformat='.tif'):
    """
    Train a model for classification and save the model in a file

    Classifiers available:
    * Random Forest;
    """

    import joblib
    import numpy as np
    import os
    from osgeo import gdal, gdal_array
    from sklearn.ensemble import RandomForestClassifier

    # Open Data
    img_ref = gdal.Open(train_rst, gdal.GA_ReadOnly)

    if type(imgs) != list:
        # Check if it is a folder
        if os.path.isdir(imgs):
            # List images in folder
            from glass.pys.oss import lst_ff

            imgs = lst_ff(
                imgs,
                file_format=fileformat if fileformat else '.tif'
            )

        else:
            imgs = [imgs]

    img_var = [gdal.Open(i, gdal.GA_ReadOnly) for i in imgs]

    # Get band number for each raster
    img_bnd = [i.RasterCount for i in img_var]
    # Check images shape
    # Return error if the shapes are different
    ref_shp = (img_ref.RasterYSize, img_ref.RasterXSize)
    for r in img_var:
        rst_shp = (r.RasterYSize, r.RasterXSize)

        if ref_shp != rst_shp:
            raise ValueError(
                'There are at least two raster files with different shape'
            )

    # Get NoData Value
    nd_val = img_ref.GetRasterBand(1).GetNoDataValue()

    # Get Number of features
    nvar = sum(img_bnd)

    # Convert imgs to Array, remove nodata values and reshape
    num_ref = img_ref.GetRasterBand(1).ReadAsArray()
    num_ref = num_ref.reshape((-1, 1))

    # Get Y - train data without nodata value
    Y = num_ref[num_ref != nd_val]

    # Get X - var data to array, delete nodata and reshape
    X = np.zeros((
        Y.shape[0], nvar),
        gdal_array.GDALTypeCodeToNumericTypeCode(
            img_var[0].GetRasterBand(1).DataType
        )
    )
    f = 0
    for r in range(len(img_var)):
        for b in range(img_bnd[r]):
            a = img_var[r].GetRasterBand(b + 1).ReadAsArray()
            a = a.reshape((-1, 1))
            a = a[num_ref != nd_val]

            X[:, f] = a

            f += 1

    # Fit model
    m = RandomForestClassifier(
        n_estimators=ntrees, random_state=0, n_jobs=-1
    )

    m.fit(X, Y)

    # Save model
    joblib.dump(m, outmdl)

    return outmdl
```

Métodos e Ferramentas:

#Abrir dados

```
110     import joblib
111     import numpy as np
112     import os
113     from osgeo import gdal, gdal_array
114     from sklearn.ensemble import RandomForestClassifier
115
116     # Open Data
117     img_ref = gdal.Open(train_rst, gdal.GA_ReadOnly)
```

- *import Joblib*: conjunto de ferramentas para fornecer *pipelining* leve através da linguagem de programação *python*. Particularmente é um *disk cache* transparente de funções e reavaliação ociosa que auxilia em processos paralelos simples e fáceis. *Joblib* é concebido para ser rápido e robusto para dados pesados e possui otimizações especificas para *arrays Numpy*.

- *import Numpy*: *NumPy*, significa *Numerical Python*, e é conhecida como uma biblioteca de computação científica construída na linguagem de programação *python*. *Array* é o tipo de dados mais comum de se trabalhar a partir desta biblioteca.

- *from osgeo import gdal:* GDAL é uma biblioteca C++ para mais de 200 formatos de dados geospaciais *raster* e vetoriais. É lançado sob uma licença *open source pela Open Source Geospacial Foudation*.

- *import sklearn.ensemble*: é uma biblioteca baseada em algoritmos de *machine learning* para classificação, regressão e deteção de anomalias.

- *import RandomForestClassifier*: classificador *random forest*.

- *gdal.Open*: função utilizada para abrir um *dataset*.

- *GA_ReadOnly*: função utilizada em conjunto com *gdal.Open(),* com o objetivo de fazer a leitura da informação de um determinado ficheiro.

#Verfica se existe uma pasta

#Lista as imagens na pasta

```
119     if type(imgs) != list:
120         # Check if it is a folder
121         if os.path.isdir(imgs):
122             # List images in folder
123             from glass.pys.oss import lst_ff
```

- *os.path.isdir*: método em *python* utilizado para verificar se o caminho especificado para uma pasta existe ou não. Este método segue um *link* representativo, ou seja, se o caminho indicado for um *link* para uma pasta que exista, o método devolve *true*.

#Aceder ao número de bandas em cada *raster*

#Confirmar se são *shape*

#Devolver erro se forem diferentes *shapes*

```
135     # Get band number for each raster
136     img_bnd = [i.RasterCount for i in img_var]
137
138     # Check images shape
139     # Return error if the shapes are different
140     ref_shp = (img_ref.RasterYSize, img_ref.RasterXSize)
141     for r in img_var:
142         rst_shp = (r.RasterYSize, r.RasterXSize)
```

- *RasterCount*: utilizado para obter o número de bandas *raster* que compõem uma imagem de satélite.

- *RasterYSize*: utilizado para obter a altura da imagem *raster*.

- *RasterXSize*: utilizado para obter a largura da imagem *raster*.

#Obter valor *NoData*

#Obter o número de *features*

#Converter imagens para *array,* remover os valores *NoData* e *reshape*

