Diogo Apóstolo

EXPLORING PROBLEMS OF OVERLAP AND DATASET SHIFT IN IMBALANCED DATA

UNIVERSIDADE D COIMBRA

1 2 9 0

1 2 9 0

UNIVERSIDADE D
COIMBRA

João Diogo de Sousa Jardim e Apóstolo

EXPLORING PROBLEMS OF OVERLAP AND DATASET SHIFT IN IMBALANCED DATA

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Professor Pedro Abreu and Miriam Santos and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

January of 2022

Faculty of Sciences and Technology

Department of Informatics Engineering

Exploring problems of overlap and dataset shift in imbalanced data

Diogo Apóstolo

January 2022

1 2 9 0

UNIVERSIDADE Ð
COIMBRA

This page is intentionally left blank.

Abstract

While it is known that imbalance on its own is not too harmful, when combined with other issues such as dataset shift and/or overlap, its impact on the degradation of the quality of data increases, becoming a real problem. Despite this, there is a lack of research works studying these issues simultaneously.

The aim of this work is to study both overlap and dataset shift in contexts of where the data is imbalanced, to understand their combined effects.

To study dataset shift, experiments were made using four cross validation algorithms, that induce different amounts of shift in the data, with the goal of understanding their impact on the performance of several machine learning algorithms. Furthermore, experiments were also done using multiple oversampling techniques to measure how much the combined effects of dataset shift and imbalance degrade the performance of these algorithms. Overall, the results confirmed that the joint effects of imbalance and dataset shift are very detrimental to the classifier's performance, with dataset shift occupying the main role in this equation, but that the use of oversampling algorithms could improve performance in some scenarios. This study also contested the current literature on the efficacy of cross validation algorithms in reducing dataset shift.

To study overlap, a new python package, pycol, was created, which aggregates most state-of-the-art complexity measures used to calculate overlap, with the goal of making it easier to experiment with these measures. Using this package a study was conducted validating a taxonomy proposed for these measures. Furthermore, the efficacy of multiple preprocessing algorithms in reducing overlap was tested in imbalanced datasets. The results revealed new information about the preprocessing algorithms and showed their usefulness in reducing multiple dimensions of overlap. The results also confirmed the validity of the families of overlap proposed in the aforementioned taxonomy, but they also indicate that each family is not independent of the others, meaning that there is some overlap degree between families.

Keywords

Imbalance, Dataset Shift, Overlap, Cross Validation, Complexity Measures

This page is intentionally left blank.

Resumo

Embora se saiba que existência de dados não balanceados não é muito prejudicial por si só, quando combinado com outros problemas como *dataset shift* e/ou *overlap*, o seu impacto na degradação da qualidade dos dados aumenta, tornando-se um problema real. Apesar disso, existe uma falta de estudos que analisem estes problemas simultaneamente.

O objetivo deste trabalho é estudar tanto o *overlap* como o *dataset shift* em contextos em que os dados são pouco balanceados, para perceber em mais detalhe os seus efeitos combinados.

Para estudar o *dataset shift*, foram realizadas experiências usando quatro algoritmos de *cross validation*, que introduzem diferentes quantidades de *dataset shift* nos dados, visando entender o seu impacto no desempenho de vários algoritmos classificação. Além disso, foram também realizadas experiências usando várias técnicas de *oversampling* para medir os efeitos combinados do *dataset shift* e dados pouco balanceados na degradação do desempenho desses algoritmos de classificação. Geralmente, os resultados confirmaram que os efeitos conjuntos dos dois problemas são muito prejudiciais ao desempenho dos classificadores, sendo que o *dataset shift* ocupa o papel principal nessa equação, porém o uso de algoritmos de *oversampling* pode melhorar o desempenho em alguns cenários. Este estudo também contestou a literatura atual sobre a eficácia de algoritmos de *cross validation* na redução do *dataset shift.*

Para estudar o *overlap*, foi criada uma biblioteca em *python*, *pycol*, que agrega a maioria das medidas de complexidade existentes no estado da arte usadas para medir o *overlap*, visando facilitar a experimentação com essas medidas. Utilizando a biblioteca foi realizado um estudo de modo a validar uma taxonomia proposta para estas medidas. Além disso, a eficácia do uso de vários algoritmos de pré-processamento na redução da *overlap* foi testada em *datasets* não balanceados. Os resultados revelaram nova informação sobre os algoritmos de pré-processamento testados e demonstraram a sua utilidade na redução de múltiplas dimensões de *overlap.* Os resultados também confirmaram a validade das famílias de *overlap* propostas na taxonomia supracitada, mas também indicam que cada família não é independente das demais.

Palavras-Chave

Dados não balanceados, *Dataset Shift*, *Overlap*, *Cross validation*, Medidas de complexidade

This page is intentionally left blank.

Contents

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

The introduction of industry 5.0, reinforced the idea that artificial intelligence should co-exist with humans in the workplace. If a company hopes to compete in the marketplace, it is key to merge aspects of automation brought by artificial intelligence and the cognitive skills of humans. With this in mind machine learning algorithms and data analysis became even more important to study than before. While this idea is true for all companies, it is the tech companies that by using their expertise in the domain can truly take advantage of this phenomenon, in fact large tech companies like Apple, Amazon and Microsoft individually rival the value of the largest companies in the oil and gas industry, greatly in part to their investment in machine learning algorithms and data analysis [1][2].

One important area of data analysis are knowledge discovery processes (KDP) also known as knowledge discovery in databases (KDD) which seek to find new knowledge in a domain [3]. More specifically they are defined as processes that extract useful and understandable patterns from data. These processes usually consist of sequential steps that range from the storing/fetching of data to visualization techniques to present the extracted information to the data miners. Particularly, in [3] the following steps of KDD are defined:

- **Understanding the application domain:** which consists in learning the prior knowledge of the problem and goals of the end user;
- **Creating a target dataset:** where the features and data samples to be examined in the data mining task are selected;
- **Preprocessing:** where the outliers and missing data are handled accordingly;
- **Data reduction:** which consists of finding useful features by applying feature reduction and transformation techniques;
- **Data mining:** in this step the data miner defines a task (such as classification, regression, clustering) according to the goals defined in step 1. Furthermore, the data miner selects the algorithms to search for patterns in the data and generates patterns according to this task, such as classification or regression models;
- **Interpreting the mining patterns:** where the data miner visualizes the extracted patterns and models;
- **Consolidation discovered knowledge:** where the knowledge extracted is incorporated into the target system or application.

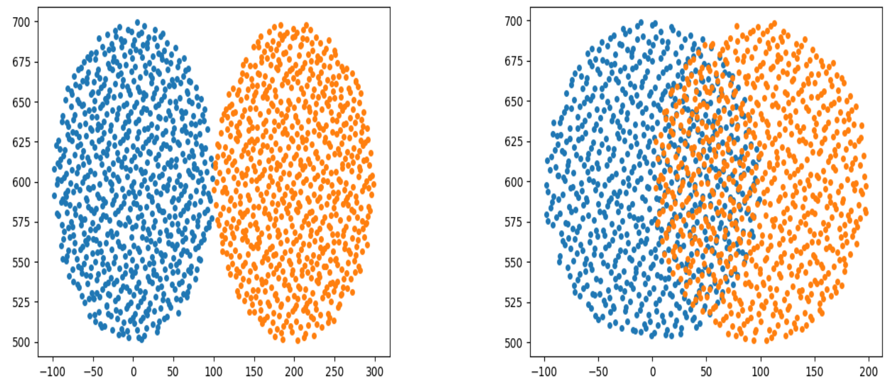
1.1 Context and Motivation

As datasets become more complex, analysing them manually became almost impossible and as such machine learning techniques have been applied to achieve that goal. However, most data available is not without its problems, which increases the complexity of the extraction of useful information. One problem, for instance, is missing data, which happens when some values on the dataset go missing due to data corruption or some mistake in recording [4][5]. Another issue is labelling noise, which is defined as incorrect labelling of some samples in a dataset. Furthermore, the feature values of the samples can also be inaccurate, potentially leading an algorithm to reach wrong conclusions. Accessibility is also a big issue, many times good datasets are not available to use by the public, or in more extreme cases have not been created yet.

The listed issues have been addressed in many research works, however this study will focus on the aspects of overlap [6][7][8][9] and dataset shift [10][11][12] in the context of imbalanced datasets, which will be described in more detail in the remainder of this section.

Most machine learning algorithms are resilient to an imbalanced class distribution. However, the aforementioned data problems become much more difficult when considering a skewed class distribution, especially when it comes to identifying the minority class, in a binary classification task.

Overlap has been studied over the years and can cause of many problems in pattern recognition tasks. It can essentially be defined as a region in the data space where different classes are present. Figure 1.1 shows an example of two datasets with low and high overlap. This context increases the complexity of the learning of the decision boundary task, which leads to misclassification of samples in the overlapped region [13].



(a) Low overlap

(b) High overlap

Figure 1.1: An example of two datasets with low and high overlap

While a high level definition of overlap is more or less agreed upon, how to measure it exactly still remains an open issue. One proposed method is thorough the use of dataset complexity measures/metrics [14]. However, overlap can be affected by multiple factors [6][13] like imbalance, data topology, dimensionality, and most measures only take into account one of these facets. Due to that, it is very difficult to obtain a metric that is capable of measuring overlap encompassing these multiple facets or to define a set of metrics that do so in conjunction [15].

Multiple works have made a comparison of various complexity measures to determine which can better characterize the datasets [16][17][14]. Most works attempt to either alter existing measures to better fit the problem domain or create new measures that compute overlap differently. In [15] a thorough review of different available complexity measures is done, not only providing their formulation but also each measure's main advantages and disadvantages. In [14] adaptations to these complexity measures were proposed for the context of imbalanced domains. One issue identified in some works is that many of the most popular complexity measures present low correlation between the overlapped calculated and the performance of the algorithm [14][17], indicating that the current set of measures needs to be further examined. Furthermore, a lot of complexity measures are formulated for simple binary problems with no categorical attributes, which makes them invalid to use in most real world datasets. Finally, a lot of other factors might influence how overlapped a problem is, for example imbalance, however many times these are not taken into account when comparing the performance between different sets of measures.

Another problem in real world data is dataset shift. In supervised machine learning a model is learned through the use of training data, however in real world scenarios it is possible that that model is no longer applicable in the new data of the testing set, implying that the data distribution in the two sets is not the same [10]. If this problem is not dealt with, then the models that were developed in the training set will be unable to perform with good results in the testing set. While in some cases the data might be inherently shifted, there are cases where the authors do not take this issue into account when creating the training and testing sets, causing dataset shift to occur (Figure 1.2). If this is considered, this type of shift can be mitigated with the proper division algorithms.

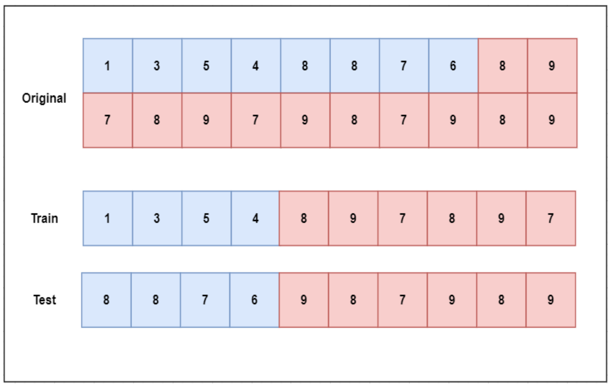


Figure 1.2: Example of covariate shift: When analysing the training dataset, it would appear the minority class (blue) only has values below 6 and the majority class (red) has values above. Although in the original dataset that is not the case has the minority class has values above 6 that end up showing in the test dataset.

Dataset shift has been tackled before, however not many works have tried to study its effects in the context of imbalanced datasets [10][18]. In [10] two cross validation algorithms, DBSCV and DOBSCV, are experimented on datasets with high imbalance ratio (IR) to study the effects of dataset shift on these types of datasets. Similarly, in [12] four cross validation algorithms, including the two previously mentioned ones, are experimented with, to establish which of them provides the least shift in the dataset and allows for a better performance overall.

Even though more works have been made in the area of dataset shift in imbalanced contexts, most works performed still do not take into account the two issues simultaneously [18], as such most algorithms and approaches to combat dataset shift might not be applicable in the context. Furthermore, even with the above-mentioned cross validation methods, it is not guaranteed that a classifier will achieve higher performances, as it can be influenced by other external factors like overlap, dimensionality and data noise [10].

1.2 Goals

While some studies have been made on the context of imbalance and dataset shift, the two problems are rarely studied together. Other works have identified that the performance of algorithms can degrade due to these issues individually, however it is important to study how each of them interact with each other, what are the consequences and potentially how they can be mitigated. With that in mind, the first goal of this work will be to:

> **Perform a sensibility analysis of the combination effects between dataset shift problems in imbalanced contexts**

With this goal in mind, the following research questions emerged in our research study:

- **RQ1: Is DOBSCV a cross validation algorithm that reduces more dataset shift in comparison to DBSCV, SCV and MSSCV?** Other studies [12] have tested these cross validation algorithms which combat dataset shift. DOBSCV seems to outperform other state-of-the-art methods in imbalanced contexts. However, due to the reduced number of used datasets, one of the main limitations of this study is the lack of generalization of its achieved results. Our research study aims to verify if this conclusion is generalizable for a setup with a higher number of datasets, multiple performance measures, with the results backed up by the use of dataset complexity measures;

- **RQ2: What is the impact that imbalance and dataset shift have on the performance of classifiers. Can oversampling algorithms increase their performance?** While it is known that both of these problems individually can degrade performance, few studies analyse them in the same context. Thus, it is important to study how these two data problems interact, so that solutions may be proposed for the problems found in this context.

To answer the first research question, multiple experiments were run with 4 classifiers (Support Vector Machine, kNN, C4.5 decision tree and Naive Bayes) on 83 datasets of the KEEL repository [19], trying out 4 different cross validation algorithms that influence the amount of dataset shift between partitions: SCV, DBSCV, DOBSCV and MSSCV. This experimental setup is similar to the one done in [12], the main differences consist in the number of used datasets (9 vs 83) and in also in the used performance metrics (1 vs 4). Finally, in our study, complexity measures [16] were also added to complement the achieved results something that has not been explored by the previous studies.

To answer the second research question, the original training datasets were also oversampled to test if decreasing the imbalance ratio in the training partition would help increase the performance even if dataset shift is present. In theory, oversampling the training dataset should make the minority class easier to classify, but if the concepts in training and testing are different due to dataset shift, then it is unclear how much these preprocessing algorithms will help in the resolution of this problem. Our study aims to verify in what cases the use of oversampling algorithms help reduce the joint impact of dataset shift and imbalance.

While many complexity measures already exist, even when used in conjunction they do not take into account every aspect that can influence overlap and thus do not give a complete picture of the dataset complexity. It is still unclear what types of overlap each metric exactly measures and while different taxonomies have been proposed [16][13] there is a lack of studies to validate those taxonomies. With is in mind the second goal of this research work is:

> **Validate the taxonomy proposed in [13] and study the effect of preprocessing algorithms on the different families encompassed in this taxonomy**

With that goal in mind, a new open-source python package that incorporates most state-of-the-art complexity measures was created, *pycol.* This package will make it easier to experiment with the metrics that exist currently, as most existing packages, are missing a large portion of current measures.

Using the pycol package the following research questions emerged:

- **RQ3: What is the validity of the taxonomy proposed in [13]?** The cited study divides overlap complexity measures into four families. This research work aims to validate the aforementioned taxonomy, by analysing if the match between metrics and families is correct;
- **RQ4: How effective are preprocessing algorithms in what concerns to overlap reduction?** In [13] some preprocessing algorithms are proposed to solve each family of overlap, but no experiments were produced to validate such fact. In this research work some of the proposed algorithms are tested to confront theory and practice [13].

To answer the third and fourth research questions some measures of each family of the taxonomy were selected along with 5 preprocessing algorithms that in theory address the types of overlap in question. The results of these measures were compared before and after the preprocessing is done. We aim to check if the preprocessing algorithms lower the metrics like expected and what is behaviour of metrics inside the same family. One scenario that could happen is that all the metrics of a family reduce their values except for one metric. This probably indicates that this metric it is not measuring the type of overlap intended, as all the remaining metrics lowered their value as expected. Furthermore, these results are also backed up by the use of a Random Forest (RF) classifying algorithm. If the decrease in overlap provided by the preprocessing algorithms is significative, then there should be a notable increase in performance after preprocessing.

1.3 Document Structure

This document is divided into 3 main sections: (i) State of the Art, where an overview of the current works on the area of overlap and dataset shift will be presented, as well as providing the necessary background knowledge to understand both of these problems; (ii) Experimental Setup and Results, where the experimental setup done for both the study in overlap and dataset shift is outlined. Furthermore, the results of both experiments and main takeaways observed as also shown; (iii) Conclusion, where the results of the experiments conducted are summarized, and steps to take in the future are outlined.

This page is intentionally left blank.

Chapter 2

State of the Art

In this section, a literature review of the current state of the art on both dataset shift and overlap in imbalanced contexts will be made. Furthermore, in a separate section, the overlap taxonomy [13] being studied will be detailed.

2.1 Dataset shift in imbalanced contexts

The problem of dataset shift can be described as a situation where the distributions of the data of the training and testing sets are different. This phenomenon can happen based on several reasons, although it is often due to sample selection bias or poor dataset division algorithms [18], and it can manifest in five different ways:

- **Covariate Shift:** In this case, the probability distributions of the inputs in the training and testing set change, but the conditional probabilities of the output values given the input values remains the same [20];
- **Prior probability shift:** In this case, the number of instances in each class varies greatly from training to test set, changing the prior probabilities from one set to the other [12];
- **Concept Shift:** Also known as concept drift, happens when the relationship between the input and the class variables changes over time, common in time series analysis [11];
- **Internal Covariate Shift:** This type of shift happens in multi-layered neural network architectures when the distributions produced from one layer to the next are too different [21];
- **Imbalanced Data:** Some literature recognizes imbalance as a type of dataset shift as well. More specifically the process of sampling done to make the minority class more visible to a model during training. Due to this scenario the imbalance ratio in the training is no longer the same as in testing which usually leads to performance increases in the minority class at the cost of decreases in the majority class [22].

Two different approaches can be taken to deal with dataset shift in imbalanced domains [18]. The first approach deals with intrinsic dataset shift, where the data itself contains some percentage of shift that degrades the algorithm's performance, for example the case of imbalance mentioned earlier. Some approaches have been proposed to reduce dataset shift either though pre-processing [23] or with ad hoc algorithms [24][25], neglecting the imbalance ratio.

The second approach deals with induced dataset shift, typically caused by inappropriate splitting of a dataset into training and testing sets, which can lead to covariate and prior probability shift. It is possible to induce covariate shift for example through the use of a stratified cross validation algorithm which randomly selects samples with no regard for their distribution in the feature space [12][10]. This study mainly focuses on these cross validation algorithms which help solve prior probability and covariate shift.

Cross validation (CV) is a common method used to divide the training and testing datasets, however standard CV does not address any kind of dataset shift. A better method is Stratified Cross Validation (SCV) which guarantees that the training and testing sets have the same number of samples of each class. This way the prior probability shift is completely avoided.