```
149     # Get NoData Value
150     nd_val = img_ref.GetRasterBand(1).GetNoDataValue()
151
152     # Get Number of features
153     nvar = sum(img_bnd)
154
155     # Convert imgs to Array, remove nodata values and reshape
156     num_ref = img_ref.GetRasterBand(1).ReadAsArray()
157     num_ref = num_ref.reshape((-1, 1))
```

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *GetNoDataValue*: vai buscar o valor sem dados para esta banda. Se não houver nenhum valor de dados fora do intervalo, um valor fora do intervalo é devolvido. O valor sem dados para uma banda, geralmente é um valor especial usado para destacar pixéis que não são dados válidos. Estes pixéis, normalmente não devem ser exibidos nem usados para contribuir em

operações de análise. O valor sem dados é devolvido como "bruto", o que significa que não há deslocamentos e escalas aplicadas.

- *ReadsArray*: utilizado em *python* para ler *arrays*.

- *reshape*: altera a nova forma do *array* sem alterar os seus dados.

#Obter X – *var data* para *array*, eliminar *NoData* e *reshape*

```
    # Get X - var data to array, delete nodata and reshape
    X = np.zeros((
        Y.shape[0], nvar),
        gdal_array.GDALTypeCodeToNumericTypeCode(
            img_var[0].GetRasterBand(1).DataType
        )
```

- *zeros*: devolve um novo *array* de formato e tipo preenchido com zeros.

- *shape*: os *arrays Numpy* têm um atributo chamado *shape* que geralmente é usado para obter a forma de um *array*, mas também devolve uma tupla que pode ser usada para alterar as dimensões do *array*.

- *GDALTypeCodeToNumbericTypeCode*: traduz um valor do tipo "numérico" num tipo de código GDAL.

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *DataType*: tipo de formato que se atribui a um ficheiro ou output.

#Modelo adequado

#Guardar modelo

```
    # Fit model
    m = RandomForestClassifier(
        n_estimators=ntrees, random_state=0, n_jobs=-1
    )

    m.fit(X, Y)

    # Save model
    joblib.dump(m, outmdl)
```

- *fit*: serve para alinhar um algoritmo de *machine learning* em *python*.

- *dump*: guarda um objeto *python* persistente num ficheiro.

Descrição:

Código muito extenso e complexo que realiza várias tarefas, mas no fundo o seu principal objeto é pegar na informação referente aos dados de treino e treinar um modelo de classificação e guardá-lo num ficheiro. Neste caso concreto o classificador treinado é o *random forest*, o que vai produzir ficheiros modelos treinados.

Imports dentro do código train_to_mdl:

```
119     if type(imgs) != list:
120         # Check if it is a folder
121         if os.path.isdir(imgs):
122             # List images in folder
123             from glass.pys.oss import lst_ff
```

Import lst_ff dentro do código train_to_mdl

```
    if type(imgs) != list:
        # Check if it is a folder
        if os.path.isdir(imgs):
            # List images in folder
            from glass.pys.oss import lst_ff
```

```
def lst_ff(w, file_format=None, filename=None, fnpart=None, rfilename=N
    """
    List the abs path of all files with a specific extension on a folde
    """

    from glass.pys import obj_to_lst

    filename = None if filename and fnpart else filename

    # Prepare file format list
    if file_format:
        formats = obj_to_lst(file_format)

        for f in range(len(formats)):
            if formats[f][0] != '.':
                formats[f] = '.' + formats[f]

    # List files
    r = []
    for (d, _d_, f) in os.walk(w):
        r.extend(f)
        break

    # Filter files by format or not
    if not file_format:
        if not rfilename:
            t = [os.path.join(w, i) for i in r]
        else:
            t = [i for i in r]

    else:
        if not rfilename:
            t = [
                os.path.join(w, i) for i in r
                if os.path.splitext(i)[1] in formats
            ]
        else:
            t = [i for i in r if os.path.splitext(i)[1] in formats]

    # Filter by filename
    if not filename and not fnpart: return t

    elif filename and not fnpart:
        filename = obj_to_lst(filename)

        _t = []
        for i in t:
            fn = fprop(i, 'fn') if not rfilename else i
            if fn in filename:
                _t.append(i)

        return _t

    elif not filename and fnpart:
        fnp = obj_to_lst(fnpart)

        _t = []
        for i in t:
            fn = fprop(i, 'fn') if not rfilename else i

            for _f in fnp:
                if _f in fn:
                    _t.append(i)
                    break

        return _t
```

(Já explicados: *Import* lst_ff incluído no Programa 1)

Imports dentro do def lst_ff:

```
81      from glass.pys import obj_to_lst
```

Import obj_to_lst dentro do código def lst_ff

```
17  def obj_to_lst(obj):
18      """
19      A method uses a list but the user gives other type of object
20
21      This method will see if the object is not a list and convert it to a list
22      """
23
24      return obj if type(obj) == list else [obj] if obj != None else None
```

(Já explicados: *Import* obj_to_lst incluído no Programa 1)

Anexo D17 – Programa 21: *Execute classification* (código existente).

Bloco de código, em *pyhton*, preparado para realizar a classificação de imagens, recorrendo ao algoritmo de classificação *Random Forest*, seguido do seu estudo de bibliotecas e funções implementadas, uma vez que o presente código foi disponibilizado pela orientação.

Execute classification

In [1]:
```
import os
from glass.rd import tbl_to_obj
from glass.rst.cls import imgcls_from_mdl

reffile  = '/home/francisco/create_grid/script5_terceira/excel/excel_scrpt5.xlsx'
refsheet = 'mainsheet'


mdf = tbl_to_obj(reffile, sheet=refsheet)


for i, row in mdf.iterrows():
    cvar = tbl_to_obj(reffile, sheet=row.sheet)

    imgcls_from_mdl(
        row.trainfile,
        [os.path.join(row.class_folder, f) for f in cvar.classvar.tolist()],
        row.result
    )
```

Análise e estudo do Programa 21: *Execute classification*

Execute classification

[1]:
```
import os
from glass.rd import tbl_to_obj
from glass.rst.cls import imgcls_from_mdl

reffile  = '/home/francisco/lisboa/excel/excel_scrpt5.xlsx'
refsheet = 'mainsheet'


mdf = tbl_to_obj(reffile, sheet=refsheet)


for i, row in mdf.iterrows():
    cvar = tbl_to_obj(reffile, sheet=row.sheet)

    imgcls_from_mdl(
        row.trainfile,
        [os.path.join(row.class_folder, f) for f in cvar.classvar.tolist()],
        row.result
    )
```

Ficheiro excel introduzido pelo utilizador que inclui todos os parâmetros necessários à execução da classificação:

- Ficheiro dos dados de treino filtrados;
- Bandas das imagens de satélite e MDE;
- Número de árvores do classificador.

Campo do ficheiro excel a ser identificado com todos os parâmetros

Programa 21: De extrema importância para executar a classificação das diferentes imagens de satélite.

Imports:

```
[1]: import os
     from glass.rd import tbl_to_obj
     from glass.rst.cls import imgcls_from_mdl
```

- *import* OS: o módulo OS em *Python* fornece funções para interagir com o sistema operacional. O sistema operacional vem sob os módulos padrão do *Python*. Este módulo fornece uma maneira flexível de usar a funcionalidade dependendo do sistema operacional. Os módulos "OS" e "os.path" incluem muitas funções para interagir com o sistema de pastas.

- *from* glass.rd *import* tbl_to_obj (…): são diferentes *scripts* pré-criados e importados de outras pastas, introduzidos no Programa 21 através do caminho de pastas. Todos os *imports* seguintes serão explicados nas secções seguintes.

Módulos e ferramentas:

```
for i, row in mdf.iterrows():
    cvar = tbl_to_obj(reffile, sheet=row.sheet)

    imgcls_from_mdl(
        row.trainfile,
        [os.path.join(row.class_folder, f) for f in cvar.classvar.tolist()],
        row.result
    )
```

- *itrrows*: método que gera um objeto iterado do *DataFrame*, que permite iterar cada linha do *DataFrame*. Cada iteração produz um objeto de índice e um objeto de linha.

- *sheet*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel utilizada para identificar o nome das bandas das imagens de satélite e MDE selecionadas para o treino do classificador).

- *trainfile*: linha da folha de excel para leitura em *python*. (nome dado à coluna presente no excel utilizada para identificar o nome os ficheiros modelo treinados)

- *os.path.join*: o módulo *os.path* é um submódulo do módulo OS em *python* usado para manipulação comum de nomes de caminho. O método *os.path.join()* em *python* une um ou mais componentes de caminho de forma inteligente. Este método concatena vários

componentes de caminho com exatamente um separador de diretório ('/') após cada parte não vazia, exceto o último componente do caminho. Exemplo: '/home/glass/francisco/'.

- *class_folder*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel que conduz à pasta dos ficheiros relativos às bandas das imagens de satélite e MDE).

- *classvar*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel que identifica as bandas das imagens de satélite e MDE para o treino do classificador).

- *tolist*: método utilizado para converter elementos de dados para lista respeitando as mesmas informações, elementos ou valores.

- *result*: linha da folha de excel para leitura em *python* (nome dado à coluna presente no excel que indica o caminho e nome dos ficheiros das imagens classificadas).

Principais *Imports* implementados no Programa 21 e descrição dos mesmos

Nas figuras seguintes são apresentados os diferentes *Imports* materializados no Programa 21 com funções e objetivos extremamente relevantes para o desempenho do código. Como foi mencionado anteriormente, a maioria dos *imports* são *scripst*/funções, desenvolvidos pela orientação, encontrados no repositório GLASS. É preciso notar que um grande número de funções implementadas no Programa 21, foram explicadas e definidas anteriormente nos outros programas e, portanto, nas secções que se seguem serão descritas apenas funções que ainda não detiveram de um esclarecimento aprofundado e oportuno.

```
[1]: import os
     from glass.rd import tbl_to_obj
     from glass.rst.cls import imgcls_from_mdl
```

(Já explicados: *Import* OS e tbl_to_obj incluídos no Programa 20)

Import imgcls_from_mdl – *from* glass.rst.cls *import* imgcls_from_mdl

[1]:
```
import os
from glass.rd import tbl_to_obj
from glass.rst.cls import imgcls_from_mdl
```

"Classificação a partir do ficheiro Modelo"

```
def imgcls_from_mdl(mdl, imgvar, outrst, fileformat='.tif'):
    """
    Classification from Model File
    """

    import os
    from joblib import load
    from osgeo import gdal, gdal_array
    import numpy as np
    from glass.wt.rst import obj_to_rst

    if type(imgvar) != list:
        # Check if it is a folder
        if os.path.isdir(imgvar):
            from glass.pys.oss import lst_ff

            imgvar = lst_ff(
                imgvar, file_format=fileformat if fileformat else '.tif'
            )

        else:
            imgvar = [imgvar]