In [26] a new cross validation algorithm is introduced to combat covariate shift, distribution-balanced stratified cross-validation (DBSCV). This method attempts to separate the data into the folds not just randomly, but by giving each fold a similar example of the one given to the previous fold (Figure 2.1). That way, the distribution of each fold will be more similar when compared to SCV.

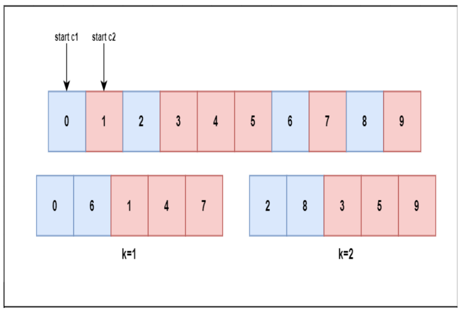


Figure 2.1: Example of DBSCV for 2 folds: For each class (blue and red) a starting sample (0 and 1) is chosen and assigned to the first fold, then the closest examples (2 and 3) are chosen and assigned to the next fold. This process repeats until there are no samples left.

An optimized version of DBSCV is proposed in [12], distribution optimized balanced stratified cross-validation (DOBSCV). While both algorithms are similar in their goal to reduce covariate shift by distributing samples of the same class as evenly as possible, DOBSCV is less sensitive to random choices since after assigning a sample to each of the k folds it picks a new random sample to start the process again (Figure 2.2).

Another baseline algorithm is also introduced Maximally-shifted stratified cross-validation (MSSCV), which is the opposite version of DBSCV, instead of assigning the closest sample to the next fold, it assigns the most distant (Figure 2.3).

In theory, this algorithm should perform the worst out of the ones tested, since it actively increases the dataset shift between partitions.

Both DBSCV, SCV along with traditional cross validation approaches were tested [26] in three artificial datasets and 9 real world datasets from the UCI data repository [27] by measuring the accuracy of a C4.5 decision tree. Results showed that DBSCV provides better results when compared to the other two CV methods. The experiments on the three artificial datasets also show a clear advantage for DBSCV, especially when the datasets

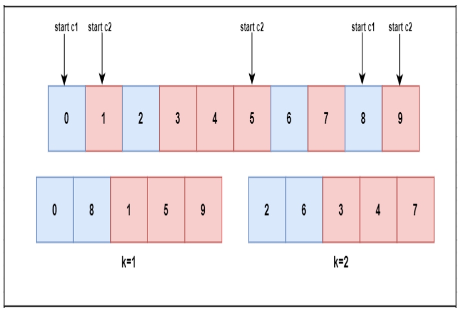


Figure 2.2: Example of DOBSCV for 2 folds: For each class (blue and red) a starting sample (0 and 1) is chosen and assigned to the first fold, then the closest examples (2 and 3) are chosen and assigned to the next fold. As all the folds have been assigned a sample, a new starting point is randomly chosen (5 and 8) and the process is repeated until all folds have been assigned a sample again. The process repeats one more time for the red class as it still has samples remaining

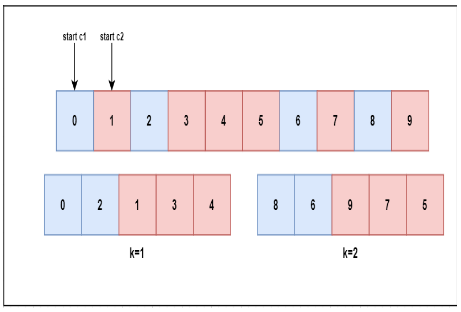


Figure 2.3: Example of MSSCV for 2 folds: For each class (blue and red) a starting sample (0 and 1) is chosen and assigned to the first fold, then the most distant examples (8 and 9) are chosen and assigned to the next fold. This process repeats until there are no samples left.

have multiple intraclass clusters. The authors propose further work analysing this cross validation method using other classification algorithms such as neural networks and by applying different distance metrics.

In [12] DBSCV, DOBSCV, MSSCV and SCV are all employed on 27 binary datasets from the KEEL repository including real and categorical features, the accuracy of 9 classifiers was measured on the partitioned datasets. Results confirmed that DBSCV is better suited to reduce the covariate shift when compared to standard SCV and that covariate shift can drastically impact the performance of classifiers as MSSCV performed much worse than the other methods. Finally, results also showed that the optimized version of DBSCV tends to have results that are slightly better than the regular version.

In [10] DOBSCV and SCV are tested in imbalanced domains, with the goal of validating that DOBSCV is more suitable as a cross validation strategy. The algorithms were tested in 66 real-world problems from the KEEL dataset repository, by evaluating the AUC performance of an SVM, kNN and C4.5 classifier. Results show that DOBSCV obtains better results overall and in datasets with a high imbalance ratio the difference between SCV and DOBSCV becomes more apparent. For future work, authors stated the use of other classification algorithms and the use of different distance metrics as possible directions to be explored.

2.2 Overlap in imbalanced contexts

Class overlap can be defined as the existence of instances with different class labels in the same region of the feature space [13]. Even though these instances belong to different classes, they share similar feature values, which can make them difficult to classify correctly.

It is common that in imbalanced domains the overlapped region is also severely imbalanced, containing mainly examples from the majority class, causing the decision boundary to shift towards the majority class, which is undesired in problems where misclassifications of the minority class have a very high cost, as it happens in many real world scenarios [6].

In [28] the effect of class overlap versus class imbalance is studied on binary artificial datasets to understand which of the two has a more negative impact on classification. Results, show that class overlap, seems to degrade classifier performance more than class imbalance, as overlap seems to always worsen the results, but imbalance only seems to affect the performance when overlap is present.

Similar experiments are done in [29] measuring the AUC of a C4.5 decision tree on artificial datasets with different degrees of overlap and imbalance. The results are similar to the previous work, as they show that imbalance on its own is not the sole factor to the decrease of performance of a classifier, and its negative effects are only truly noticeable when combined with class overlap.

Another study was made in [30], with the objective of measuring the effects of class imbalance and overlap, this time using the performance of kNN classifier on artificial datasets with different levels of overlap and imbalance. Similar conclusions are reached, in that the combination of both imbalance and overlap contribute to the degradation of the performance of the classifier.

In [31] this domain is studied by testing multiple classifiers: 1NN, MLP, Naive Bayes (NB), RBF, and C4.5 Decision Tree in order to understand the effect of class overlap and imbalance on different sets of classifiers. The experiments conducted are divided in two main scenarios: one where the data imbalance in the overlap region is the same as it the non overlapped region; a second scenario where the IR in the overlapped region is the inverse of the ratio on the rest of the dataset. Results show that kNN's performance is more dependent on the IR of the overlapped region than on the size of it, and that the overall imbalance ratio is not as important as the local imbalance ratio and as the size of the overlapped region. Results also show that the class more represented in overlap regions tends to be better classified by methods based on global learning, while the class less represented in such regions tends to be better classified by local methods.

While measuring imbalance has a precise formulation that is mostly agreed upon through the imbalance ratio, measuring overlap is done through the use of dataset complexity measures [13]. However, overlap can be influenced by multiple factors, like data typology, imbalance or dimensionality and most complexity measures do not take into account all of these factors. So not only do these measures individually not give a complete idea of how much overlap there is, there is also not an agreed upon set of measures that best works for most datasets. Beyond these problems there isn't an agreed upon taxonomy that identifies which type of overlap each of the metrics is sensitive to.

In the following section, an overview of the taxonomy proposed in [13] will be made. Presenting the complexity measures of each family, listing their advantages and disadvantages, as well as the different types of overlap each of them measure.

2.3 Complexity Measures

In [13] a taxonomy of overlap complexity measures is introduced, dividing them according to the type of overlap they measure: (i) Structural Overlap, (ii) Feature Overlap, (iii) Instance-Level Overlap, (iv) Multiresoltuion overlap. In the following sections, a brief explanation of these complexity measures will be done.

2.3.1 Feature Overlap

Feature overlap complexity measures categorize overlap over individual features of the dataset, by analysing the ability these features have to separate the classes.

F1

The F1 measure calculates the maximum Fisher's discriminant ratio among all features in a dataset [14], with the aim to determine how close two classes are (Equation 2.1):

$$F1 = max(f_k) \tag{2.1}$$

$$f_k = \frac{(\mu_{c_1} - \mu_{c_2})^2}{\sigma_{c_1}^2 - \sigma_{c_2}^2} \tag{2.2}$$

where μ_{c_i} and σ_{c_i} are the mean and standard deviation for feature k of class c_i.

In its original formulation F1 is a measure only for binary datasets, however over the years new formulations [15] have been proposed for multi-class problems, for example Equation 2.3.

$$f_k = \frac{\sum_{i=1}^{n_c} n_{c_i} * (\mu_{c_i} - \mu)^2}{\sum_{i=1}^{n_c} \sum_{j=1}^{n_{c_i}} (x_{ij} - \mu_{c_i})^2} \tag{2.3}$$

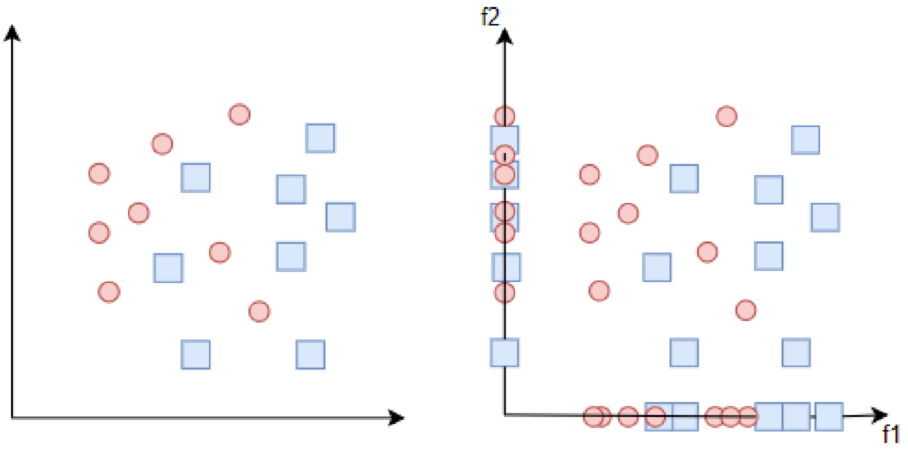
in this formulation μ_{c_i} is the mean of feature k for class c_i, n_{c_i} is the number of samples in class c_i, μ is the mean of feature k and x_{ij} is the value for feature k for an instance j, of class c_i. In [15] Equation 2.3 is quantified to have $O(m \cdot n)$ computational complexity, where m is the number of features in the dataset and n is the total number of samples in the dataset.

As the F1 measure does not have an upper bound, the following formulation was also proposed in order to bound its values in the interval from 0 to 1 [14]:

$$F1 = \frac{1}{1 + max(f_k)} \tag{2.4}$$

Using equation 2.4, low values of F1 indicate that one feature in the dataset can mostly separate the present classes, as it shows there exists little overlap between them on at least one dimension.

Figure 2.4 shows an example of this measure for a 2 dimensional plane, where both feature axis are used to project the 2D dataset. Out of the two features f1, is able to separate the dataset better, which means it has a lower f_k value.



(a) Original Dataset (b) Feature Projection

Figure 2.4: F1 example for a 2D dataset. The samples are projected on both features' axis in order to determine which of the two features better separates the classes

One problem with this metric is that only one feature is considered for the projection. If there exists a hyperplane that is oblique to the feature axis and also completely separates the feature space, its existence will not be reflected on the values of F1 obtained.

Another issue present with this measure is that there is no easy way to extend it for datasets with categorical features, due to it calculating means and standard deviations for each feature [32]. This is an issue that will be found in most of the F* complexity measures, as it will be shown next.

F1v

The F1v measure finds the vector projection that can best separate two classes considering the two class Fisher criterion dF [15], as shown in Equation 2.5:

$$dF = \frac{d^t B d}{d^t W d} \tag{2.5}$$

where B is the between-class scatter matrix, W is the within class scatter matrix and d is the directional vector where the samples are projected to. Equations 2.6, 2.7 and 2.8 detail how to calculate these values, where μ_{c_i} is the mean vector for all features of class c_i, p_{c_i} is the proportion of samples in class c_i and Σ_{c_i} is the scatter matrix of class c_i.

$$d = W^{-1}(\mu_{c_1} - \mu_{c2}) \tag{2.6}$$

$$B = (\mu_{c_1} - \mu_{c_2})(\mu_{c_1} - \mu_{c_2})^t \tag{2.7}$$

$$W = p_{c_1} * \Sigma + p_{c_2} * \Sigma \tag{2.8}$$

Similar to the F1 measure, in order to bound its values in the interval from 0 to 1, the following formulation is proposed:

$$F1v = \frac{1}{1 + dF} \tag{2.9}$$

Lower values of F1v indicate a small amount of overlap between classes, as it signifies that a linear hyperplane is capable of separating them.

As this measure is only considered for binary classification problems, for multi-class problems a one-vs-one (OvO) or one-vs-all (OvA) approach must be chosen. For a binary classification problem [15] quantifies this measure with $O(m \cdot n + m^3)$ temporal complexity and using an OvO approach a complexity of $O(m \cdot n \cdot n_c + m^3 \cdot {n_c}^2)$, where m is the number of features in the dataset, n is the number of samples and n_c is the number of classes. So while this measure can be seen as an improvement for F1, it is more computationally costly.

Finally, much like F1, this measure can not easily be computed for datasets with categorical features, as it must calculate means and standard deviations.

F2

The F2 measure uses the maximum and minimum values of each feature, f_i to calculate the volume of the overlapped region [15]. The value is normalized by calculating the ratio between the overlapping interval and the range of both classes, c_1 and c_2, as shown in Equation 2.10. Figure 2.5 shows an example of calculation of F2 for a 2 dimensional dataset.

$$F2 = \prod_{i=1}^{m} \frac{overlap(f_i)}{range(f_i)} = \prod_{i=1}^{m} \frac{max\,\{0, minmax(f_i) - maxmin(f_i)\}}{maxmax(f_i) - minmin(fi)} \tag{2.10}$$

where

$$\begin{aligned} minmax(f_i) &= min(max(f_{i,c_1}), max(f_{i,c_2})) \\ maxmin(f_i) &= max(min(f_{i,c_1}), min(f_{i,c_2})) \\ maxmax(f_i) &= max(max(f_{i,c_1}), max(f_{i,c_2})) \\ minmin(f_i) &= min(min(f_{i,c_1}), min(f_{i,c_2})) \end{aligned}$$

Higher values of F2 correspond to higher complexity and vice-versa. Using this formulation, if at least one feature of the dataset completely separates both classes, the value of F2 is 0 indicating that the dataset is linearly separable and therefore easy to classify.

The original formulation of this metric does not consider class imbalance, as such in [14] the following formulation is proposed, which calculates the F2 score per class:

$$F2_{c_i} = \prod_{i=1}^{m} \frac{max\,\{0, minmax(f_i) - maxmin(f_i)\}}{max(f_{i,c_i}) - min(f_{i,c_i})} \tag{2.11}$$

Much like the previous measure to use F2 in a multi-class scenario, an OvO or OvA approach must be taken. Besides this, Similarly, one of the other problems with this metric is that symbolic attributes must be converted to numeric in order to compute the maximum and the minimum values. Considering a OvO decomposition of this problem the temporal complexity of F2 is $O(m \cdot n \cdot n_c)$ [15], where m is the number of features, n_c is the number of classes and n is the number of samples in the dataset.

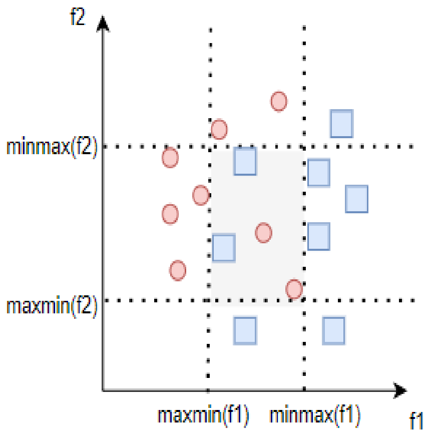


Figure 2.5: F2 Example. The area in grey represents the overlap region and its volume is the value for F2.

F3

For each feature, f_i, F3 calculates the number of samples, x_j, in regions where two classes overlap and divides this value by the number of total samples, n, obtaining a ratio of overlapped samples. In [14] a different formulation is proposed which calculates the minimum value of all the ratios of each feature (Equation 2.12). An example of F3 is shown in Figure 2.6

$$F3 = min(\frac{n_{overlap}(f_i)}{n}) \tag{2.12}$$

$$n_{overlap}(f_i) = |\{x_j \in f_i : x_j > maxmin(f_i) \wedge x_j < minmax(f_i)\}| \tag{2.13}$$

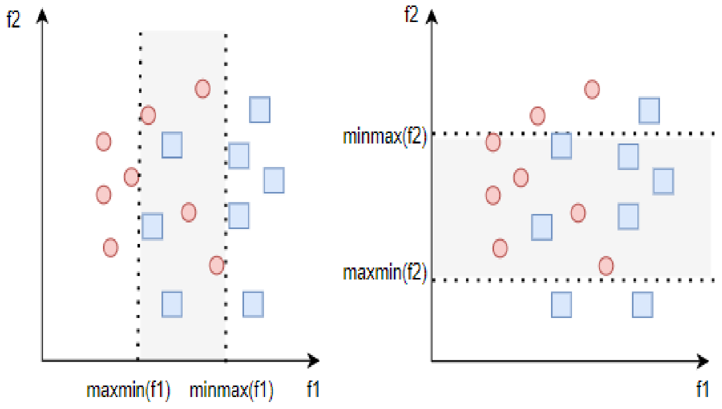


Figure 2.6: F3 Example for both features. The area in grey represents the overlap region in each feature. F3 is calculated according to how many samples are inside the overlapped region

The values of F3 are bounded from 0 to 1, where lower values indicate a less complexity, as there is at least one feature where there are not many samples in the overlap region.

This formulation, once again, does not consider class imbalance, as such even if the minority class in completely inside the overlap region the values of F3 can still indicate low complexity if the ratio of imbalance if high, a different formulation has been proposed in [14] to combat this problem, calculating F3 per class instead of globally:

$$F3_{c_i} = min(\frac{n_{overlap,c_i}(f_i)}{n_{c_i}}) \tag{2.14}$$

Much like F2 described previously, F3 shares the same problems with symbolic features and multiple class problems for the same reasons. F3 also shares the same temporal complexity as F2 [15].

F4

While F3 focuses on individual features, F4 calculates the discriminatory power of all features [14]. As such, a similar procedure to F3 is applied multiple times for each individual feature. An example of F4 being calculated is shown in Figure 2.7

To start, the most discriminative feature according to the F3 measure is determined, all the samples that can be correctly separated using this feature are removed from the dataset, along with the feature chosen. This process is repeated for remaining features and samples in the dataset, until either there are no more features or no more samples to remove. F4 is then found by calculating the ratio of remaining samples and the total samples in the original dataset.

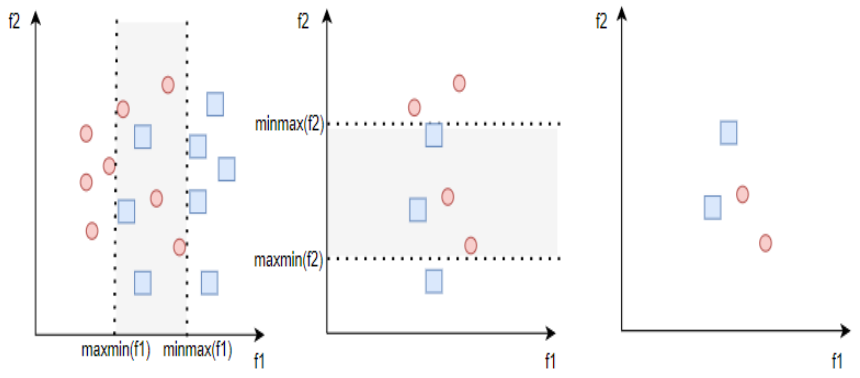


Figure 2.7: F4 Example. Similarly to F3 the samples that can be correctly classified by each individual feature are calculated. Afterwards these samples are removed. This process is repeated for all features and the remaining samples are used to calculate F4

Much like the F3 measure, F4 is bounded in the interval from 0 to 1 where higher values indicate higher overlap and vice versa. F4 also shares the same problems has F3 when it comes to class imbalance, multiclass problems and symbolic attributes. A similar approach to the one used in F3 can be used to mitigate the problem of class imbalance in each step of the iteration, as proposed in [14]. Because F4 applies the F3 measure multiple times for each feature the temporal complexity of this method is $O(m^2 \cdot n \cdot n_c)$ [15], where m is the number of features in the dataset and n_c is the number of classes. So while this measure is more powerful than F3 it takes longer to compute.

Input Noise

The input noise metric determines the amount of samples that fall inside another class' boundaries [13]. To calculate this, for each class, its boundaries are determined for every feature, then it counts, for each sample, x_i, in how many dimensions a sample falls into the boundaries of another class. This value is then normalized by $n * m$, where n is the total amount of samples and m is the amount of features in every sample (Equation 2.15).

$$IN = \frac{\sum_i^n o(x_i)}{n * m} \tag{2.15}$$

where $o(x_i)$ is the number of dimensions for sample x_i where it falls in the boundaries of other classes.

This measure shares similar issues with the previously described measures when it comes to symbolic attributes and class imbalance. This measure calculates a value for each sample for every feature, therefore the temporal complexity of this method for a binary problem is $O(m \cdot n)$.

2.3.2 Structural Overlap

This set of measures takes into account the morphology of the data by analysing the internal structure of the classes.

N1

The N1 measure builds a minimum spanning tree for all the samples in the dataset, by considering the whole dataset as a complete graph. Afterwards it analyses each edge of the MST and counts how many of them link vertexes of different classes [14]. An example of this process is show in Figure 2.8. To normalize this value, it is divided by the total amount of samples (Equation 2.16).

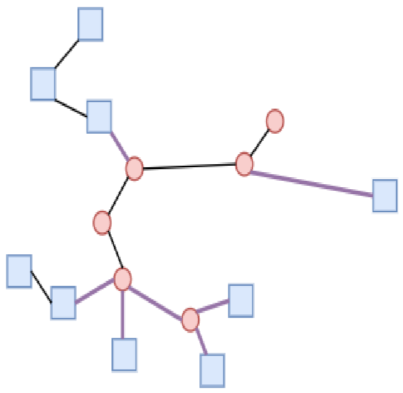


Figure 2.8: N1 Example. The edges that link two samples of a different class are marked in purple.

$$N1 = \frac{1}{n} | \{x_i : \exists (x_i, x_j) \in E \wedge y_i \neq y_j\} | \tag{2.16}$$

where x_i is a dataset sample, n is the total number of samples, E represents all the edges in the MST and y_i is the class label of sample x_i.

A higher value of N1 indicates that the classes are more intertwined, as many edges on the MST link two samples of different classes.

Similarly to measures described before, N1 has a bias for the majority class, as the normalization factor does not consider class imbalance. Following the same logic as for the F2, F3 and F4 metric in [14] a per class N1 metric is proposed to mitigate this problem.

$$N1_{c_i} = \frac{1}{n_{c_i}} |\, \{x_{i,c_i} : \exists (x_{i,c_i}, x_j) \in E \wedge c_i \neq y_j\} \,| \tag{2.17}$$

For multi-class domains, this metric can be applied in its presented formulation, as it is generic enough that the number of classes does not make an impact.

Depending on the distance metric used to calculate the MST, N1 may or may not have issues with symbolic features, as such these features must either be converted to numerical ones, or the distance metric must be able to deal with symbolic attributes.

As described in [32] one problem with this metric is that MSTs are not unique for a set of points, as such different N1 values are obtained depending on the order of samples presented to the MST algorithm. To combat this problem, it is suggested to run the N1 measure multiple times with the points shuffled each time and calculate the mean of all runs.

Finally, another problem arises for the instances in the boundary between 2 classes, even if they are linearly separable, if the samples in the boundary as closer to the samples of the opposite class rather as samples of its own class, then N1 might have high values even though the dataset is in practice not complex to classify.

In [15] this measure is calculated to have $O(m \cdot n^2)$ temporal complexity, where m is the number of features in the dataset.

N2

The N2 measure calculates the ratio between the inter and intra class distances [14]. To this end the distance, d, of every sample, x_i, to its nearest neighbour from the same class, NN_{c_1} is calculated, as well as the distance between every sample to the nearest neighbour from the opposite class, NN_{c_2}, as showed in Equation 2.18. An example of the calculation of N2 is shown in Figure 2.9.

$$r = \frac{\sum_i^n d(x_i, NN_{c_1})}{\sum_i^n d(x_i, NN_{c_2})} \tag{2.18}$$

Then the value of N2 is bounded between 0 and 1 by the following formula:

$$N2 = \frac{r}{1+r} \tag{2.19}$$

Higher values of N2 indicate a higher overlap, as it signifies that in most cases the distance between the nearest neighbour from the opposite class is smaller than the distance to the nearest neighbour of the same class.

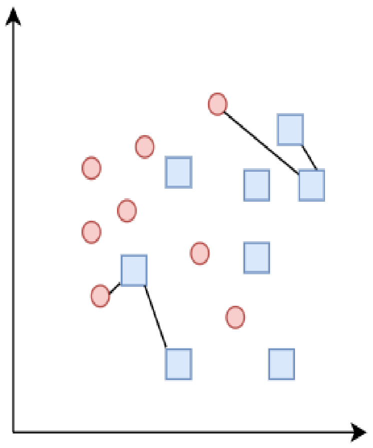


Figure 2.9: N2 Example for 2 samples. The rightmost blue sample is not in the overlap region, so its nearest neighbours from the same class is closer to it than its nearest neighbour from the opposite class. The leftmost blue sample is in the overlap region, and it is much closer to its nearest neighbour of the opposite class. These distances would be calculated for every sample to obtain the value of N2

N2 shares a similar problem to N1 if the samples are spread over a long, thin structure across the boundary [32]. Since most samples will be closer to ones of the opposite class, giving high values of N2 even though the problem might be linearly separable.

To deal with class imbalance, similarly to N1, the following alteration is also proposed in [14] to calculate N2 on a per class basis:

$$r_{c_1} = \frac{\sum_i^{n_{c1}} d(x_i, NN_{c_1})}{\sum_i^{n_{c2}} d(x_i, NN_{c_2})} \tag{2.20}$$

As for symbolic attributes, N2 works the same way as N1 and other distance based complexity measures. This measure shares the same temporal complexity as N1.

T1

T1 recursively expands hyperspheres around each sample in the dataset, each hypersphere grows until it collides with a hypersphere centred around a sample of different class, at this point both hyperspheres stop expanding [14]. This algorithm finishes when no more spheres can be grown, and it removes the hyperspheres that are completely contained inside another. A example of T1 can be seen in Figure 2.10 and in Appendix B an example of the full calculation can be found. The value for T1 is then the ratio of the amount of spheres remaining and the total number of samples, n in the dataset (Equation 2.21)

$$T1 = \frac{\#Hyperspheres}{n} \tag{2.21}$$

If there is a small amount of overlap then the number of hyperspheres is small and the value of T1 will be close to 0, on the other hand if there is a lot of overlap there will be a lot of hyperspheres remaining and the value of T1 will be close to 1.

Similarly to measures described before, a per class measure is proposed in [14] to combat class imbalance.

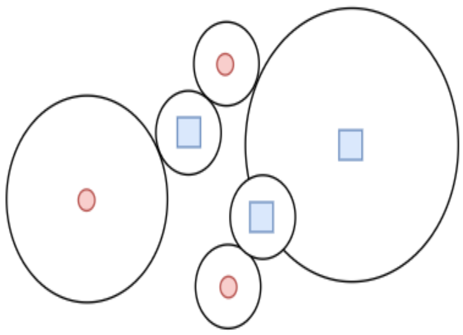


Figure 2.10: T1 example. Each sample extended a sphere centred on itself until it found another sphere from a sample of the opposite class.

T1 shares similar issues for categorical features as measures described before, since it has to consider each sample as a hypersphere centre. This geometrical interpretation of the samples is in part lost when considering categorical attributes.

In [15] the temporal complexity of T1 is calculated as $O(m \cdot n^2)$, where m is the number of features in the dataset. This value is the same as the N1 and N2 metrics.

NSG

NSG is similar to T1 however instead of calculating the ratio between hyperspheres and total number of samples, it calculates the average of samples per hypersphere [33]:

$$NSG = \frac{1}{\#Hyperspheres} \sum_{i}^{\#Hyperspheres} N_i \tag{2.22}$$

where N_i is the number of samples inside hypersphere i.

Similar adjustments made to T1 can be made to calculate a per class value of NSG, furthermore it shares the same problem for categorical features, for the same reasons. This measure also shares the same temporal complexity as T1.

ICSV

Another measure similar to T1 is ICSV, which also computes the hyperspheres that cover the dataset the same way. The density of each hypersphere is calculated according to the following equation, where N is the number of samples inside a hypersphere, H, and V is the volume of the hypersphere [33]:

$$\rho = \frac{N}{V} \tag{2.23}$$

Then the ICSV value is given by the standard deviation of the densities of all hyperspheres, as shown in Equation 2.25, where n_H is the number of hyperspheres.

$$\mu_\rho = \frac{1}{n_H} \sum_{i=1}^{n_H} \rho_i \tag{2.24}$$

$$ICSV = \sqrt{\frac{1}{n_H} \sum_{i=1}^{n_H} (\rho_i - \mu_\rho)^2} \tag{2.25}$$

High values of ICSV signify great changes in the densities, signifying a more complex domain.

This measure is complementary to T1 and NSG, as in where these compute some sort of mean with the hyperspheres, ICSV calculates the standard deviation, which gives a measure of variation in scale in the feature space [33].

ONB

ONB uses a similar method to T1 to compute its value, by finding a set of hyperspheres that completely cover the dataset, where each hypersphere stops growing when it finds an example of the opposite class [34].

After this algorithm is applied, two measures can be computed ONB_{tot} and ONB_{avg}. The first measure is similar to T1, the ratio between the number of hyperspheres necessary to cover the dataset and the total number of samples n:

$$ONB_{tot} = \frac{\#Hyperspheres}{n} \tag{2.26}$$

While the ONB_{tot} and $T1$ measures might seem to calculate the same value, ONB_{tot} uses a different hypersphere coverage algorithm, where a sample does not need to be completely covered by another to be removed. This allows it to achieve more minimal coverage and more optimal boundaries [34].

The second measure determines the average of hyperspheres on a per-class basis, for all classes in the dataset:

$$ONB_{avg} = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{\#Hyperspheres_{c_i}}{n_{c_i}} \tag{2.27}$$

Both these measures suffer from the same downsides as T1 when it comes to datasets with categorical features. This measure also shares the same temporal complexity as T1.

LSC

In [15] the local set of a sample x_i is defined as the set of points whose distance to x_i is smaller than the distance from x_i to the nearest neighbour of the opposite class, also known as its nearest enemy (Equation 2.28)

$$LS(x_i) = \{x_j : d(x_i, x_j) < d(x_i, ne(x_i))\} \tag{2.28}$$

where $ne(x_i)$ calculates the nearest enemy of x_i.

The number of examples in the local set may be an indicator of how close a sample is to the decision boundary. If the LS cardinality is lower, the example may either be near the decision boundary or in a region surrounded by examples of the opposite class.

Using 2.28 to calculate the local set of every sample, it is possible to summarize these values as follows, where n represents the total number of samples:

$$LSC = 1 - \frac{1}{n^2} \sum_{i=1}^{n} |LS(x_i)| \tag{2.29}$$

Higher values of LSC indicate a more complex dataset, indicating that on average the cardinality of the LS is small. In the most complex case the cardinality of all local sets will be 1, indicating that the classes are completely overlapped, in the scenario the value for LSC will be $1 - \frac{1}{n}$.

LSC can be deeply affected by labelling noise [17] and outliers, as if a single sample is in a region mostly dominated by examples of the opposite class, these samples will have a lower cardinality because they are influenced by this single sample.

Another potential problem pointed in [17] is that, as LSC considers the local density of each sample, it is possible, that samples in the decision boundary can have a very high LS cardinality if that region is very dense, yielding misleading results. This problem might be even more amplified in the cases of high imbalance, as such, using a per-class measure as shown before might be beneficial.

This measure shares the same temporal complexity as T1, N1 and N2 [15].

Clst

This measure follows a similar approach to T1 by determining the number of clusters that cover the dataset [17]. The algorithm starts by calculating the local set of each sample, similarly to LSC. Then the samples are ordered in descending order according to the size of their local set. The samples with highest LS cardinality are chosen to be the cluster core. If a sample is part of the LS of a cluster core then belongs to that cluster, otherwise if it does not belong to the LS of any cluster core, a new cluster is created with this sample as its core. This process is repeated for every sample until all of them have been assigned to a cluster. The final Clst value is given by Equation 2.30, where n is the number of samples in the dataset.

$$Clst = \frac{Clusters}{n} \tag{2.30}$$

Much like LSC, Clst can be heavily affected by labelling noise [17] as such it is proposed that these outlier values are removed before these metrics are applied.

DBC

DBC can be seen as an extension of T1 and N1 measures described previously. It starts by calculating the set of hyperspheres that completely cover the dataset much like in T1, afterwards using the hypersphere centres it calculates a MST and counts the number of vertexes of different classes that are linked by an edge, much like in N1. [35] The value for this measure is the computed as follows:

$$DBC = \frac{N_{inter}}{\#Hyperspheres} \tag{2.31}$$

where N_{inter} is the number of vertexes of different classes linked by an edge in the MST, and $\#Hyperspheres$ is the total number of hyperspheres needed to cover the dataset.

This results will give us a good indication of the complexity of the decision boundary, much like N1, if the value of DBC is high, then there is a high degree of interleaving between classes.

This measure shares the same issues as N1 described previously, as it uses a similar formulation. And in terms of temporal complexity it combines both the complexity of T1 and N1.

2.3.3 Instance Level Overlap

These measures analyse the domain on a local level, usually by calculating some value considering the error of a k nearest neighbour classifier. Many of these measures analyse each sample individually, and an overall complexity measure is calculated by averaging all the values.

R-Value

The R-Value calculates the ratio between the points that are in the overlap region and the total number of points in the dataset [36]. The overlap region is defined by looking at the set of nearest neighbours for every sample. Samples that are surrounded by others of the opposite class are considered to be in the overlap region. A sample, x_m, of class C_i, i.e. x_{im}, is considered surrounded by samples of the opposite class, C_j, when the number of its nearest neighbours that belong to class C_j, $|kNN(x_{im}, C_j)|$, is above a certain threshold θ. The following equation translates this idea, where the λ function is equal to 1 if its argument is above 0 and equal to 0 otherwise.

$$r(C_i, C_j) = \sum_{m=1}^{|C_i|} \lambda(|kNN(x_{im}, C_j)| - \theta) \tag{2.32}$$

Then using these values we can calculate the R-value, i.e. the number of samples in the overlapped region, the following way:

$$R(C_i, C_j) = \frac{r(C_i, C_j) + r(C_j, C_i)}{|C_i| + |C_j|} \tag{2.33}$$

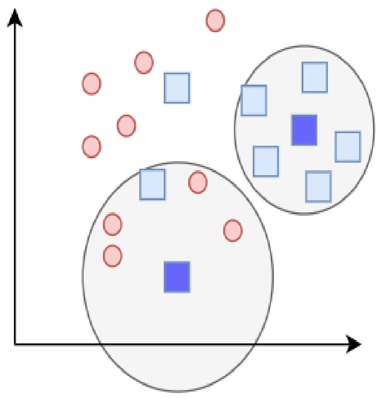


Figure 2.11: R-Value Example for two samples for k=5 and θ=1. The samples marked in dark blue search for its 5 nearest neighbours. The bottom sample is surrounded by samples of the opposite class so its λ value is equal to 1, while the other sample is surrounded by others of the same class so its λ is equal to 0. This process is repeated for every sample and the R-Value is calculated according to the equations mentioned previously

The R-Value ranges from 0 to 1, where 0 indicates that no samples are in the overlapped region and 1 indicates that the two classes C_i and C_j are completely overlapped.

Using this formulation the results can be biased towards the majority class, as shown in [37], where even though the value of AUC decreased drastically as the imbalance ratio increased, the value of R stayed almost constant. These experiments lead to the conclusion that the contribution of the majority class should not be proportional to the number of negative instances, since most of its instances are not in the overlapped region.

As such, in a new formulation was proposed to take this factor into account.

$$R_{aug} = \frac{1}{1+IR}(IR * r(C_{neg}, C_{pos}) + r(C_{pos}, C_{neg})) \tag{2.34}$$

where IR is the imbalance ratio and $r(C_i, C_j)$ is now calculated using:

$$r(C_i, C_j) = \frac{1}{|C_i|}\sum_{m=1}^{|C_i|} \lambda(|kNN(x_{im}, C_j)| - \theta) \tag{2.35}$$

In cases where IR=1, this formulation is equal to the one presented previously.

N3

N3 represents the ratio between the number of samples, x_i, that have its nearest neighbour, NN_i, belonging to a different class than that of its own and the total number of points in the dataset, n, i.e. the error rate of a 1NN classifier [14]. Equation 2.36 demonstrates this concept formally.

$$N3 = \frac{1}{n} |\{x_i : y_i \neq y_{NN_i}\}| \tag{2.36}$$

N3 is bounded in the interval from 0 to 1, where values close to 0 represent a low error rate and vice-versa. Less complex domains will, in theory, have lower values for N3.

In [14] a new formulation for N3 was proposed to combat class imbalance, by calculating N3 on a per-class basis, similarly to N1 and N2 described previously:

$$N3_{c_i} = \frac{1}{n_{c_i}} | \left\{ x_{i,c_i} : y_{i,c_i} \neq y_{NN_{i,c_i}} \right\} | \tag{2.37}$$

In [15] the temporal complexity of this measure is calculated to be $O(m \cdot n^2)$, where m is the number of features in the dataset.