    # Read model file
    rf = load(mdl)

    # Open feature images
    img_var = [gdal.Open(i, gdal.GA_ReadOnly) for i in imgvar]

    # Get NoData Value
    nd_val = img_var[0].GetRasterBand(1).GetNoDataValue()

    # Get band number of each raster
    img_bnd = [i.RasterCount for i in img_var]
    # Check images shape
    ref_shp = (img_var[0].RasterYSize, img_var[0].RasterXSize)
    if len(img_var) > 1:
        for r in range(1, len(img_var)):
            rst_shp = (img_var[r].RasterYSize, img_var[r].RasterXSize)

            if ref_shp != rst_shp:
                raise ValueError(
                    'There are at least two raster files with different shape'
                )

    # Get features number
    nvar = sum(img_bnd)

    # Convert feature images to array
    X = np.zeros(
        (ref_shp[0], ref_shp[1], nvar),
        gdal_array.GDALTypeCodeToNumericTypeCode(
            img_var[0].GetRasterBand(1).DataType
        )
    )

    f = 0
    for r in range(len(img_var)):
        for b in range(img_bnd[r]):
            X[:, :, f] = img_var[r].GetRasterBand(b + 1).ReadAsArray()

            f += 1

    # Reshape
    nshp = (X.shape[0] * X.shape[1], X.shape[2])
    n_x = X[:, :, :nvar].reshape(nshp)

    # Predict
    y_cls = rf.predict(n_x)

    # Reshape result
    res = y_cls.reshape(X[:, :, 0].shape)

    # Place nodata values
    tmp = img_var[0].GetRasterBand(1).ReadAsArray()
    np.place(res, tmp==nd_val, 255)

    # Export result
    obj_to_rst(res, outrst, imgvar[0], noData=255)

    return outrst
```

Métodos e Ferramentas:

#Verifica se é uma pasta

```
305     import os
306     from joblib import load
307     from osgeo import gdal, gdal_array
308     import numpy as np
309     from glass.wt.rst import obj_to_rst
310
311     if type(imgvar) != list:
312         # Check if it is a folder
313         if os.path.isdir(imgvar):
314             from glass.pys.oss import lst_ff
```

- *import Joblib*: conjunto de ferramentas para fornecer *pipelining* leve através da linguagem de programação *python*. Particularmente é um *disk cache* transparente de funções e reavaliação ociosa que auxilia em processos paralelos simples e fáceis. *Joblib* é concebido para ser rápido e robusto para dados pesados e possui otimizações especificas para *arrays Numpy*.

- *import load*: reconstrói um objeto *python* a partir de um ficheiro criado com o *joblib.dump*.

- *import Numpy*: *NumPy*, significa *Numerical Python*, e é conhecida como uma biblioteca de computação científica construída na linguagem de programação *python*. *Array* é o tipo de dados mais comum de se trabalhar a partir desta biblioteca.

- *from osgeo import gdal:* GDAL é uma biblioteca C++ para mais de 200 formatos de dados geospaciais *raster* e vetoriais. É lançado sob uma licença *open source pela Open Source Geospacial Foudation*.

- *os.path.isdir*: método em *python* utilizado para verificar se o caminho especificado para uma pasta existe ou não. Este método segue um *link* representativo, ou seja, se o caminho indicado for um *link* para uma pasta que exista, o método devolve *true*.

#Leitura do ficheiro modelo

#Abrir as imagens

#Obter valor *NoData*

#Obter o número da banda de cada *raster*

```
323     # Read model file
324     rf = load(mdl)
325
326     # Open feature images
327     img_var = [gdal.Open(i, gdal.GA_ReadOnly) for i in imgvar]
328
329     # Get NoData Value
330     nd_val = img_var[0].GetRasterBand(1).GetNoDataValue()
331
332     # Get band number of each raster
333     img_bnd = [i.RasterCount for i in img_var]
```

- *gdal.Open*: função utilizada para abrir um *dataset*.

- *GA_ReadOnly*: função utilizada em conjunto com *gdal.Open(),* com o objetivo de fazer a leitura da informação de um determinado ficheiro.

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *GetNoDataValue*: vai buscar o valor sem dados para esta banda. Se não houver nenhum valor de dados fora do intervalo, um valor fora do intervalo é devolvido. O valor sem dados para uma banda, geralmente é um valor especial usado para destacar pixéis que não são dados válidos. Estes pixéis, normalmente não devem ser exibidos nem usados para contribuir em operações de análise. O valor sem dados é devolvido como "bruto", o que significa que não há deslocamentos e escalas aplicadas.

- *RasterCount*: utilizado para obter o número de bandas *raster* que compõem uma imagem de satélite.

#Confirmar as imagens *shape*

```
335     # Check images shape
336     ref_shp = (img_var[0].RasterYSize, img_var[0].RasterXSize)
337     if len(img_var) > 1:
338         for r in range(1, len(img_var)):
339             rst_shp = (img_var[r].RasterYSize, img_var[r].RasterXSize)
340 
341             if ref_shp != rst_shp:
342                 raise ValueError(
343                     'There are at least two raster files with different shape'
344                 )
```

- *RasterYSize*: utilizado para obter a altura da imagem *raster*.

- *RasterXSize*: utilizado para obter a largura da imagem *raster*.

#Obter o número de *features*

#Converter as imagens para *array*

```
346     # Get features number
347     nvar = sum(img_bnd)
348 
349     # Convert feature images to array
350     X = np.zeros(
351         (ref_shp[0], ref_shp[1], nvar),
352         gdal_array.GDALTypeCodeToNumericTypeCode(
353             img_var[0].GetRasterBand(1).DataType
354         )
355     )
```

- *zeros*: devolve um novo *array* de formato e tipo preenchido com zeros.

- *GDALTypeCodeToNumbericTypeCode*: traduz um valor do tipo "numérico" num tipo de código GDAL.

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *DataType*: tipo de formato que se atribui a um ficheiro ou output.