N4

The N4 measure starts by creating a new test dataset, by interpolating samples from the same class from the original dataset. This measure is very useful to use alongside N3 as it shows how resilient the dataset is to the introduction of new samples. The synthetic samples have the same class label of the two samples used to create it. For each new sample, $x_i^{'}$, its nearest neighbour in the original dataset is determined, NN_i, afterwards their class labels are compared and N4 is computed [14]. Equivalent to predicting the error rate on the test set, by using the original dataset as the training set. Equation 2.38 describes this process formally and Figure 2.12 shows an example for 3 samples.

$$N4 = \frac{1}{n} | \left\{ x_i^{'} : y_i^{'} \neq y_{NN_i} \right\} | \tag{2.38}$$

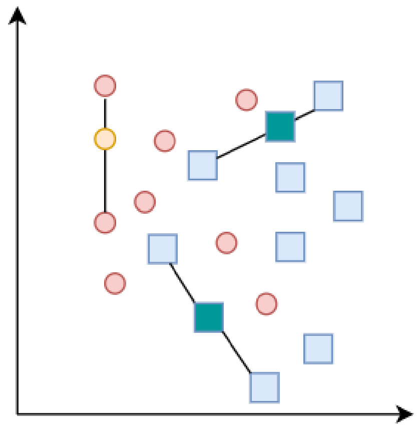


Figure 2.12: N4 example showing 3 artificial samples being generated marked at yellow and dark blue. These artificial samples were interpolated using the two other samples that they are linked to in the figure. Afterwards these artificial samples are classified using a 1NN classifier and N4 is given by the error of the classifier.

Similarly to N3, values of N4 are bounded in the interval from 0 to 1, where lower values signify lesser complex domains.

The same formulation for imbalanced domains followed in N3 can be applied to N4 [14]:

$$N4_{c_i} = \frac{1}{n_{c_i}} | \left\{ x_{i,c_i}^{'} : y_{i,c_i}^{'} \neq y_{NN_{i,c_i}} \right\} | \tag{2.39}$$

In [15] the temporal complexity of this measure is calculated to be $O(m \cdot n^2)$, where m is the number of features in the dataset.

degOver

Similarly to the R Value in [38] the degree of overlap is calculated by calculating the k nearest neighbours (k=5) of every sample, x_i. If all its 5 nearest neighbours belong to the same class then the sample in is a non overlap region, otherwise the sample belongs to an overlapped region.

Afterwards the ratio between the overlapping examples, $n_{overlap}$ and the total number of points, n, is calculated:

$$degOver = \frac{n_{overlap}}{n} \tag{2.40}$$

Values of degOver are bounded in the interval from 0 to 1, where lower values indicate less overlapped domains, however this metric suffers from the same problem as R value in the context of imbalanced domains, as the imbalance ratio is not considered in the formula presented.

Separability Index

The Separability Index (SI) is similar to N3, however instead of calculating the error rate of a 1NN classifier, it calculates its accuracy [39][40]:

$$SI = \frac{1}{n} | \{x_i : y_i = y_{NN_i}\} | \tag{2.41}$$

This complexity measure suffers from the same drawbacks as N3, as it is its complementary measure. It is, however, possible to calculate a similar per class value of SI, much like it was done with N3 to combat the problem of class imbalance. This measure also shares the same temporal complexity as N3.

kDN

k-Disagreeing Neighbours (kDN) calculates the ratio of the nearest neighbours of a sample, x_i that do not share its class [41]:

$$kDN(x_i) = \frac{1}{k} | \{x_v : y_v \neq y_i\} | \tag{2.42}$$

In this equation x_v is a sample belonging to the k nearest neighbours of x_i. x_v has class label y_v and x_i has class label y_i.

Values close to 0 signify that the sample is surrounded by other samples of its class and is not in an overlapped region. While values close to 1 may signify that the sample is close to the decision boundary.

A global measure for the entire dataset can be calculated using the following equation where n represents the total number of samples in the dataset:

$$kDN_{avg} = \frac{1}{n} \sum_{i=1}^{n} kDN(x_i) \tag{2.43}$$

CM

The CM measure is similar to the one previously described, as it considers the k nearest neighbours for every sample in the dataset. In this case, an example is considered to be in the overlapped region if more than half of its nearest neighbours are of the opposite class [14].

CM can then be defined in relation to kDN as follows [13]:

$$CM = \frac{1}{n} | \{x_i : kDN(x_i) > 0.5\} | \tag{2.44}$$

In [14] a variation of CM is defined, that considers only the minority class in the overlapped region, since class imbalance tends to hide overlap of the less represented class, as mentioned before. The metric is then defined as:

$$CM = \frac{1}{n_{c_{min}}} | \{x_{i,c_{min}} : kDN(x_{i,c_{min}}) > 0.5\} | \tag{2.45}$$

Nearest neighbour approaches like the CM measure can lead to erroneous results for datasets where the nearest neighbours vary their distances widely, as such the closest neighbours should have more weight while computing the metric. To that end, in [14] extensions of CM that use a weighted knn classifier are introduced, weighted CM (wCM) and dual weighted CM (dwCM). With the wCM measure, each neighbour j of a sample of the minority class i is given a weight based on their distance. This weight is normalized using the distance to the closest $d(x_i, x_1^{NN})$ and most distant neighbour $d(x_i, x_k^{NN})$ as follows:

$$W_{ij} = \frac{d(x_i, x_k^{NN}) - d(x_i, x_j^{NN})}{d(x_i, x_k^{NN}) - d(x_i, x_1^{NN})} \tag{2.46}$$

dwCM works similarly to CM, however each sample x_i is dually weighted, to increase the robustness to the choice of the value of k, as follows:

$$W_{ij} = \frac{d(x_i, x_k^{NN}) - d(x_i, x_j^{NN})}{d(x_i, x_k^{NN}) - d(x_i, x_1^{NN})} * \frac{d(x_i, x_k^{NN}) + d(x_i, x_j^{NN})}{d(x_i, x_k^{NN}) + d(x_i, x_1^{NN})} \tag{2.47}$$

In [14] the CM measures prove to have more correlation to the sensitivity metric when compared to F1, F2 and N2 when tested in the KEEL and UCI repository datasets. Values for CM, wCM and dwCM were almost 0 for datasets where the sensitivity was 1 and vice versa. Out of the 3 variations, the correlation was higher for dwCM.

D3

Another measure very similar to kDN and CM is the D3 measure, which aims to measure the density of each class in the overlap region [42]. As such, for each class, D3 is defined as:

$$D3_{c_j} = | \{x_i : 1 - kDN(x_i) < 0.5\} | \tag{2.48}$$

High values of $D3_{c_j}$ indicates that are few samples of class c_j are in the overlap region, signifying that most samples are in a safe region. Much like the previous metrics, it is important to pay special attention to the D3 value of the minority class, since a high value of D3 in the majority class does not imply the same for the minority class.

Similarly to CM, this measure only considers the neighbours and not the distance to them, which may lead to better results, as demonstrated in [14].

Borderline examples

When considering the 5 nearest neighbours for all samples, a borderline example in a dataset, can be defined as one that has 2 or 3 of its nearest neighbours belonging to the opposite class [43][44]. If the sample has 4 of its nearest neighbours belonging to the opposite class, it is considered as rare. Additionally, if the sample has all of its neighbours belonging to the opposite class, it is considered an outlier. Finally, a Safe sample is one that has only one or no neighbours that belong to the opposite class.

Using this formulation, it is possible to estimate the proportion of the overlapped region, by calculating the amount of borderline examples in the dataset [13]. The following equation shows this idea, where $n_{borderline}$ is the number of borderline examples and n is the total number of samples in the dataset:

$$Overlap = \frac{n_{borderline}}{n} \tag{2.49}$$

This formulation may be applied to the entire dataset or on a per-class basis, similarly to what is done in the complexity measures described above.

IPoints

The IPoints measure takes advantage of the definition of local set (LS) clustering defined previously. In some cases, clusters will only contain one sample. This may indicate that the LS of these cluster cores share their instances with the LS of other clusters with a higher LS cardinality [17]. However, there is also the possibility that the LS of that cluster core only contains the sample itself in its local set, indicating that it is probably surrounded by samples of the opposite class. When this happens, this sample is considered an Invasive Point.

The number of Invasive Points is calculated and then normalized with the total number of samples, n in the dataset:

$$InvasivePoints = \frac{\#InvasivePoints}{n} \tag{2.50}$$

This measure is bounded between 0 and 1 where higher values signify more complex datasets.

2.3.4 Multi Resolution Overlap

This group of measures uses multi resolution techniques to compute the complexity of different sized regions of the dataset. Starting by analysing the domain at a more local level and recursively growing the search space to get a more global picture. This way, a trade-off is obtained between structural overlap and local overlap [13].

MRCA

This measure identifies regions of different complexity in the feature space. Each data sample is assigned a profile space [45] which is used to cluster and subsequently measure complexity.

To generate the profile space of each sample, x, hyperspheres of different radius, σ, and centre in x are drawn. Afterwards, a value is calculated taking into account $N_\sigma^+(x)$ which represents the number of samples of the positive class inside the hypersphere and $N_\sigma^-(x)$ represents the number of samples of the negative class, as shown in Equation 2.51, where y(x) is equal to 1 if x belongs to the positive class and -1 otherwise.

$$\psi_D(x,\sigma) = y(x)\frac{N_\sigma^+(x) - N_\sigma^-(x)}{N_\sigma^+(x) + N_\sigma^-(x)} \tag{2.51}$$

ψ can range from -1 to 1 where values closer to -1 indicate that there is a strong imbalance inside the hypersphere, where most samples have a label different from x. On the other hand, values close to 1 also indicate a high imbalance inside the hypersphere, but with class labels of the same class as x. Finally, values close to 0 indicate that the number of positive and negative samples is balanced inside the hypersphere.

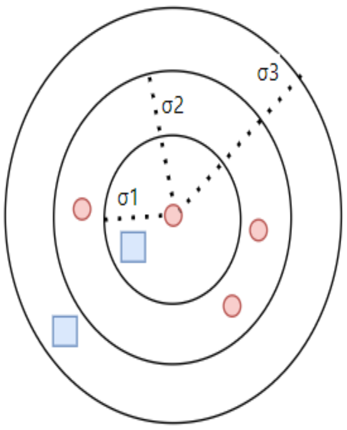


Figure 2.13: Example of a profile calculation using 3 values of σ. For $\sigma1$ there is one sample of the opposite class therefore $\psi(x,\sigma1) = -1$. For $\sigma2$ there is one sample of the opposite class, plus 3 samples of the same class therefore $\psi(x,\sigma2) = 0.75$. Finally, for $\sigma3$ there is an additional sample of the opposite class so $\psi(x,\sigma3) = 0.6$. This process is repeated for each sample and clustering is performed according to the profiles generated.

Repeating this process for different values of σ the profile for a sample x is defined as follows:

$$p = [\psi(x,\sigma_1), \psi(x,\sigma_2), ..., \psi(x,\sigma_m)] \tag{2.52}$$

The set of all profiles p defines the profile space P.

After setting up the profiles for each sample, a centroid based clustering method like k-means is applied to P. Then, for each cluster, $\Delta^{(k)}$, its complexity can be defined as the average of the Multi Resolution Index (MRI) of each pattern, p, contained in it:

$$MRI(p) = \frac{1}{2m}\sum_{j=1}^{m} w_j * [1 - p_j] \tag{2.53}$$

$$w_j = 1 - \frac{j-1}{m} \tag{2.54}$$

$$MRI^{(k)} = \frac{1}{|\Delta^{(k)}|} \sum_{p \in \Delta^k} MRI(p) \tag{2.55}$$

Lower values for $MRI^{(k)}$ correspond to clusters characterized by a high imbalance that agrees with the labelling of the hypersphere centres, i.e. $\psi = 1$. On the other hand, the worst case happens when $MRI^{(k)}$ is closer to 1, indicating high imbalance, but the labels are different from the cluster elements, $\psi = -1$. In cases where the cluster is balanced, $MRI^{(k)}$ approaches the value 0.5.

Results in [45] show that MRI is successful in sorting the clusters in decreasing order of complexity, achieving the best performance by using 3 clusters which split the dataset into high, medium and low complexity clusters.

The calculation for ψ is only valid for binary classification problems, and as such this metric is invalid for non-binary datasets. Adaptations using One Vs One or One Vs All can be made similarly to other metrics described previously.

C1 and C2

Similarly to measures described before, The C1 measure also looks at the nearest neighbours of every sample in the dataset. However, instead of looking for a set number of neighbours like kDN and CM, it iterates over multiple values of k and in each iteration computes the ratio of neighbours that belong to the same class, $P_k(x)$. Afterwards, an average is calculated for all values of k. The average is subtracted to 1 so that high values indicate higher complexity [32]. An example of C1 being calculated for a sample can be seen in 2.14.

$$Complexity(x_i) = 1 - \frac{\sum_{k=1}^{K} P_k(x)}{K} \tag{2.56}$$

The overall complexity can be calculated by averaging the complexity value obtained for each sample as follows, where n is the total number of samples in the dataset:

$$C1 = \frac{\sum_{i=1}^{n} Complexity(x_i)}{n} \tag{2.57}$$

One potential downside of this measure is that it does not take into account the distance to the neighbours in the calculation of $P_k(x)$. So a neighbour that is very close to x will have the same contribution as a neighbour that is very far away [32].

The C2 measure introduced in [32] aims to correct this potential downside by using the distance to each neighbour in the calculation of its value, by weighting each neighbour with the similarity to x.

As such, $P_k(x)$ is calculated as the mean similarity to x considering the neighbours of the same class as x.

Values of C1 and C2 are bounded between 0 and 1, where high values indicate higher complexity. In the case of C2, high values of $P_k(x)$ not only indicate that x has many neighbours of the same class, but also that those neighbours are close to x.

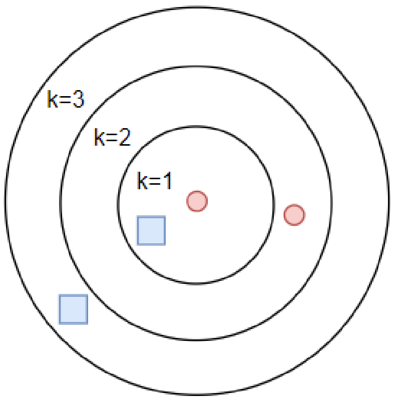


Figure 2.14: Example of C1 for one sample and k=3. For k=1 the nearest neighbour is from the opposite class so $P_1(x) = 0$. For k=2 there is an neighbour from each class so $P_2(x) = 0.5$. Finally, for k=3 there are two neighbours from the opposite class and one neighbour from the same class so $P_2(x) = 0.33$. The C1 value for this sample is $1 - \frac{0.83}{3}$

Purity

This measure recursively partitions feature space into hypercubes at different resolutions [46][47], each resolution corresponds to the number of hypercubes in the feature space. With the increase of the resolutions, more hypercubes cover the feature space and there are fewer samples inside them.

In [46] the data space is divided from resolution 0, where there is no partitioning, to 32, where each feature axis is divided 32 times. An example of this partitioning can be seen in figure 2.15.

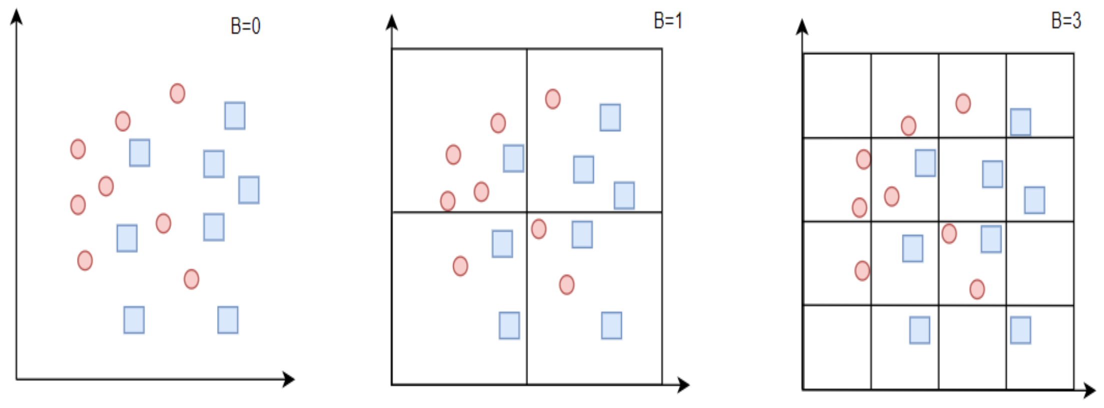


Figure 2.15: Example of partitioning the feature space used for the calculation of purity for 3 resolutions. At resolution B=0 no partitioning is done. At B=1 4 cells are generated and the purity for each of these cells is calculated. Finally, at B=3, there are 16 cells and the purity of each one is calculated. As the resolution increases the value of purity of each cell should increase as well

At each resolution, the purity of each cube is calculated according to the following formulation:

$$S_{H_t} = \sqrt{\frac{K_l}{K_l - 1} \sum_{i=1}^{K_l} (p_{il} - \frac{1}{K_l})^2} \tag{2.58}$$

$$p_{il} = \frac{x_{il}}{\sum_{i=1}^{K_l} x_{il}} \tag{2.59}$$

where K_l is the number of classes, H_l is the number of hypercubes, and x_{il} is the number of samples of class i in the cube H_l.

The overall purity measure is calculated with a weighted sum of the purity of all hypercubes:

$$S_H = \sum_{l=1}^{H} S_{H_l} * \frac{n_l}{n} \tag{2.60}$$

where n is the total number of samples and n_l is the number of samples in the hypercube l.

The value of S_H is calculated for every resolution, r, and multiplied by a weight factor of $\frac{1}{2^r}$. Afterwards, the AUC curve is calculated considering the curve S_H vs normalized resolution (the resolution scaled between 0 and 1). The value is bounded between 0 and 1 where higher values represent less overlapped problems and vice versa. It is expected that the purity will increase as the resolution increases, as each hypercube will be of smaller size.

One downside of this measure is that if in a given hypercube, two clusters of different classes with the same number of samples are present, then this measure will be 0 even though the clusters might be completely separable.

Neighbourhood Separability

This measure is similar to Purity in the sense that it also splits the feature space into hypercubes according to a resolution parameter. However, this measure is more sensitive to the shape of the decision boundaries [46] and does not suffer from the same issue mentioned above.

In each hypercube H_l, for each sample x_j its k nearest neighbours are found and the proportion of the neighbours that belong to the same class is calculated, p_{kj}. This is repeated for several values of k, from 1 to the maximum number of samples of the same class as x_j in that hypercube. Afterwards it is possible to plot, for every sample, a curve of p_{kj} vs normalized values of k and calculate the area under that curve, ϕ_j. Thus, the average neighbourhood separability of each hypercube is calculated as:

$$p_l = \frac{1}{n_l} \sum_{j=1}^{n_l} \phi_j \tag{2.61}$$

Then, similarly to purity, the overall value of the measure for a resolution, r, is calculated taking the weighted sum of all hypercubes, which then weighted by $\frac{1}{2^r}$.

$$S_{NN} = \sum_{l=1}^{H} p_l * \frac{n_l}{n} \tag{2.62}$$

Finally, the AUC is calculated for the plot of S_{NN} vs the normalized resolution. The values are bounded between 0 and 1, where values closer to 1 represent more overlapped domains.

2.4 Conclusions

While multiple works have been made on the topic of dataset shift, few tackle the issue with the context of imbalance in mind [18]. As pointed out in [10] there may be several external factors that affect the performance of a classification algorithm, other than dataset shift, and it would be beneficial to study several factors simultaneously to understand their impact.

The few studies that research both dataset shift and imbalance try to reduce the amount of dataset shift by using cross validation algorithms. Usually these algorithms are optimizations made to SCV that further reduce the amount of shift. As for imbalance the majority of studies confront the idea that when imbalance is high the effects of dataset shift are more notable, massively degrading the classifiers' performance. Despite coming to this conclusion they fail to test strategies such as the use of preprocessing algorithms to reduce the IR on the training partition to see if it is possible to combat the joint effects of imbalance and dataset shift.

In the context of overlap, a taxonomy was presented showing different families of overlap and measures belonging to each family. It is clear that overlap can have multiple facets, however is it unclear if the taxonomy proposed truly captures these facets. Furthermore, alongside finding a set of measures that capture these dimensions of overlap, it is important to explore methods that reduce overlap in each of these dimensions. Despite this, there is a lack of studies attempting to propose a taxonomy for measuring overlap or validating existing ones. Moreover, there is also a lack of studies that study algorithms that attempt to reduce overlap, especially in imbalanced contexts.

This page is intentionally left blank.

Chapter 3

Experimental Setup and Results

In this section the experimental setup and results of the experiments performed for overlap and dataset shift are shown. The section is divided into three main subsections, one for dataset shift and the remaining two for overlap. The subsection about dataset shift starts with the details from experimental setup, followed by the results of the experiments done to answer the first and second research questions. The first subsection on overlap, focuses on the validation of the *pycol* package. Finally, the second section on overlap follows a similar structure to the previous section about dataset shift.

3.1 Dataset Shift

This section starts by detailing the setup for the experiments performed on the problem of dataset shift, which goes into detail about the algorithms and datasets used as well as the justification behind these choices. This section then follows with the results and discussion where the first two research questions are answered.

3.1.1 Experimental Setup

The experimental setup was divided into two phases as shown in Figure 3.1: (i) Partitioning and Oversampling; (ii) Complexity and Performance measures extraction. Each of the phases will be explained in more detail hereafter.

The 83 real datasets used were collected from the KEEL repository, more specifically the datasets were taken from the Imbalanced Datasets For Classification section. The main characteristics of the datasets are summarized in Table 3.1.

As the table shows the datasets vary in imbalance ratio (IR), while some have relatively lower values like the bupa dataset with 1.38, the more extreme cases have values above 20. The large range and variety of IRs should help to understand the combined effects of imbalance and dataset shift. Furthermore, these datasets are all binary and range from 3 to 20 features, which are in all cases continuous.

In the first phase, the datasets are partitioned into 5 folds using the four cross validation methods described previously: Stratified Cross Validation (SCV), Distributed Balanced Stratified Cross Validation (DBSCV), Distributed Optimized Balanced Stratified Cross Validation (DOBSCV) and Maximally Shifted Stratified Cross Validation (MSSCV).

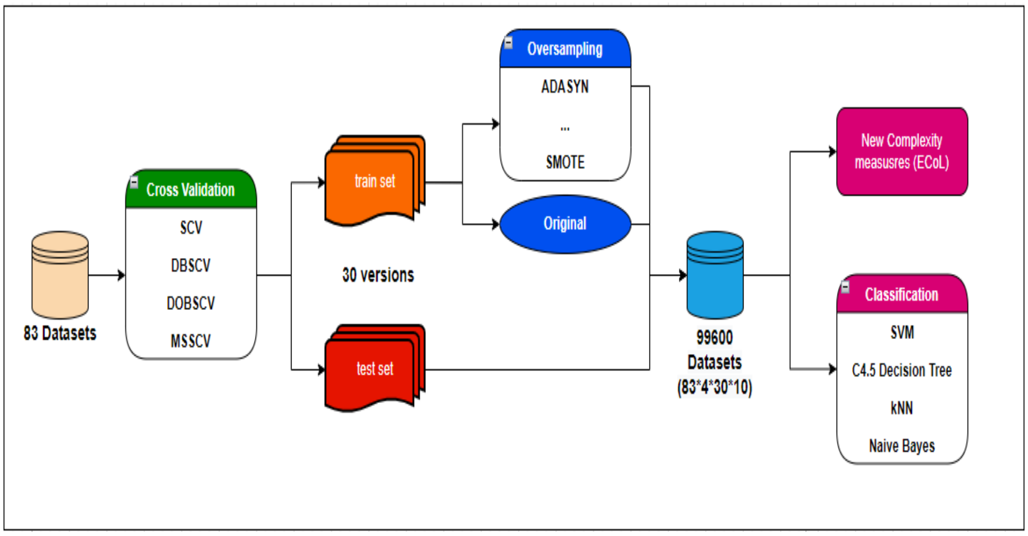


Figure 3.1: Experimental setup for the dataset shift experiments

Datasets	Size	Features	IR
bupa	345	6	1.38
pageblocks-1-3vs4	472	10	1.57
glass1	214	9	1.82
ecoli-0vs1	220	7	1.86
wisconsin	683	9	1.86
pima	768	8	1.87
cmc1vs2	961	9	1.89
iris0	150	4	2.00
glass0	214	9	2.06
german	1000	20	2.33
yeast1	1484	8	2.46
haberman	306	3	2.78
vehicle2	846	18	2.88
vehicle1	846	18	2.90
vehicle3	846	18	2.99
glass-0-1-2-3vs4-5-6	214	9	3.20
transfusion	748	4	3.20
vehicle0	846	18	3.25
ecoli1	336	7	3.36
newthyroid1	215	5	5.14
ecoli2	336	7	5.46
balance-scaleBvsR	337	4	5.88
balance-scaleBvsL	337	4	5.88
segment0	2308	19	6.02
glass6	214	9	6.38
yeast3	1484	8	8.10
ecoli3	336	7	8.60
pageblocks0	5472	10	8.79
ecoli-0-3-4vs5	200	7	9.00
yeast-2vs4	514	8	9.08
ecoli-0-6-7vs3-5	222	7	9.09
ecoli-0-2-3-4vs5	202	7	9.10
glass-0-1-5vs2	172	9	9.12
yeast-0-3-5-9vs7-8	506	8	9.12
yeast-0-2-5-6vs3-7-8-9	1004	8	9.14
yeast-0-2-5-7-9vs3-6-8	1004	8	9.14
ecoli-0-4-6vs5	203	7	9.15
ecoli-0-1vs2-3-5	244	7	9.17
ecoli-0-2-6-7vs3-5	224	7	9.18
glass-0-4vs5	92	9	9.22
ecoli-0-3-4-6vs5	205	7	9.25
ecoli-0-3-4-7vs5-6	257	7	9.28
yeast-0-5-6-7-9vs4	528	8	9.35
vowel0	988	13	9.98
ecoli-0-6-7vs5	220	7	10.00
glass-0-1-6vs2	192	9	10.29
ecoli-0-1-4-7vs2-3-5-6	336	7	10.59
ecoli-0-1vs5	240	7	11.00
glass-0-6vs5	108	9	11.00
glass-0-1-4-6vs2	205	9	11.06
glass2	214	9	11.59
ecoli-0-1-4-7vs5-6	332	7	12.28
cleveland-0vs4	173	14	12.31
ecoli-0-1-4-6vs5	280	7	13.00
shuttle-c0-vs-c4	1829	9	13.87
yeast-1vs7	459	8	14.30
glass4	214	9	15.46
ecoli4	336	7	15.8
abalone9-18	731	8	16.4
dermatology-6	358	34	16.9
thyroid-3vs2	703	21	18.00
glass-0-1-6vs5	184	9	19.44
pageblocks-1vs3-4-5	5144	10	21.27
shuttle-6vs2-3	230	9	22.00
yeast-1-4-5-8vs7	693	8	22.10
pageblocks-1-2vs3-4-5	5473	10	22.69
glass5	214	9	22.78
yeast-2vs8	482	8	23.10
flare-F	1066	11	23.79
car-good	1728	6	24.04
pageblocks-1vs4-5	5116	10	24.20
car-vgood	1728	6	25.58
kr-vs-k-zero-onevsdraw	2901	6	26.63
yeast4	1484	8	28.10
winequality-red-4	1599	11	29.17
poker-9vs7	244	10	29.50
yeast-1-2-8-9vs7	947	8	30.57
abalone-3vs11	502	8	32.47
yeast5	1484	8	32.73
kr-vs-k-threevseleven	2935	6	35.23
winequality-red-8vs6	656	11	35.44
abalone-17vs7-8-9-10	2338	8	39.31
abalone-21vs8	581	8	40.50

Table 3.1: Datasets used for the Dataset Shift Experiments

The partitioning phase is particularly relevant to answer the first research question posed, regarding the use of cross validation algorithms to reduce dataset shift, also comparing the results obtained with the ones in [12] where the same algorithms were used. Due to the random component of these methods, for each one, the cross-validation was performed 30 times, creating 30 versions for each pair of dataset/CV method.

The following oversampling algorithms, implemented in the KEEL framework, are then applied to all versions of the datasets: ROS, SMOTE, Safe-Level SMOTE, Borderline-SMOTE, ADASYN, AHC, ADOMS, SMOTE-TL and SMOTE-ENN.

The oversampling phase is important to answer RQ2 regarding the interaction between dataset shift and imbalance. To that end experiments were made analysing the differences between the performance of classifiers on the original and oversampled datasets. Furthermore, another set of experiments wese also conducted analysing these classifiers with respect to the imbalance ratio of the original datasets. The idea behind oversampling the datasets is to test if increasing the representativity of the minority class would increase the performance despite the existence of shift in the data.

By the end of this phase there exists 99600 datasets: the original 83 x 4 cross validation methods x30 versions x9 oversampling algorithms and the originals.

For the second phase the following complexity measures F1, F1v, F2, F3, F4, N1, N2, N3, N4, T1, T2 [8] were calculated from the 99600 datasets by using the ECol tool. Using these complexity measures will help to justify the results obtained, particularly when comparing the performance obtained with the different CV algorithms.

Finally, all the datasets were classified with the following set of algorithms: SVM, C4.5 Decision Tree, kNN and Naive Bayes, implemented in WEKA. The following measures were then extracted from the test datasets: Precision, Recall, F-Measure and AUC. These 4 classifiers were picked to test with a variety of classification paradigms, the existence of dataset shift and imbalance could show different behaviours depending on the classification paradigm used.

3.1.2 Results and Discussion

RQ1: Is DOBSCV a cross validation algorithm that reduces more dataset shift in comparison to DBSCV, SCV and MSSCV?

This section presents the experiments made to answer RQ1, starting by showing box plots of the cross validation algorithms versus some performance metrics taken. Statistical tests are also conducted to complement the initial analysis. Finally, the results of the complexity measures are also shown.

The initial set of experiments started by analysing the C4.5 and kNN classifiers, to test if the experiments performed in [12] maintained the same results. Figures 3.2-3.3 show the graph of the performance vs cross validation method, for the F measure and AUC measures obtained in the test datasets.

These results show a similar outcome to [12], where datasets partitioned with DBSCV and DOBSCV provide the best results out of the four cross validation algorithms used. When analysing both figures we can see that out of the four algorithms these two distinguish themselves slightly from SCV while MSSCV provides the worst results overall. However, the performance of the DOBSCV did not outperform (they seem to be somewhat equal) DVSCV which contradicts the results of [12],

These results were similar for the other remaining classifiers (full results can be found in the appendix A in Figure A.3 and Figure A.4).

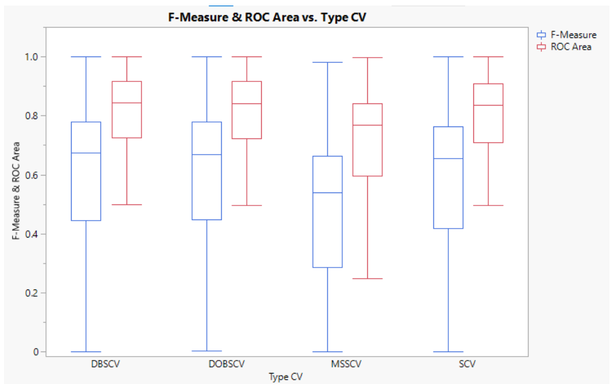


Figure 3.2: Method of Cross Validation vs Performance for the C4.5 decision tree

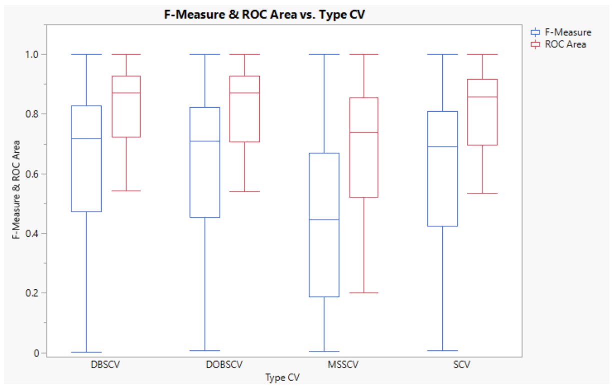


Figure 3.3: Method of Cross Validation vs Performance for the kNN

Before presenting the results of the statistical tests, Table 3.2 shows the ranking performance metrics for the C4.5 Decision Tree, grouped by Oversampling Algorithm and Cross Validation method (similarly, the full results can be consulted in appendix A in tables A.1-A.4).

The table shows that among the 4 algorithms DBSCV or DOBSCV always perform both at the top for every metric and every oversampling algorithm either ranking 1º or 2º. The most common scenario is that DBSCV performs 1º for most metrics, but there are some cases. where DBSCV and DOBSCV tie as both perform 1º on two metrics and 2º on the remaining two. Consulting Table A.4 in appendix A with the averages, results show that DBSCV slightly outperformed DOBSCV in most cases. It is also possible to notice that the use of an oversampling algorithm greatly increases the performance of the classifier, about 12% compared to the original datasets used. Finally, SCV and MSSCV consistently performed 3º and 4º respectively, which is consistent with the results in [12].

A non-parametric Friedman test for each of the oversampling algorithms was conducted for a significance level of $p < 0.05$, for the F Measure values, to assess statistical differences between the CV algorithms. Afterwards post-hoc tests were conducted for the oversampling algorithms when statistical differences were found. The results of these statistical tests can be found in the appendix A (Tables A.5 - A.15), and they confirm the same ideas observed in the exploratory analysis done previously.

For all the oversampling algorithm analysed, clear differences are found between every CV algorithm, except for the comparison between DBSCV and DOBSCV. These two algorithms are shown to not be statistically different, with the lowest p-value being 0.3403 in the original datasets. This is illustrated in Table 3.2 where the 1º and 2º place is usually

Algorithm	CV	F-Measure	Precision	Recall	AUC	Avg
ADASYN-I	**DBSCV**	**1**	**1**	**1**	**1**	**1**
	DOBSCV	2	2	2	2	2
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
ADOMS-I	**DBSCV**	**2**	**2**	**1**	**1**	**1.5**
	DOBSCV	**1**	**1**	**2**	**2**	**1.5**
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
AHC-I	**DBSCV**	**2**	**1**	**1**	**1**	**1.25**
	DOBSCV	1	2	2	2	1.75
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
Borderline_SMOTE-I	**DBSCV**	**2**	**2**	**1**	**1**	**1.5**
	DOBSCV	**1**	**1**	**2**	**2**	**1.5**
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
Original	**DBSCV**	**1**	**1**	**1**	**1**	**1**
	DOBSCV	2	2	2	2	2
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
ROS-I	**DBSCV**	**1**	**1**	**1**	**1**	**1**
	DOBSCV	2	2	2	2	2
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
SMOTE-I	**DBSCV**	**1**	**2**	**1**	**1**	**1.25**
	DOBSCV	2	1	2	2	1.75
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
SMOTE_ENN-I	**DBSCV**	**2**	**2**	**1**	**1**	**1.5**
	DOBSCV	**1**	**1**	**2**	**2**	**1.5**
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
SMOTE_TL-I	**DBSCV**	**1**	**2**	**1**	**1**	**1.25**
	DOBSCV	2	1	2	2	1.75
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3
Safe-Level_SMOTE-I	**DBSCV**	**1**	**2**	**1**	**1**	**1.25**
	DOBSCV	2	1	2	2	1.75
	MSSCV	4	4	4	4	4
	SCV	3	3	3	3	3

Table 3.2: Ranking for the Performance Measures for C4.5 Decision tree for all datasets, grouped by oversampling algorithm and cross validation method. Highlighted are the best performing algorithms according to the average ranking of the 4 performance metrics

occupied by these two algorithms, with none occupying the first position consistently.

If we assume, the amount of dataset shift in the data is tied to the performance of the classification algorithm, then two possible explanations can be found:

- (i) DOBSCV does not reduce the amount of dataset shift in comparison to DBSCV, and therefore it does not outperform it, like what was expected;
- (ii) DOBSCV does reduce the amount of dataset shift, but not significantly enough to show statistical differences.

Of the two options, the results lead more towards the first explanations, since when looking at the exploratory analysis, it appears that datasets partitioned with DOBSCV are responsible for giving slightly lower performance measures.

In what concerns to data complexity, the results show the absolute differences between the complexity measures of the training and testing sets, grouped by the cross validation method. If the cross validation method is attempting to combat dataset shift, then, the difference between the complexity of the training and testing sets should be lower than for datasets split with a cross validation algorithm that does not take this factor into account, since both the training and testing partitions should have equal representation. Due to the lack of studies linking overlap and dataset shift it is unclear how the different families behave exactly, which is something this experiment also aims to make clearer. We are particularly interested in analysing the values obtained by DBSCV and DOBSCV which showed previously to be comparable. If the two algorithms are similar as the previous experiments have indicated then the values of the complexity measures found should be similar as they would be splitting the datasets the same way.

The results for this experiment can be seen in Table 3.3, the measures are grouped into families according to the taxonomy defined in the state of the art. T2 does not belong to any of these families, so it is analysed separately.

	F1	**F1v**	**F2**	**F3**	**F4**	**N3**	**N4**	**N1**	**N2**	**T1**	**T2**
DBSCV	0.008	0.0236	0.0379	0.2001	0.1893	0.0431	0.0062	0.0496	0.0541	0.0382	0.0914
DOBSCV	0.0098	0.0269	0.038	0.2067	0.1963	0.0351	0.0063	0.0451	0.0451	0.0406	0.0914
MSSCV	0.1013	0.0889	0.0606	0.3614	0.2761	0.0341	0.0283	0.0448	0.0449	0.1112	0.0914
SCV	0.0139	0.0363	0.0417	0.223	0.2038	0.0156	0.0083	0.0295	0.0256	0.0458	0.0914

Table 3.3: Difference between the complexity measures of the training and test partition by cross validation method of the original datasets

The results show that for algorithms like MSSCV, designed for inducing dataset shift, the differences between the training and testing set are much larger in comparison to the remaining algorithms, noticeable especially in the case of the feature level metrics F1, F1v, F3, F4. As for the remaining families there are minor differences for MSSCV shown in the case of N4 in the instance level family and some more significant differences in the case of T1 in the structural family.