```
    f = 0
    for r in range(len(img_var)):
        for b in range(img_bnd[r]):
            X[:, :, f] = img_var[r].GetRasterBand(b + 1).ReadAsArray()

            f += 1
```

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *ReadAsArray*: utilizado em *python* para ler *arrays*.

#Reshape (manipular)

#Prever

#Resultado manipulado

#Inserir valores *NoData*

```
    # Reshape
    nshp = (X.shape[0] * X.shape[1], X.shape[2])
    n_x = X[:, :, :nvar].reshape(nshp)

    # Predict
    y_cls = rf.predict(n_x)

    # Reshape result
    res = y_cls.reshape(X[:, :, 0].shape)

    # Place nodata values
    tmp = img_var[0].GetRasterBand(1).ReadAsArray()
    np.place(res, tmp==nd_val, 255)
```

- *shape*: os *arrays Numpy* têm um atributo chamado *shape* que geralmente é usado para obter a forma de um *array*, mas também devolve uma tupla que pode ser usada para alterar as dimensões do *array*.

- *reshape*: altera a nova forma do *array* sem alterar os seus dados.

- *predict*: utilizado para testar dados. Ajuda a prever as classes do conjunto de dados de treino com base nos valores previstos do modelo *random forest*.

- *GetRasterBand*: vai buscar o objeto (banda) para um conjunto de dados. O número da banda a ser retirada é de 1 a *GetRasterCount ()*.

- *ReadAsArray*: utilizado em *python* para ler *arrays*.

- *np.place*: altera os elementos de um *array* com base nos valores definidos e atribuídos como entrada.

Descrição:

Um dos códigos finais que realiza a classificação das imagens acedendo aos ficheiros dos modelos treinados produzidos no Programa 20. Este código acaba por ser muito semelhante ao executado no Programa 20, no entanto tem a finalidade de pegar nos ficheiros dos modelos e aplicar a classificação às imagens.

Imports dentro do código imgcls_from_mdl:

```
305     import os
306     from joblib import load
307     from osgeo import gdal, gdal_array
308     import numpy as np
309     from glass.wt.rst import obj_to_rst
310
311     if type(imgvar) != list:
312         # Check if it is a folder
313         if os.path.isdir(imgvar):
314             from glass.pys.oss import lst_ff
```

***Import* obj_to_rst dentro do código imgcls_from_mdl**

```
305     import os
306     from joblib import load
307     from osgeo import gdal, gdal_array
308     import numpy as np
309     from glass.wt.rst import obj_to_rst
310
311     if type(imgvar) != list:
312         # Check if it is a folder
313         if os.path.isdir(imgvar):
314             from glass.pys.oss import lst_ff
```

"*Array* para *Raster*"

```python
def obj_to_rst(inArray, outRst, template, noData=None, geotrans=None):
    """
    Send Array to Raster
    """

    from osgeo            import gdal, osr, gdal_array
    from glass.prop     import drv_name
    from glass.prop.rst import compress_option

    if type(template).__name__ == 'Dataset':
        img_template = template
    else:
        img_template  = gdal.Open(template)

    geo_transform = img_template.GetGeoTransform() if not geotrans else \
        geotrans
    rows, cols    = inArray.shape
    drv_n         = drv_name(outRst)
    driver        = gdal.GetDriverByName(drv_n)

    c_opt = compress_option(drv_n)
    if c_opt:
        out       = driver.Create(
            outRst, cols, rows, 1,
            gdal_array.NumericTypeCodeToGDALTypeCode(inArray.dtype),
            options=[c_opt]
        )
    else:
        out       = driver.Create(
            outRst, cols, rows, 1,
            gdal_array.NumericTypeCodeToGDALTypeCode(inArray.dtype)
        )
    out.SetGeoTransform(geo_transform)
    outBand       = out.GetRasterBand(1)

    if noData or noData==0:
        outBand.SetNoDataValue(noData)

    outBand.WriteArray(inArray)

    proj = osr.SpatialReference(wkt=img_template.GetProjection())

    if proj:
        out.SetProjection(img_template.GetProjection())

    outBand.FlushCache()

    return outRst
```

(Já explicados: *Import* obj_to_rst incluído no Programa 2)

Import lst_ff dentro do código imgcls_from_mdl

```
305     import os
306     from joblib import load
307     from osgeo import gdal, gdal_array
308     import numpy as np
309     from glass.wt.rst import obj_to_rst
310
311     if type(imgvar) != list:
312         # Check if it is a folder
313         if os.path.isdir(imgvar):
314             from glass.pys.oss import lst_ff
```