Overall, it appears that no matter the type of overlap, the CV algorithm that most stands out is always MSSCV, having always at least one metric per family with significant differences. Furthermore, for an algorithm like SCV which only take into account prior probability shift, the differences between partitions are slightly higher than for the remaining two algorithms, especially for the measures mentioned above, which corresponds to what is seen in the previous studies.

More importantly, the results show that both DBSCV and DOBSCV obtain very similar values for most complexity measures observed, confirming the idea that the optimization done to DOBSCV is not enough to reduce dataset shift as the difference between the training and testing partition is almost the same regardless of which algorithm was used to split them.

It is also interesting to note the T2 values obtained. This metric indicates the average number of samples per dimension, and it was the same for all cross validation algorithm. This is because all CV algorithms distribute the same number of samples per fold, so the average number of samples per dimension will the same, regardless of the CV algorithm used. Because of this, the T2 metric proves to be not very useful when trying to quantify the amount of dataset shift.

On the other hand the remaining measures are helpful at identifying covariate shift, but not prior probability, since these metrics focus only on the distribution of the samples on the feature space and not on the proportion of samples in each class. As far as the best set of metrics to measure covariate shift, it appears that the feature overlap metrics along with T1 show the greatest differences when the amount of shift increases as pointed out previously. These metrics are very sensitive to noisy samples than the remaining ones which might explain why the higher differences are found.

In conclusion the main takeaways from the experiments using complexity measures are:

- (i) MSSCV presents the highest differences for every family of measures among all algorithms, this is consistent with the results presented in the literature;
- (ii) SCV which only takes into account prior probability shift making the differences between partitions higher than DBSCV and DOBSCV;
- (iii) DBSCV and DOBSCV are very close in values for most of the measures studied, which further confirms the idea that the two algorithms induce the same amount dataset shift;
- (iv) The feature metrics along with T1 seem to be the best metrics to detect dataset shift, probably due to the fact that they are very sensitive to the existence of noisy samples.

RQ2: What is the impact that imbalance and dataset shift have on the performance of classifiers. Can oversampling algorithms increase their performance?

This section details the results obtained for the experiments made in an attempt to answer RQ2 about the interaction between dataset shift and imbalance. Similarly to the previous section, the experiments start by showing some box plots of the oversampling algorithms versus some performance metrics. These results are then validated using statistical tests. Finally, results are also shown for the experiments comparing performance versus the imbalance ratio of datasets.

First, an experiment for the kNN classifier was conducted to measure the differences between oversampled datasets and the original one (Figure 3.4) with the goal of understanding if the determining factor on the performance was the oversampling algorithm used or the cross validation algorithm. In theory using an oversampling algorithm should reduce the imbalance ratio of the training dataset and make it, so the minority class is more visible during training process which would boost the performance of classifiers. On the other hand because the process is done after the dataset is split into training and testing set it will do nothing to combat dataset shift, so if a concept that is not initially present in training, but it exists in testing, then oversampling will hardly make the two sets more equal in order to reduce dataset shift.

In the previous section it was already established that the choice of a CV algorithm does influence the performance of a classification algorithm, if the use of an oversampling algorithm does matter then noticeable differences will also be found between the original and oversampled datasets.

The results show that the use of an oversampling algorithm does not necessarily increase the performance of the classifier, for example the case of SMOTE-TL and SMOTE-ENN which give slightly worse results as seen in Figure 3.4.

The experiments performed may lead to the conclusion that the performance of a classifier is more influenced by the amount of dataset shift than the amount of imbalance presented in the data.

To confirm this idea, we can observe the results of another classifier, the C4.5 decision tree, already studied previous section (Figure 3.5).

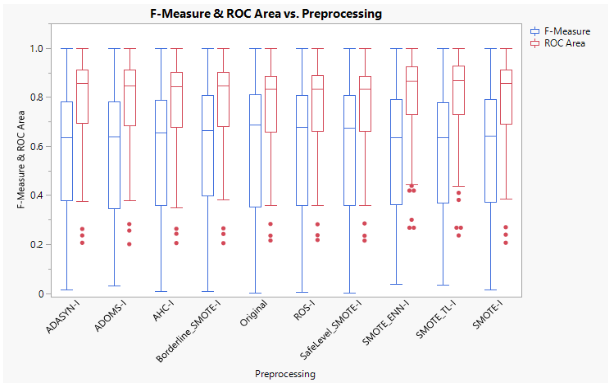


Figure 3.4: Oversampling Algorithm vs Performance for the kNN classifier

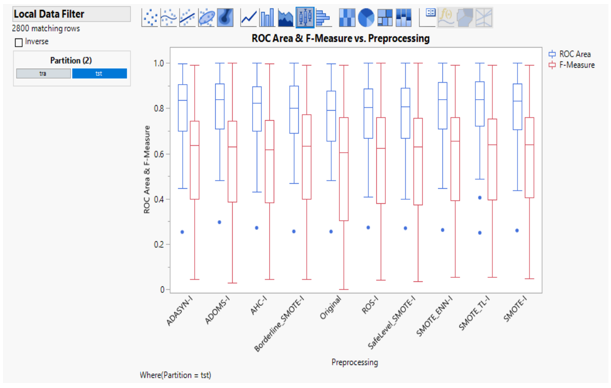


Figure 3.5: Oversampling Algorithm vs Performance for the C4.5 Decision Tree

The results for this classifier contradict the initial idea that the use of an oversampling algorithm does not affect the performance of the classifier, while the difference is not as noticeable as with the choice of a CV algorithm, there appears to be a slight difference between the original and oversampled datasets, especially in the F measure, where the box plot of the original datasets shows as noticeable lower average.

Similar experiments for the remaining classifiers were conducted and can be found in the appendix A (Figure A.1 and Figure A.2). Overall the results are similar in the sense that depending on the classifier, the degree of imbalance can have an impact on performance, however, the amount of dataset shift always impacts the performance of the classifiers.

Furthermore, statistical tests showing the differences between the oversampled and original datasets were also conducted. For that, all the oversampled datasets were grouped up into a single category and a Wilcoxon test was conducted, at a $p < 0.05$ significance level, comparing them to the original. Table 3.4 show the test results for all the classification algorithms. The expected results should match the ones observed in the exploratory analysis where for global algorithms like the C4.5 decision tree, the difference between the oversampled and original datasets will be significative. While for local algorithms such as kNN no difference between approaches should be found.

The results show that the difference between the using oversampled datasets is dependent on the classification algorithm in question, which is somewhat expected. In particular, the SVM has clear differences no matter the CV algorithm, with MSSCV being slightly lower. The C4.5 algorithm presents more interesting results, where all the p-values are around the 0.05 level of significance, except for the MSSCV algorithm which is much lower, at a p-value of 0.01. As for the other 2 classification algorithms, there is no CV method

Classifier	Algorithm	Z Value	p-value
SVM	DBSCV	551.0	0.0001
	DOBSCV	539.0	0.0001
	MSSCV	339.0	0.0
	SCV	531.0	0.0001
kNN	DBSCV	1203.0	0.9785
	DOBSCV	1192.0	0.9262
	MSSCV	983.0	0.1795
	SCV	1177.0	0.8553
NB	DBSCV	1055.0	0.3619
	DOBSCV	1060.0	0.3778
	MSSCV	1087.0	0.3628
	SCV	1089.0	0.369
C4.5	DBSCV	923.0	0.0615
	DOBSCV	907.0	0.0496
	MSSCV	826.0	0.0148
	SCV	906.0	0.0489

Table 3.4: Results for the classifiers

where statistical differences were found. Although it is interesting to notice that the lower p-values are always found for the MSSCV algorithm.

In conclusion the observed results showed that:

- (i) Increasing the amount of shift always significantly degraded the performance of the classifier, the amount of imbalance only affect the performance of certain classifiers;
- (ii) Global classifiers like C4.5 and SVM are more affected by imbalance than local ones like kNN, with the exception to the rule being Naive Bayes which is resilient to imbalance despite its global approach. However, this is consistent with the literature, where this classifier has showed some resilience to this factor.

To further explore this topic, some experiments were conducted with MSSCV and DBSCV, analysing the original datasets and their imbalance ratio. In theory, as the imbalance ratio increases, the difference in performance between the training and testing partitions should increase. Furthermore, for algorithms like MSSCV, large difference values should be seen for lower imbalance ratios when compared to DBSCV. The results of this experiment can be seen in Figures 3.6 and 3.7. The theorized outcome matches what is shown by the two figures mentioned. Particularly while DBSCV does present some high differences, even for lower imbalance ratios below 5, for these same imbalance values, MSSCV has a much higher concentration of values with F-Measure differences above 0.8 as demonstrated by the spread of points in each figure. As the imbalance ratio value increases, the differences between CV algorithms become less significant, with both algorithms following a similar trend of increasing the F-measure differences between partitions. It should be noted that even for the higher imbalance ratio values, MSSCV almost always seems to have higher values.

The main conclusions achieved with this experiment analysing the imbalance ratios were:

- (i) As literature suggests, between dataset shift and imbalance the determining factor that most degrades an algorithm's performance is dataset shift, however when both issues are combined the consequences are much more evident;

- (ii) If a bad partitioning algorithm which induces high dataset shift is used the difference in performance between partitions can be high even for low IR;
- (iii) For very high IRs (30-40) the choice of CV algorithm stops being as important. This is because the classification algorithm will perform poorly regardless of the partitioning method. But a method that induces less shift is still a better choice overall.

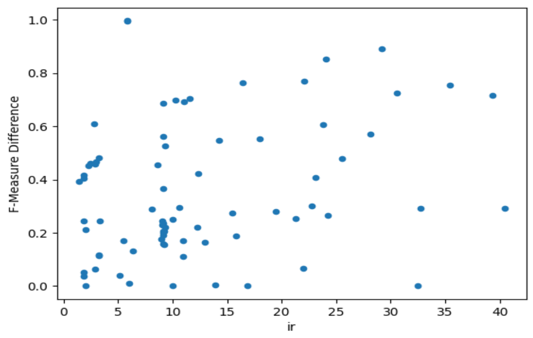


Figure 3.6: Imbalance Ratio vs Difference in performance for DBSCV

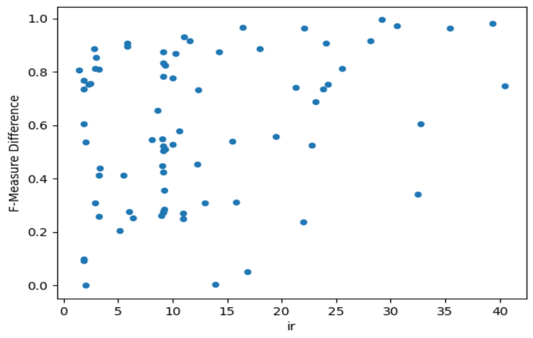


Figure 3.7: Imbalance Ratio vs Difference in performance for MSSCV

3.2 Overlap - pycol Package

With the goal of making it easier to experiment with most state-of-the-art complexity measures, a new Python package was created, *pycol* [1], which implemented the 28 measures described in the previous Complexity Measures section, along with other measures like L1, L2 and L3 [16] which are not found in the taxonomy.

3.2.1 Implementation Details

This package was created using *Pymfe* as its base, starting by implementing all the measures it already had. Afterwards, the remaining measures were implemented from scratch using the formulation provided in their original research article, adding 17 new complexity measures.

The per class versions of some metrics listed previously, like F1 and N1 for example, were also be implemented, similarly to the ImbCol package. Also, for measures that were originally thought only for binary contexts, a *one vs one* and *one vs all* approach can be selected via function parameters.

Distance based complexity measures, like kDN and D3 for example, used the HEOM [48] distance function so that datasets with categorical attributes can also be evaluated. This approach is not valid for measures like F1 which calculate means and standard deviation of each feature, so currently these measures are not usable for datasets with these types of features.

As for the dataset file format, currently the *pycol* package uses the *arff* format, however in the future other formats can be considered to make the package easier to use.

3.2.2 Validation

The validation of the implemented measures was divided into two groups. The first group was used for the measures already implemented in *pymfe*, these measures were compared to the results given by the ones implemented in *pycol* package. The remaining measures without an available implementation were tested in artificial datasets. Table 3.5 shows which measures are implemented in both *pycol*, *pymfe*, DCoL and ECoL.

The artificial datasets were created using the data generator introduced in [49]. With this generator, it is possible to create data clusters of variable size and topology. The generator divides the creation of samples in multiple regions, the number of regions, their size, shape and location can all be configured by the user. For each type of shape available, there is an algorithm that uniformly fills the area inside with safe and borderline samples. Afterwards, the area around the region is populated with rare and outlier examples.

Results for the first group of measures were compared in 4 of the KEEL repository datasets. The characteristics of these datasets are shown in Table 3.6 and the comparison is presented in Table 3.7. For non-binary datasets, the OvO results are summarized using a mean.

All measures except F1 and N2 give the exact same result in both packages for every dataset, indicating the implementation is indeed valid for at least those. As for F1, the difference in results is due to a slight change in the implementation where the means of each feature is not normalized, justifying variations between both approaches. Finally, for

[1]https://github.com/DiogoApostolo/pycol

Measure	pycol	pymfe	DCoL	ECoL
F1	Yes	Yes	Yes	Yes
F1v	Yes	Yes	Yes	Yes
F2	Yes	Yes	Yes	Yes
F3	Yes	Yes	Yes	Yes
F4	Yes	Yes	Yes	Yes
N1	Yes	Yes	Yes	Yes
N2	Yes	Yes	Yes	Yes
N3	Yes	Yes	Yes	Yes
N4	Yes	Yes	Yes	Yes
L1	Yes	Yes	Yes	Yes
L2	Yes	Yes	Yes	Yes
L3	Yes	Yes	Yes	Yes
LSC	Yes	Yes	Yes	Yes
T1	Yes	Yes	Yes	Yes
NSG	Yes	No	No	No
ICSV	Yes	No	No	No
ONB	Yes	No	No	No
Clst	Yes	No	No	No
DBC	Yes	No	No	No
R-Value	Yes	No	No	No
degOver	Yes	No	No	No
SI	Yes	No	No	No
kNN	Yes	No	No	No
CM	Yes	No	No	No
D3	Yes	No	No	No
IPoints	Yes	No	No	No
MRCA	Yes	No	No	No
C1	Yes	No	No	No
C2	Yes	No	No	No
Purity	Yes	No	No	No
Sep. Index	Yes	No	No	No

Table 3.5: Data complexity Packages and implemented measures

Name	#Samples	#Features	#Classes
newthyroid	215	5	3
ecoli	335	7	8
balance	625	4	3
titanic	2200	4	2

Table 3.6: KEEL repository dataset characteristics

N2 the differences are also very small between the two packages, which is likely due to the default distance metrics used in each approach, which are slightly different in terms of normalization.

For the second group of measures, three sets of test were made. In the first set of tests, simple artificial datasets were created where it was possible to manually calculate the expected value of each of the metrics, this process is exemplified in Appendix B for the T1, Augmented R-Value and Purity measures.

	newthyroid		ecoli		balance		titanic	
Measure	**pycol**	**pymfe**	**pycol**	**pymfe**	**pycol**	**pymfe**	**pycol**	**pymfe**
F1	**0.5429**	**0.5124**	**N.A**	**0.5677**	**0.8342**	**0.8306**	**0.8370**	**0.9030**
F1v	0.0498	0.0498	0.1240	0.1240	0.2292	0.2292	0.4356	0.4356
F2	0.0005	0.0005	0.000	0.0000	1.000	1.0000	1.000	1.000
F3	0.1349	0.1349	0.9569	0.9569	0.5980	0.5980	1.000	1.000
N1	0.1023	0.1023	0.3035	0.3035	0.2752	0.2752	0.3198	0.3198
N2	**0.2368**	**0.2478**	**0.4160**	**0.3966**	**0.4036**	**0.4231**	**0.0270**	**0**
N3	0.0279	0.0279	0.2083	0.2083	0.2128	0.2128	0.2221	0.2221
N4	0.0093	0.0093	0.1398	0.1398	0.1312	0.1312	0.4329	0.4329
LSC	0.7702	0.7702	0.9741	0.9741	0.9663	0.9663	0.9999	0.9999
T1	0.2279	0.2279	0.7529	0.7529	0.3648	0.3648	0.004	0.004

Table 3.7: Complexity measures results for the KEEL datasets

The second set of tests starts by creating two clusters of different classes with each 750 samples. The overlapped region between these clusters is increased until the two clusters are completely overlapped (Figure 3.8).

Ideally, if the complexity measures are implemented correctly, its values will indicate higher complexity as the overlapped region increases. Results for the second group of metrics in each of the artificial datasets are presented in Table 3.8.

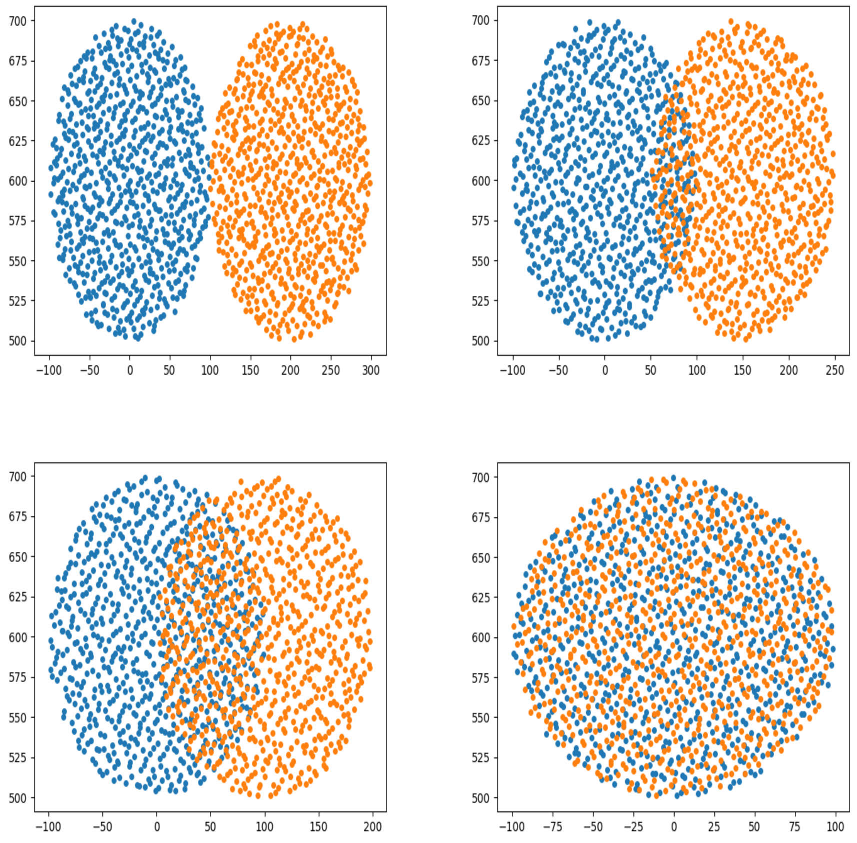


Figure 3.8: Artificial dataset with increasing regions of overlap

Overall, the results show that all the metrics have values according to the expected, as when

Measure	Test 1	Test 2	Test 3	Test 4
R value	0.003	0.114	0.2953	0.7107
D3	[2,3]	[89,82]	[232,211]	[532,534]
CM	0.003	0.114	0.2953	0.7106
kDN	0.0052	0.095733	0.24067	0.584133
DBC	0.0096	0.2181	0.5776	0.8535
SI	0.99667	0.7180	0.5776	0.6773
input noise	0.499	0.5927	0.7410	0.9983
borderline	0.8	14.93	40.73	98.067
deg overlap	0.0147	0.1753	0.4346	0.9993
C1	0.1328	0.2003	0.3037	0.5011
C2	0.1664	0.2267	0.3199	0.5031
Clst	0.004	0.122	0.3366	0.7147
purity	0.0228	0.0247	0.0181	0.0001
neigh. sep.	0.2965	0.26976	0.2237	0.1228

Table 3.8: Complexity Measure Results in the Artificial Datasets

the overlapped region increases, the values tend to increase also. A notable exception to this rule are the SI, purity and neighbourhood separability measures, however this measures work differently from the rest where smaller values indicate higher complexity, so the values presented still indicate that the implementation is valid.