```
76  def lst_ff(w, file_format=None, filename=None, fnpart=None, rfilename=N
77      """
78      List the abs path of all files with a specific extension on a folde
79      """
80
81      from glass.pys import obj_to_lst
82
83      filename = None if filename and fnpart else filename
84
85      # Prepare file format list
86      if file_format:
87          formats = obj_to_lst(file_format)
88
89          for f in range(len(formats)):
90              if formats[f][0] != '.':
91                  formats[f] = '.' + formats[f]
92
93      # List files
94      r = []
95      for (d, _d_, f) in os.walk(w):
96          r.extend(f)
97          break
98
99      # Filter files by format or not
100     if not file_format:
101         if not rfilename:
102             t = [os.path.join(w, i) for i in r]
103         else:
104             t = [i for i in r]
105
106     else:
107         if not rfilename:
108             t = [
109                 os.path.join(w, i) for i in r
110                 if os.path.splitext(i)[1] in formats
111             ]
112         else:
113             t = [i for i in r if os.path.splitext(i)[1] in formats]
114
115     # Filter by filename
116     if not filename and not fnpart: return t
117
118     elif filename and not fnpart:
119         filename = obj_to_lst(filename)
120
121         _t = []
122         for i in t:
123             fn = fprop(i, 'fn') if not rfilename else i
124             if fn in filename:
125                 _t.append(i)
126
127         return _t
128
129     elif not filename and fnpart:
130         fnp = obj_to_lst(fnpart)
131
132         _t = []
133         for i in t:
134             fn = fprop(i, 'fn') if not rfilename else i
135
136             for _f in fnp:
137                 if _f in fn:
138                     _t.append(i)
139                     break
140
141         return _t
142
```

(Já explicados: *Import* lst_ff incluído no Programa 1)

Anexo E – Repositório de matrizes de confusão e tabelas de contingência

O respetivo anexo tem como fim apresentar os resultados obtidos mediante da classificação das imagens, representando as diferentes matries de confusão e tabelas de contigência dos dados combinados de classi_D4 e classi_D8, com a respetiva introdução dos diferentes dados de treino F0, F1, F2, F3 e F4.

Leganda das tabelas de contingência:

1 – *Sealed Surface*;

3 – *Herbaceous*

4 – *Forest*

5 – *Shrubland*

6 – *Partly vegetated or non-vegetated*

8 – *Water*

OA – *Overall Accuracy*

Tabela XXV – Matriz de confusão da classificação classi_D4_F0.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	16	10	7	1	2	36	44.4%	55.6%
3	7	339	21	12		379	89.4%	10.6%
4	1	12	129	4		146	88.4%	11.6%
5		4	14	19	1	38	50%	50%
6					1	1	100%	0%
Total geral	24	365	171	36	4	600		
Ext. Utilizador	66.7%	92.9%	75.4%	52.8%	25%	OA		
Erro de Comissão	33.3%	7.1%	24.6%	47.2%	75%	84%		
f1-score	53.3%	91.1%	81.4%	51.4%	40%			

Tabela XXVI – Matriz de confusão da classificação classi_D4_F1.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	11	12	1			24	45.8%	54.2%
3		347	14	4		365	95.1%	4.9%
4		25	132	14		171	77.2%	22.8%
5		12	4	19	1	36	52.8%	47.2%
6		2			2	4	50%	50%
Total geral	11	398	151	37	3	600		
Ext. Utilizador	100%	87.2%	87.4%	51.4%	66.7%	OA		
Erro de Comissão	0%	12.8%	12.6%	48.6%	33.3%	85.1%		
f1-score	62.9%	91%	82%	52.1%	57.1%			

Tabela XXVII – Matriz de confusão da classificação classi_D4_F2.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	5	19				24	20.8%	79.2%
3		363	2			365	99.5%	0.5%
4		161	5	5		171	2.9%	97.1%
5		21	2	13		36	36.1%	63.9%
6		2			2	4	50%	50%
Total geral	5	566	9	18	2	600		
Ext. Utilizador	100%	64.1%	55.6%	72.2%	100%	OA		
Erro de Comissão	0%	35.9%	44.4%	27.8%	0%	64.7%		
f1-score	34.5%	78%	5.6%	48.1%	66.7%			

Tabela XXVIII – Matriz de confusão da classificação classi_D4_F3.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	5					5	100%	0%
3	19	364	160	19	2	564	64.5%	35.5%
4		1	-			1	0%	0%
5			11	17		28	60.7%	39.3%
6					2	2	100%	0%
Total geral	24	365	171	36	4	600		
Ext. Utilizador	20.8%	99.7%	0%	47.2%	50%	OA		
Erro de Comissão	79.2%	0.3%	0%	52.8%	50%	64.7%		
f1-score	34.5%	78.4%	0%	53.1%	66.7%			

Tabela XXIX – Matriz de confusão da classificação classi_D4_F4.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	1					1	100%	0%