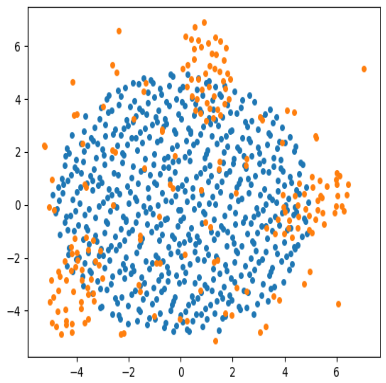
After these metrics were validated with the second set of tests, a third set of more complex artificial datasets was used, which were taken from [49]. Table 3.9 presents the characteristics of the datasets and a 2D view of the datasets is presented in Figure 3.9:

Name	#Samples	#Features	#Classes	Class Ratio
cirles-2d	800	2	2	1:3
spheres-2d	3000	2	2	1:1
paw3-2d	1000	2	2	1:9
paw3-3d	1500	2	3	1:7

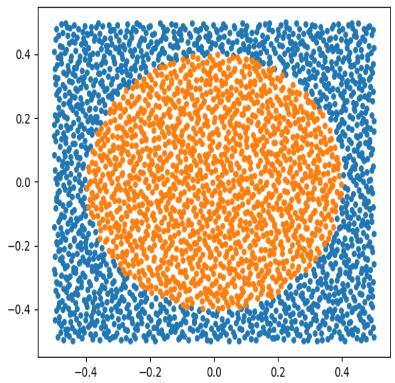
Table 3.9: Second set of artificial dataset characteristics

Overall, this second set of datasets present very clear local regions where little overlap is present, except for *circles-2d* which have a more considerable amount of overlap in each of the three main clusters of the minority class (Table 3.9).

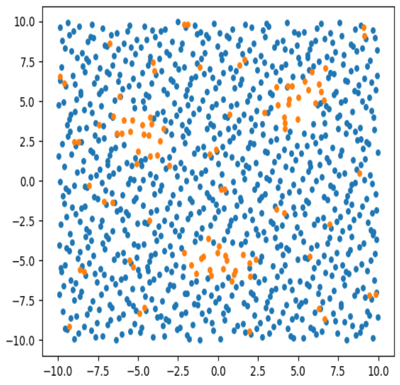
As most of the experimented metrics take into account the local region around each sample, it is expected that the values for the measures will represent lower complexity, except input noise which is a feature based metric and should have high values, since none of the two features is able to linearly separate any of the datasets. The results of these experiments are present in Table 3.10 and are mostly within expectations, as most of the measures are corresponded to low complexity. The measures between 0 and 1 are lower than 0.5, when 1 represents high complexity, and closer to 1 when in the opposite case. Also, as expected, input noise, being a feature based metric gives very high values, representing high complexity. The two measures that got results which defied expectations were purity and neighbourhood separability, which both have similar formulations. However, being multi-resolution metrics it is most likely due to the need for better parametrization which is very dataset dependent.



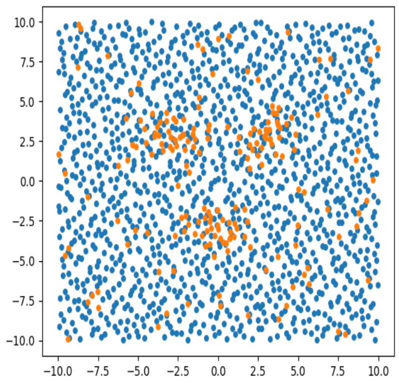
(a) circles-2d



(b) spheres-2d



(c) paw3-2d



(d) paw3-3d

Figure 3.9: Second set of artificial datasets

Measure	**circles-2d**	**spheres-2d**	**paw3-2d**	**paw3-3d**
R value	0.2050	0.01967	0.0819	0.0799
D3	[41,123]	[29,30]	[39,84]	[11,69]
CM	0.205	0.01967	0.082	0.08
kDN	0.2557	0.0334	0.1433	0.1366
DBC	0.3632	0.0538	0.2347	0.2095
SI	0.87625	0.9893	0.8393	0.927
input noise	0.9568	0.7958	0.9886	0.9760
borderline	33.625	5.433	16.7333	16.888
deg overlap	0.5787	0.0923	0.3800	0.3600
C1	0.1864	0.08047	0.0	1.9047
C2	0.3256	0.3682	0.0111	0.1815
Clst	0.2987	0.02467	0.1653	0.1590
purity	0.024	0.008	0.003	0.037
neigh. sep.	0.2214	0.2881	0.0360	0.2568

Table 3.10: Complexity measure results in the second set of artificial datasets

3.3 Overlap - Taxonomy Validation

After the validation of the implemented metrics in *pycol*, the next step consists in the validation of the taxonomy presented in [13] which illustrates the state of the art of the overlap families. Furthermore, another important aspect is to better understand what preprocessing algorithms work best for each type of complexity.

In theory certain types of preprocessing algorithms are equipped to deal with a specific types of overlap. In [13] some proposals are made with algorithms for each type of overlap in the taxonomy. For example, it is suggested that cleaning algorithms proposed in [50] NB-Basic and NB-Tomek are good at dealing with instance overlap since in their cleaning process they exclusively uses local information from the neighbourhood of each sample.

The overall idea is to select a few algorithms that address each type of overlap and look at how the metrics react before and after the preprocessing. For example, in theory a dataset with high instance overlap should lower its instance metrics significantly after a cleaning algorithm is used, but the remaining metrics should maintain the same results as this algorithm is not meant to address the other families. Furthermore, if the reduction of these metrics is truly significative there should be a notable increase in performance when comparing the results of a classifier before and after preprocessing. If that is not the case then it could indicate that the metric is not measuring the type of overlap that was initially expected, or that this type of overlap does not affect the classifier being studied.

Another potential outcome is if out of all metrics of the same family one among them does not reduce its value significantly after preprocessing, while the others all reduce their value as expected. This could indicate that this metric does not belong in this family of the taxonomy, as all the other metrics reacted as expected and this metric remained the same indicating that it is not measuring the same type of overlap as the others.

3.3.1 Experimental Setup

In order to understand if the complexity metrics are indeed measuring the type overlap that is assigned to them in the taxonomy the following setup was followed: (i) Preprocessing, the datasets were processed using multiple oversampling and cleaning algorithms, each algorithm tries to reduce at least one of the types of overlap identified in the taxonomy. (ii) Complexity Extraction, where the complexity measures of both the original and the preprocessed datasets are taken. (iii) Classification/Performance Extraction, this process happens at the same time as the complexity extraction, both the preprocessed and original datasets are classified by a random forest and the performance of the model is measured using the F-Measure.

The main idea behind this process is that a preprocessing algorithm that reduces a specific type of overlap should lower the values of the complexity metrics of that family. Furthermore, when analysing the performance of the classification algorithms there should be higher boosts in performance for the datasets that had their complexity reduced.

The following preprocessing algorithms were considered: NB-Basic [50], NB-Tomek [50], SMOTE-ENN, MWMOTE [51], Feature Selection. NB-Basic and NB-Tomek are both cleaning algorithms, with NB-Tomek being an alteration to make NB-Basic more efficient at removing overlapped samples. It is expected that these two algorithms address primarily instance level overlap due to being neighbourhood based approaches. SMOTE-ENN is also expected to primarily reduce instance overlap for the same reason, but being an oversampling algorithm instead of a cleaning approach different results could occur. MWMOTE is an oversampling algorithm which considers the distribution of clusters on top of local information, and as such is expected to reduce both local and structural complexity. Finally, the Feature selection process aims at selecting the most discriminative features in the dataset and removing the remaining ones. This process should increase the feature overlap slightly, but it is predicted that the classification algorithms will perform equally or slightly better despite this.

As for the complexity measures used, three to four measures of each family were selected from the ones implemented in the package. The multi-resolution family was excluded from this study due to the fact it requires too much parametrization for each metric, finding good parameters for these metrics is too dependent on the dataset used, and initial results were very inconsistent. For the Feature metrics F1, F1v and the imbalanced version of F3 and F4 were picked. For the instance level metrics the imbalanced version of N3 and N4 alongside Augmented R (AugR) and kDN were chosen. The choice of these metrics was based on the objective of studying the sensitivity of metrics prepared for imbalance such as AugR in comparison to metrics like kDN which do not take this into account. Most of the remaining instance metrics were excluded as many are very similar to kDN and AugR and would not add much to the study. Finally, for the structural metrics the imbalanced version of N1, N2 alongside ONB were chosen. Usually T1 is used for structural metrics however ONB was opted as it is a considerable optimization in comparison. When the imbalance version of a metric was used two values were obtained one for the majority class and another for the minority class, as we are mostly interested in the minority class that value was chosen.

The datasets used were all taken from the imbalance section of the KEEL repository. A total of 53 binary datasets were chosen with high imbalance ratios. After complexity extraction two groups of datasets were formed according to the families of complexity measures they represented. Ideally three groups would be formed one for each of the families being studied, but it was only possible to find 2 groups one with instance and structural overlap high and feature overlap low, and another with high feature overlap and lower values for instance and structural overlap.

Dataset groups

The first group presents both high structural and instance overlap in at least one of the metrics while showing lower values for the feature overlap. While the values for the feature overlap might be higher for F1 and F3, the values are always be relatively low for F1v and F4 which represent more complete versions of both previously mentioned metrics. Table 3.11 shows the values for the instance, structural and feature complexity metrics.

dataset	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
glass-0-1-4-6_vs_2	0.9167	0.6667	0.1264	0.0833	0.9166	0.6361	0.4772	0.7568	0.3632	0.5417	0.0000
yeast-1-4-5-8_vs_7	0.8571	0.9048	0.0807	0.0494	0.9523	0.6107	0.4900	0.8506	0.5775	0.8313	0.5617
glass-0-1-6_vs_2	0.8333	0.8333	0.1526	0.1037	1.0000	0.5857	0.4735	0.7793	0.3439	0.5704	0.0963
glass-0-1-5_vs_2	0.8333	0.6667	0.1719	0.0992	1.0000	0.6368	0.4854	0.8787	0.3863	0.4959	0.0826
glass2	0.6667	0.6667	0.1333	0.0867	0.8333	0.5877	0.4430	0.7524	0.3892	0.5267	0.0600
yeast-1_vs_7	0.7143	0.5714	0.1050	0.0652	0.7619	0.5661	0.4629	0.7116	0.2786	0.7671	0.4627

Table 3.11: Complexity metrics for the group I datasets

For all datasets the values of N3 and N4 are above 0.5, and while the values of kDN appear to be quite low it should be noted that the highest value obtained was about 0.3. The values of AugR are also relatively low, but the highest value for any dataset was around

0.26.

Similarly to the previous family all datasets present a high value (above 0.5) for at least one of the structural metrics. Particularly all datasets have very high values for N1 and with slightly lower values for N2. Finally, even though the values for ONB are all below 0.5 the highest values for this metric were around 0.51, meaning that these are very high values considering the sample.

As mentioned, while the values of F1 and F3 can be high for some datasets, a corresponding low value for F1v and F4 are usually found, most likely indicating low feature complexity. The exception to this rule is yeast-1-4-5-8_vs_7 which has values slightly above 0.5. However, as the values for the instance and structural families are very high, which compensates for the high feature overlap.

The second group represents datasets with a low instance and structural overlap, but high feature level overlap, the opposite of the previous group. These datasets have at least one high feature level metric with high values, with the main focus being on the F1v and F4 metrics. Table 3.12 present the instance, structural and feature level overlap for the selected datasets.

dataset	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
pima	0.4947	0.4149	0.3193	0.2658	0.6117	0.4966	0.4447	0.6568	0.3463	0.9926	0.9591
yeast-0-2-5-6_vs_3-7-8-9	0.3286	0.5714	0.1048	0.0625	0.5571	0.4287	0.3108	0.6073	0.1887	0.8935	0.7898
yeast-0-2-5-7-9_vs_3-6-8	0.2143	0.2429	0.0511	0.0327	0.3429	0.3771	0.1649	0.3675	0.1126	0.8707	0.7486
yeast-0-5-6-7-9_vs_4	0.4722	0.4722	0.1086	0.0838	0.6944	0.5041	0.3460	0.4329	0.1987	0.9189	0.6973
ecoli-0-1-4-7_vs_2-3-5-6	0.4286	0.1905	0.0703	0.0339	0.4762	0.4448	0.2968	0.6395	0.1005	0.8856	0.6822

Table 3.12: Complexity metrics for the group II datasets

For the instance level metrics the values found were all below 0.5 for all the datasets. It should be noted however that values above 0.2 for both kDN and AugR might appear small, but these were some of the highest values found in all the datasets, potentially revealing some drawbacks of the these metrics' sensitivity. However, even if these values are considered high the remaining instance and structural level metrics are all low.

As for the structural level overlap similar values are below 0.5 were chosen. Some datasets still have a somewhat high value for N1, but present a lower value for N2 and ONB, so they were still selected for this category.

Finally, as previously mentioned, for the feature level overlap, the goal was to find datasets with both high F1v and F4. All the datasets shown present a F4 of above 0.6, an F3 of above 0.8, which will make up for ideal higher values of F1v which were not found in any dataset.

3.3.2 Results

This section shows the results of the complexity measures before and after the preprocessing algorithms were applied to the datasets. Alongside this the results of the RF classifier are also shown for every preprocessing algorithm used.

Unlike the Results section in the dataset shift section, the discussion and answers to the research questions are presented in a separate subsection following this one, since the results being shown help to answer both research questions and conclusions can only be drawn after all results are presented.

The sections analysing the complexity measures show tables for the datasets of group I and group II. These tables highlight in green decreases of metrics of more than 10% and in red increases of more than 10% after the use of a preprocessing algorithm.

NB-Basic Comparison

In this section the comparison between the complexity metrics of the two groups will be examined before and after the preprocessing of the NB-Basic algorithm. It is expected that the values of the instance lower significantly [13] as this algorithm uses local information with nearest neighbour approaches to remove overlapped samples. Table 3.13 shows the difference between the two approaches for the 3 families examined.

dataset	**preproc**	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
glass-0-1-4-6_vs_2	original	0.9167	0.6667	0.1264	0.0833	0.9166	0.6361	0.4772	0.7568	0.3632	0.5417	0.0000
glass-0-1-4-6_vs_2	NB-Basic	0.2500	0.0000	0.1500	0.0714	0.4166	0.4530	0.1287	0.3537	0.1761	0.3929	0.0000
yeast-1-4-5-8_vs_7	original	0.8571	0.9048	0.0807	0.0494	0.9523	0.6107	0.4900	0.8506	0.5775	0.8313	0.5617
yeast-1-4-5-8_vs_7	NB-Basic	0.3810	0.4762	0.0806	0.0645	0.5238	0.4791	0.3012	0.7793	0.3439	0.5704	0.0963
glass-0-1-6_vs_2	original	0.8333	0.8333	0.1526	0.1037	1.0000	0.5857	0.4735	0.7793	0.3439	0.5704	0.0963
glass-0-1-6_vs_2	NB-Basic	0.1667	0.0833	0.1276	0.0690	0.3333	0.3810	0.1793	0.4665	0.1790	0.4655	0.0000
glass-0-1-5_vs_2	original	0.8333	0.6667	0.1719	0.0992	1.0000	0.6368	0.4854	0.8787	0.3863	0.4959	0.0826
glass-0-1-5_vs_2	NB-Basic	0.2500	0.0000	0.2450	0.1250	0.5000	0.4349	0.1369	0.6452	0.1158	0.3500	0.0000
glass2	original	0.6667	0.6667	0.1333	0.0867	0.8333	0.5877	0.4430	0.7524	0.3892	0.5267	0.0600
glass2	NB-Basic	0.2500	0.0000	0.1424	0.1061	0.5833	0.4501	0.0879	0.3676	0.1512	0.3636	0.0000
yeast-1_vs_7	original	0.7143	0.5714	0.1050	0.0652	0.7619	0.5661	0.4629	0.7116	0.2786	0.7671	0.4627
yeast-1_vs_7	NB-Basic	0.4286	0.2381	0.1043	0.0929	0.52384	0.4764	0.2997	0.5379	0.1377	0.5571	0.1571

Table 3.13: Complexity measure values for before and after NB-Basic for the group I datasets

As expected the instance level metrics lowered significantly, specifically when looking at N3 and N4, however the same can not be said for kDN and AugR which seemed to either maintain their value or increase slightly.

As for the structural metrics there was also a clear decrease on all values, especially for N1 and ONB. While it was expected a decrease due to the removal of some samples which would simplify the structure of the dataset, the values lowered a lot more than it was initially expected.

Finally, for the feature level metrics also benefited from a decrease which was more significant than expected. However, the decrease was not as high as for some structural and instance level metrics analysed previously.

The results of this preprocessing algorithm are also shown for the datasets in group II in tables 3.14. Overall it is expected that the instance overlap metrics were lowered as they were with the previous datasets, but, because these values are already low to begin with, this should not cause a performance increase when analysing the performance of the classifier. As far as the other two families, following the literature [13], there should be a small decrease in structural overlap, but there should be no significant decrease in feature overlap as this method does not seek to address it, however the results from group I suggest that significative decreases in both can be seen.

The results seen for the instance level metrics are within expectations. The values lowered to almost zero in most cases, similarly to the datasets in group I. Even in the cases where one of the metrics did not reduce its value significantly, the remaining three more than compensated for it.