3	14	5	2	1		22	22.7%	77.3%
4	4	261	157	32		454	34.6%	65.4%
5	5	99	12	2	2	120	1.7%	98.3%
6					2	2	100%	0%
Total geral	24	365	171	36	4	600		
Ext. Utilizador	4.2%	1.4%	91.8%	5.6%	50%	OA		
Erro de Comissão	95.8%	98.6%	8.2%	94.4%	50%	27.8%		
f1-score	8%	2.6%	50.2%	2.6%	66.7%			

Tabela XXX – Matriz de confusão da classificação classi_D8_F0.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	16	10	7		2	35	45.7%	54.3%
3	7	337	20	12		376	89.6%	10.4%
4		14	130	4		148	87.8%	12.2%
5	1	4	14	19	1	39	48.7%	51.3%
6				1	1	2	50%	50%
Total geral	24	365	171	36	4	600		
Ext. Utilizador	66.7%	92.3%	76%	52.8%	25%	OA		
Erro de Comissão	33.3%	7.7%	24%	47.2%	75%	83.8%		
f1-score	54.2%	91%	81.5%	50.7%	33.3%			

Tabela XXXI – Matriz de confusão da classificação classi_D8_F1.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	11	11	1	1		24	45.8%	54.2%
3		346	15	4		365	94.8%	5.2%
4		24	133	14		171	77.8%	22.2%
5		12	4	19	1	36	52.8%	47.2%
6		2			2	4	50%	50%
Total geral	11	395	153	38	3	600		
Ext. Utilizador	100%	87.6%	86.9%	50%	66.7%	OA		
Erro de Comissão	0%	12.4%	13.1%	50%	33.3%	85.1%		
f1-score	62.9%	91.1%	82.1%	51.4%	57.1%			

Tabela XXXII – Matriz de confusão da classificação classi_D8_F2.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	5	19				24	20.8%	79.2%
3		363	2			365	99.5%	0.5%
4		164	3	4		171	1.8%	98.2%
5		20	1	15		36	41.7%	58.3%
6		2			2	4	50%	50%
Total geral	5	568	6	19	2	600		
Ext. Utilizador	100%	63.9%	50%	78.9%	100%	OA		
Erro de Comissão	0%	36.1%	50%	21.1%	0%	85.1%		
f1-score	34.5%	77.8%	3.4%	54.5%	66.7%			

Tabela XXXIII – Matriz de confusão da classificação classi_D8_F3.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	5					5	100%	0%
3	19	364	160	18	2	563	64.7%	35.3%
4		1	-			1	0%	0%
5			11	18		29	62.1%	37.9%
6					2	2	100%	0%
Total geral	24	365	171	36	4	600		
Ext. Utilizador	20.8%	99.7%	0%	50%	50%	OA		
Erro de Comissão	79.2%	0.3%	0%	50%	50%	64.8%		
f1-score	34.5%	78.4%	0%	55.4%	66.7%			

Tabela XXXIV – Matriz de confusão da classificação classi_D8_F4.

classes	1	3	4	5	6	Total geral	Ext. Produtor	Erro Omissão
1	1					1	100%	0%
3	14	6	2	1		23	26.1%	73.9%
4	4	263	155	23		445	34.8%	65.2%
5	5	96	14	11	2	128	8.6%	91.4%
6					2	2	100%	0%
Total geral	24	365	171	36	4	600		
Ext. Utilizador	4.2%	1.6%	90.6%	30.6%	50%	OA		
Erro de Comissão	95.8%	98.4%	9.4%	69.4%	50%	29.1%		
f1-score	8%	3.1%	50.3%	13.4%	66.7%			

Tabela XXXV – Tabela de contingência da classificação classi_D4_F0.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	118708	20151	2023	88	1863	31	142864	83.1%	16.9%
3	86013	2031959	124700	44590	9086	185	2296533	88.5%	11.5%
4	2977	23942	614191	62858	232	12	704212	87.2%	12.8%
5	1686	12881	111170	72651	4666	69	203123	35.8%	64.2%
6	12724	5754	981	255	17917	23	37654	47.6%	52.4%
8	8	56	41	3	3483	831	4422	18.8%	81.2%
Total geral	222116	2094743	853106	180445	37247	1151	3388808		
Ext. Utilizador	53.4%	97%	72%	40.3%	48.1%	72.2%	OA		
Erro Comissão	46.6%	3%	28%	59.7%	51.9%	27.8%	83.9%		
f1-score	65.0%	92.5%	78.9%	37.9%	47.8%	29.8%			

Tabela XXXVI – Tabela de contingência da classificação classi_D4_F1.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	72349	62570	2785	81	5063	16	142864	50.6%	49.4%
3	9657	2105235	126303	44350	10989	2	2296536	91.7%	8.3%
4	13	24456	616607	62796	339	1	704212	87.6%	12.4%
5	137	13556	111721	72652	5033	24	203123	35.8%	64.2%
6	10054	6815	718	139	19920	8	37654	52.9%	47.1%
8	3	37	73	2	3601	706	4422	16%	84%
Total geral	92213	2212669	858207	180020	44945	757	3388811		
Ext. Utilizador	78.5%	95.1%	71.8%	40.4%	44.3%	93.3%	OA		
Erro Comissão	21.5%	4.9%	28.2%	59.6%	55.7%	6.7%	85.2%		
f1-score	61.6%	93.4%	78.9%	37.9%	48.2%	27.3%			

Tabela XXXVII – Tabela de contingência da classificação classi_D4_F2.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	58526	82431	764	45	1091	7	142864	41%	59%
3	4986	2273637	6185	11341	383	6	2296538	99%	1%
4	12	649067	26360	28765	7	1	704212	3.7%	96.3%
5	110	123441	29220	50127	177	48	203123	24.7%	75.3%
6	8243	18052	616	284	10436	23	37654	27.7%	72.3%
8	9	629	53	5	2926	800	4422	18.1%	81.9%
Total geral	71886	3147257	63198	90567	15020	885	3388813		
Ext. Utilizador	81.4%	72.2%	41.7%	55.3%	69.5%	90.4%	OA		
Erro Comissão	18.6%	27.8%	58.3%	44.7%	30.5%	9.6%	71.4%		
f1-score	54.5%	83.5%	6.9%	34.1%	39.6%	30.1%			

Tabela XXXVIII – Tabela de contingência da classificação classi_D4_F3.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	58155	83429	247	67	955	11	142864	40.7%	59.3%
3	5019	2268874	2484	19870	284	9	2296540	98.8%	1.2%
4	11	656168	1089	46939	4	1	704212	0.2%	99.8%
5	133	117210	3538	82036	123	83	203123	40.4%	59.6%
6	8509	19819	137	401	8732	56	37654	23.2%	76.8%
8	30	896	0	9	2511	976	4422	22.1%	77.9%
Total geral	71857	3146396	7495	149322	12609	1136	3388815		
Ext. Utilizador	80.9%	72.1%	14.5%	54.9%	69.3%	85.9%	OA		
Erro Comissão	19.1%	27.9%	85.5%	45.1%	30.7%	14.1%	71.4%		
f1-score	54.2%	83.4%	0.3	46.6%	34.7%	35.1%			

Tabela XXXIX – Tabela de contingência da classificação classi_D4_F4.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	8338	92928	2560	31154	7677	207	142864	5.8%	94.2%
3	363	72917	1666086	556367	639	167	2296539	3.2%	96.8%
4	0	531	699503	4163	2	13	704212	99.3%	0.7%
5	7	1060	189218	11513	283	1042	203123	5.7%	94.3%
6	936	20282	975	5514	8197	1750	37654	21.8%	78.2%
8	17	416	81	158	1104	2646	4422	59.8%	40.2%
Total geral	9661	188134	2558423	608869	17902	5825	3388814		
Ext. Utilizador	86.3%	38.8%	27.3%	1.9%	45.8%	45.4%	OA		
Erro Comissão	13.7%	61.2%	72.7%	98.1%	54.2%	54.6%	23.7%		
f1-score	10.9%	5.9%	42.9%	2.8%	29.5%	51.6%			

Tabela XL – Tabela de contingência da classificação classi_D8_F0.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	118755	20069	2042	92	1875	31	142864	83.1%	16.9%
3	86093	2031695	125276	44098	9186	185	2296533	88.5%	11.5%
4	2987	23931	613717	63327	237	13	704212	87.1%	12.9%
5	1673	13170	111440	72043	4728	69	203123	35.5%	64.5%
6	12506	5816	1018	245	18042	27	37654	47.9%	52.1%
8	6	52	39	4	3504	817	4422	18.5%	81.5%
Total geral	222020	2094733	853532	179809	37572	1142	3388808		
Ext. Utilizador	53.5%	97%	71.9%	40.1%	48%	71.5%	OA		
Erro Comissão	46.5%	3%	28.1%	59.9%	52%	28.5%	84.2%		
f1-score	65.1%	92.5%	78.8%	37.6%	48%	29.4%			

Tabela XLI – Tabela de contingência da classificação classi_D8_F1.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	72343	62882	2918	88	4617	16	142864	50.6%	49.4%
3	9712	2104819	127300	43904	10799	1	2296535	91.7%	8.3%
4	15	24432	616122	63308	334	1	704212	87.5%	12.5%
5	136	13864	111986	72053	5061	23	203123	35.5%	64.5%
6	9981	6844	743	143	19936	7	37654	52.9%	47.1%
8	1	34	77	2	3599	709	4422	16%	84%
Total geral	92188	2212875	859146	179498	44346	757	3388810		
Ext. Utilizador	78.5%	95.1%	71.7%	40.1%	45%	93.7%	OA		
Erro Comissão	21.5%	4.9	28.3%	59.9%	55%	6.3%	85.2%		
f1-score	61.6%	93.4%	78.8%	37.7%	48.6%	27.4%			

Tabela XLII – Tabela de contingência da classificação classi_D8_F2.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	58806	82134	732	44	1141	7	142864	41.2%	5.8%
3	5092	2273435	6149	11478	379	5	2296538	995	1%
4	11	645663	28093	30438	6	1	704212	4%	96%
5	107	120248	30630	51911	184	43	203123	25.6%	74.4%
6	8156	17986	610	296	10588	18	37654	28.1%	71.9%
8	8	627	51	8	2949	779	4422	17.6%	82.4%
Total geral	72180	3140093	66265	94175	15247	853	3388813		
Ext. Utilizador	81.5%	72.4%	42.4%	55.1%	69.4%	91.3%	OA		
Erro Comissão	18.5%	27.6%	57.6%	44.9%	30.6%	8.7%	71.5%		
f1-score	54.7%	83.6%	7.3%	34.9%	40%	29.5%			

Tabela XLIII – Tabela de contingência da classificação classi_D8_F3.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	58505	83040	232	73	1004	10	142864	41%	59%
3	5185	2268387	2533	20131	296	8	2296540	98.8%	1.2%
4	12	648088	2709	53397	5	1	704212	0.4%	99.6%
5	139	114068	5267	83455	119	75	203123	41.1%	58.9%
6	8536	19632	145	373	8921	47	37654	23.7%	76.3%
8	23	885	0	5	2539	970	4422	21.9%	78.1%
Total geral	72400	3134100	10886	157434	12884	1111	3388815		
Ext. Utilizador	80.8%	72.4%	24.9%	53%	69.2%	87.3%	OA		
Erro Comissão	19.2%	27.6%	75.1%	47%	30.8%	12.7%	71.5%		
f1-score	54.4%	83.5%	0.8%	46.3%	35.3%	35.1%			

Tabela XLIV – Tabela de contingência da classificação classi_D8_F4.

classes	1	3	4	5	6	8	Total geral	Ext. Produtor	Erro Omissão
1	8434	93503	2630	30542	7523	232	142864	5.9%	94.1%
3	411	76392	1652372	566568	628	168	2296539	3.3%	96.7%
4	0	567	683604	20025	2	14	704212	97.1%	2.9%
5	7	1072	165272	35350	279	1143	203123	17.4%	82.6%
6	908	20296	954	5577	8132	1787	37654	21.6%	78.4%
8	15	459	65	160	1088	2635	4422	59.6%	40.4%
Total geral	9775	192289	2504897	658222	17652	5979	3388814		
Ext. Utilizador	86.3%	39.7%	27.3%	5.4%	461%	44.1%	OA		
Erro Comissão	13.7%	60.3%	72.7%	94.6%	53.9%	55.9%	24%		
f1-score	11.1%	6.1%	42.6%	8.2%	29.4%	50.7%			