As for the structural metrics there was a significant decrease of N1 in most cases while, N2 and ONB decreased but not as significantly. Although it should be noted that out of the 3 metrics N1 was the one that presented the highest values, so it is expected that the

dataset	**preproc**	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
pima	original	0.4947	0.4149	0.3193	0.2658	0.6117	0.4966	0.4447	0.6568	0.3463	0.9926	0.9591
pima	NB-Basic	0.0106	0.0000	0.0311	0.0311	0.0372	0.2819	0.3160	0.2431	0.1274	0.0881	0.0000
yeast-0-5-6-7-9_vs_4	original	0.4722	0.4722	0.1086	0.0838	0.6944	0.5041	0.3460	0.4329	0.1987	0.9189	0.6973
yeast-0-5-6-7-9_vs_4	NB-Basic	0.0556	0.1111	0.0452	0.0407	0.1111	0.3720	0.1466	0.2995	0.0686	0.8281	0.1448
ecoli-0-1-4-7_vs_2-3-5-6	original	0.4286	0.1905	0.0703	0.0339	0.4762	0.4448	0.2968	0.6395	0.1005	0.8856	0.6822
ecoli-0-1-4-7_vs_2-3-5-6	NB-Basic	0.1905	0.0476	0.0392	0.0270	0.2857	0.3409	0.1864	0.4003	0.0619	0.8716	0.6014
yeast-0-2-5-6_vs_3-7-8-9	original	0.3286	0.5714	0.1048	0.0625	0.5571	0.4287	0.3108	0.6073	0.1887	0.8935	0.7898
yeast-0-2-5-6_vs_3-7-8-9	NB-Basic	0.0857	0.1714	0.0721	0.0697	0.1857	0.2753	0.1273	0.4731	0.1353	0.7787	0.3402
yeast-0-2-5-7-9_vs_3-6-8	original	0.2143	0.2429	0.0511	0.0327	0.3429	0.3771	0.1649	0.3675	0.1126	0.8707	0.7486
yeast-0-2-5-7-9_vs_3-6-8	NB-Basic	0.0571	0.2286	0.0173	0.0165	0.1143	0.2589	0.0728	0.3175	0.0914	0.7546	0.4969

Table 3.14: Complexity measure values for before and after NB-Basic for the group II datasets

others would not decrease as much.

Finally, the feature level metrics lowered in a much more significant way than expected. As mention previously, this algorithm was only meant to address the instance level overlap, however it seems that it also has a strong effect in reducing the feature level overlap metrics as well. The decrease was even more noticeable than the one seen for group I, although this is probably due to the fact that these values were much higher to begin with.

Similar results can be found in Appendix C for NB-Tomek (Table C.1 and Table C.2), another cleaning approach that follows a similar process.

MWMOTE Comparison

MWMOTE is an oversampling algorithm which focuses on data clusters and the overall structure of the dataset and not just the nearest neighbours of each sample like the previously mentioned cleaning approaches. Therefore, it is expected that this dataset reduces mainly the structural overlap of the datasets.

Table 3.15 shows the differences between the complexity measures of the original and oversampled datasets.

dataset	**preproc**	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
glass-0-1-4-6_vs_2	original	0.9167	0.6667	0.1264	0.0833	0.9167	0.6362	0.4773	0.7568	0.3632	0.5417	0.0000
glass-0-1-4-6_vs_2	MWMOTE	0.0343	0.0806	0.1571	0.1290	0.1947	0.2217	0.1542	0.7576	0.2574	0.7500	0.1828
yeast-1-4-5-8_vs_7	original	0.8571	0.9048	0.0807	0.0494	0.9524	0.6108	0.4900	0.8506	0.5775	0.8313	0.5617
yeast-1-4-5-8_vs_7	MWMOTE	0.0000	0.6992	0.0559	0.0527	0.6877	0.0018	0.0591	0.8533	0.5376	0.9118	0.7710
glass-0-1-6_vs_2	original	0.8333	0.8333	0.1526	0.1037	1.0000	0.5859	0.4736	0.7793	0.3439	0.5704	0.0963
glass-0-1-6_vs_2	MWMOTE	0.0171	0.0542	0.1597	0.1285	0.2160	0.2250	0.1444	0.7792	0.2376	0.7642	0.3419
glass-0-1-5_vs_2	original	0.8333	0.6667	0.1719	0.0992	1.0000	0.6369	0.4855	0.7524	0.3892	0.5267	0.0600
glass-0-1-5_vs_2	MWMOTE	0.0251	0.1080	0.1731	0.1445	0.2226	0.2382	0.1832	0.8762	0.2840	0.7202	0.3815
glass2	original	0.6667	0.6667	0.1333	0.0867	0.8333	0.5877	0.4438	0.7524	0.3892	0.5267	0.0600
glass2	MWMOTE	0.0193	0.0618	0.1494	0.1341	0.1913	0.2167	0.1484	0.7514	0.2646	0.7428	0.3729
yeast-1_vs_7	original	0.7143	0.5714	0.1050	0.0652	0.7619	0.5662	0.4629	0.7116	0.2786	0.7671	0.4627
yeast-1_vs_7	MWMOTE	0.0000	0.5753	0.0689	0.0663	0.4656	0.0053	0.0841	0.7078	0.2931	0.8754	0.6775

Table 3.15: Complexity Measure values before and after MWMOTE for the group I datasets

The reduction of the instance level metrics was very significative, especially for the case of N3 and N4 where the reduction was even larger than the cleaning algorithms analysed before. On the other hand the kDN and AugR metrics maintained the same values after the preprocessing algorithm was used.

The value of the structural level metrics decreased significantly, more so than any of the cleaning algorithms analysed previously. This is to be expected has mentioned previously this algorithm takes into consideration structural properties of the datasets.

Finally, the feature level overlap metrics also went according to what was expected to where no significant decrease was found between the two approaches, in fact in most cases the metrics increased drastically, especially the value of F3 and F4.

It is interesting to note the difference between this oversampling approach and the previously examined cleaning approaches when it comes to the feature overlap metrics, where the cleaning methods tend to reduce the amount of feature overlap, the contrary is observed for an oversampling approach.

Although this difference between approaches is noticeable, it is not yet clear if it changes the performance of the classification algorithms. For example, it is still possible that even though the feature overlap increased with this approach, that a classification algorithm will perform better after this preprocessing method was performed, when compared to a cleaning approach. Potentially signifying that the feature overlap metrics do not provide much information about how overlapped a domain is.

Similarly to the previous sections the results for the preprocessing of the datasets in group II is also shown (Table 3.16). Much like the previously shown algorithms, MWMOTE does not address feature overlap in any meaningful capacity. On the other hand it is expected a decrease on the structural and instance overlap metrics like what was observed for datasets in group I.

dataset	**preproc**	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
pima	original	0.4947	0.4149	0.3193	0.2658	0.6117	0.4966	0.4447	0.6568	0.3463	0.9926	0.9591
pima	MWMOTE	0.2277	0.2151	0.2892	0.2224	0.3944	0.4350	0.3883	0.6285	0.3285	0.9928	0.9662
yeast-0-5-6-7-9_vs_4	original	0.4722	0.4722	0.1086	0.0838	0.6944	0.5041	0.3460	0.4329	0.1987	0.9189	0.6973
yeast-0-5-6-7-9_vs_4	MWMOTE	0.0000	0.3780	0.0578	0.0508	0.4365	0.0063	0.0704	0.4336	0.2032	0.8889	0.7372
ecoli-0-1-4-7_vs_2-3-5-6	original	0.4286	0.1905	0.0703	0.0339	0.4762	0.4448	0.2968	0.6395	0.1005	0.8856	0.6822
ecoli-0-1-4-7_vs_2-3-5-6	MWMOTE	0.0344	0.0591	0.0754	0.0538	0.1040	0.1885	0.1053	0.5304	0.1307	0.9173	0.6963
yeast-0-2-5-6_vs_3-7-8-9	original	0.3286	0.5714	0.1048	0.0625	0.5571	0.4287	0.3108	0.6073	0.1887	0.8935	0.7898
yeast-0-2-5-6_vs_3-7-8-9	MWMOTE	0.0001	0.5244	0.0674	0.0663	0.2798	0.0057	0.0721	0.6173	0.1987	0.9035	0.7800
yeast-0-2-5-7-9_vs_3-6-8	original	0.2143	0.2429	0.0511	0.0327	0.3429	0.3771	0.1649	0.3675	0.1126	0.8707	0.7486
yeast-0-2-5-7-9_vs_3-6-8	MWMOTE	0.0001	0.3062	0.0350	0.0371	0.1592	0.0061	0.0437	0.3655	0.1322	0.8991	0.8255

Table 3.16: Complexity Measure values before and after MWMOTE for the group II datasets

The instance level metrics had a similar behaviour to the algorithms analysed before, where most of them decreased a fair amount. The amount by which the values decreased is not as significative as the ones observed in the NB-Basic and NB-Tomek algorithms or even as significative as the decreases seen on the datasets of group I. Granted the initial values of these datasets were not nearly as high.

Similarly to the instance level metrics the decrease seen for these datasets was not as significative as the ones seen for this group with the previous algorithms. Furthermore, the decrease was also not nearly as significative as the one seen for the datasets of group I, using this same algorithm, but similarly to the instance level metrics the values of the original datasets were not nearly as high in comparison to group I.

As for feature overlap metrics the results match the expected outcome, where for the most part no decrease was seen in any of the metrics, in some cases the values even increased after the algorithm was applied. This behaviour is very similar to what was seen in the datasets of group I analysed previously.

SMOTE-ENN

Similar to the previous algorithm, SMOTE-ENN also introduces new samples in the dataset, however after oversampling a cleaning approach is also undertaken to remove overlapped samples. Both these oversampling and cleaning approach are done taking into consideration the nearest neighbours of the samples and not any structural information provided by the dataset and as such there should not be a big decrease in structural overlap like it was found with the MWMOTE method.

Table 3.17 shows the results for the complexity measures before and after SMOTE-ENN is applied for the datasets of group I.

dataset	**preproc**	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
glass-0-1-4-6_vs_2	original	0.9167	0.6667	0.1264	0.0833	0.9167	0.6362	0.4773	0.7568	0.3632	0.5417	0.0000
glass-0-1-4-6_vs_2	SMOTE-ENN	0.0161	0.2177	0.1273	0.1094	0.1452	0.1540	0.1161	0.7720	0.2639	0.7383	0.0352
yeast-1-4-5-8_vs_7	original	0.8571	0.9048	0.0807	0.0494	0.9524	0.6108	0.4900	0.8506	0.5775	0.8313	0.5617
yeast-1-4-5-8_vs_7	SMOTE-ENN	0.0000	0.3848	0.1105	0.1005	0.1022	0.0816	0.0820	0.8093	0.4889	0.9114	0.7697
glass-0-1-6_vs_2	original	0.8333	0.8333	0.1526	0.1037	0.1022	0.0816	0.0820	0.7793	0.3439	0.5704	0.0963
glass-0-1-6_vs_2	SMOTE-ENN	0.0000	0.2000	0.1342	0.1111	0.1417	0.1516	0.1314	0.7852	0.2238	0.7613	0.2840
glass-0-1-5_vs_2	original	0.8333	0.6667	0.1719	0.0992	1.0000	0.6369	0.4855	0.8787	0.3863	0.4959	0.0826
glass-0-1-5_vs_2	SMOTE-ENN	0.0000	0.1414	0.1317	0.1202	0.1717	0.1210	0.1188	0.8934	0.2774	0.6971	0.3413
glass2	original	0.6667	0.6667	0.1333	0.0867	0.8333	0.5877	0.4438	0.7524	0.3892	0.5267	0.0600
glass2	SMOTE-ENN	0.0150	0.1429	0.1181	0.1033	0.1504	0.1513	0.1286	0.7595	0.2446	0.7269	0.1107
yeast-1_vs_7	original	0.7143	0.5714	0.1050	0.0652	0.7619	0.5662	0.4629	0.7116	0.2786	0.7671	0.4627
yeast-1_vs_7	SMOTE-ENN	0.0000	0.3390	0.1123	0.1231	0.1199	0.1011	0.0973	0.6710	0.2745	0.8735	0.7032

Table 3.17: Complexity Measure values before and after SMOTE-ENN for the group I datasets

Similar results are found between this approach and MWMOTE where N3 and N4 decrease notably but kDN and AugR more or less maintain the same values. When compared to the basic cleaning approaches it seems that N3 and N4 find a larger decrease, similar to what was observed with MWMOTE as well.

The results obtained for the structural metrics were very similar to the ones obtained with MWMOTE, however as mentioned previously this algorithm does not take into account structural properties of the dataset, so there should not have been a big decrease like what was observed previously with MWMOTE.

Finally, for the feature overlap metrics, a similar effect was found to the previous oversampling method, where no notable decrease was observed, with some cases even increasing the value.

The results for the group II datasets using this algorithm were also analysed and are shown in Table 3.18. Overall the results expected somewhat similar to the ones theorized for MWMOTE with a predicted decrease in the already small values of instance and structural overlap. The results for feature overlap are not as clear to predict as this algorithm both oversamples and cleans the dataset, and as such it is possible that it decreases feature overlap like what was observed for the cleaning approaches or that it maintains/increases it like it was seen for the MWMOTE oversampling approach.

dataset	**preproc**	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
pima	original	0.4947	0.4149	0.3193	0.2658	0.6117	0.4966	0.4447	0.6568	0.3463	0.9926	0.9591
pima	SMOTE-ENN	0.0800	0.5400	0.2193	0.1764	0.3900	0.3001	0.1868	0.4634	0.2271	0.8455	0.7618
yeast-0-5-6-7-9_vs_4	original	0.4722	0.4722	0.1086	0.0838	0.6944	0.5041	0.3460	0.4329	0.1987	0.9189	0.6973
yeast-0-5-6-7-9_vs_4	SMOTE-ENN	0.0031	0.2086	0.0648	0.0591	0.0675	0.1203	0.0786	0.4187	0.1755	0.9212	0.7803
ecoli-0-1-4-7_vs_2-3-5-6	original	0.4286	0.1905	0.0703	0.0339	0.4762	0.4448	0.2968	0.6395	0.1005	0.8856	0.6822
ecoli-0-1-4-7_vs_2-3-5-6	SMOTE-ENN	0.0000	0.1061	0.0397	0.0315	0.0657	0.0826	0.0506	0.6223	0.1073	0.9056	0.6707
yeast-0-2-5-6_vs_3-7-8-9	original	0.3286	0.5714	0.1048	0.0625	0.5571	0.4287	0.3108	0.6073	0.1887	0.8935	0.7898
yeast-0-2-5-6_vs_3-7-8-9	SMOTE-ENN	0.0017	0.4307	0.0865	0.0766	0.0842	0.0852	0.0819	0.5862	0.2378	0.9298	0.8702
yeast-0-2-5-7-9_vs_3-6-8	original	0.2143	0.2429	0.0511	0.0327	0.3429	0.3771	0.1649	0.3675	0.1126	0.8707	0.7486
yeast-0-2-5-7-9_vs_3-6-8	SMOTE-ENN	0.0000	0.2620	0.0406	0.0397	0.0527	0.0686	0.0444	0.3639	0.1204	0.9016	0.8310

Table 3.18: Complexity Measure values before and after SMOTE-ENN for the group II datasets

Both the instance and structural metrics behaved as expected as both decreased somewhat in comparison to its original values.

Overall the result for the feature overlap metrics was still a positive as most of the values still decreased, however most likely due to the nature of the oversampling part of the algorithm the decrease in value was not nearly as much as the ones seen by the cleaning approaches. However, unlike MWMOTE there was still a noticeable decrease overall.

Feature Selection (FS)

The idea behind this preprocessing method is to reduce the amount of redundant features. While removing these features will probably increase the feature based metrics it is unclear if the performance of classifiers will be affected by this increase. Table 3.19 shows the results for the group I datasets before and after this pre-processing method, while Table 3.20 shows the results for group II.

dataset	preproc	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
glass-0-1-4-6_vs_2	orig	0.9167	0.6667	0.1264	0.0833	0.9167	0.6362	0.4773	0.7568	0.3632	0.5417	0.0000
glass-0-1-4-6_vs_2	FS	0.8333	0.7500	0.1250	0.0694	0.9167	0.8072	0.5114	0.4507	0.3010	0.3819	0.3819
yeast-1-4-5-8_vs_7	orig	0.8571	0.9048	0.0807	0.0494	0.9524	0.6108	0.4900	0.8506	0.5775	0.8313	0.5617
yeast-1-4-5-8_vs_7	FS	0.9048	0.8571	0.0790	0.0432	1.0000	0.9387	0.5074	0.7265	0.5043	0.9177	0.9177
glass-0-1-6_vs_2	orig	0.8333	0.8333	0.1526	0.1037	1.0000	0.5859	0.4736	0.7793	0.3439	0.5704	0.0963
glass-0-1-6_vs_2	FS	0.6667	0.5000	0.1096	0.0963	0.7500	0.4287	0.4116	0.4033	0.2939	0.2815	0.2815
glass-0-1-5_vs_2	orig	0.8333	0.6667	0.1719	0.0992	1.0000	0.6369	0.4855	0.8787	0.3863	0.4959	0.0826
glass-0-1-5_vs_2	FS	0.6667	0.7500	0.1405	0.1074	0.9167	0.6837	0.4255	0.4958	0.3220	0.3967	0.3967
glass2	orig	0.6667	0.6667	0.1333	0.0867	0.8333	0.5877	0.4438	0.7524	0.3892	0.5267	0.0600
glass2	FS	0.9167	0.3333	0.1320	0.1133	1.0000	0.8460	0.5054	0.4339	0.3729	0.2200	0.2200
yeast-1_vs_7	orig	0.7143	0.5714	0.1050	0.0652	0.7619	0.5662	0.4629	0.7116	0.2786	0.7671	0.4627
yeast-1_vs_7	FS	0.8571	0.7143	0.1056	0.0745	0.9048	0.5668	0.4635	0.4732	0.2419	0.7422	0.7422

Table 3.19: Complexity measure values before and after Feature Selection for the group I datasets

dataset	preproc	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
pima	original	0.4947	0.4149	0.3193	0.2658	0.6117	0.4966	0.4447	0.6568	0.3463	0.9926	0.9591
pima	FS	0.4628	0.3936	0.3156	0.2509	0.6489	0.3893	0.4284	0.5252	0.3229	0.9870	0.9870
yeast-0-5-6-7-9_vs_4	original	0.4722	0.4722	0.1086	0.0838	0.6944	0.5041	0.3460	0.4329	0.1987	0.9189	0.6973
yeast-0-5-6-7-9_vs_4	FS	0.5000	0.5833	0.1038	0.0811	0.6944	0.3200	0.3276	0.3613	0.1584	0.8135	0.8135
ecoli-0-1-4-7_vs_2-3-5-6	original	0.4286	0.1905	0.0703	0.0339	0.4762	0.4448	0.2968	0.6395	0.1005	0.8856	0.6822
ecoli-0-1-4-7_vs_2-3-5-6	FS	0.3810	0.4762	0.0661	0.0508	0.4762	0.3135	0.2614	0.3705	0.1001	0.7839	0.7839
yeast-0-2-5-6_vs_3-7-8-9	original	0.3286	0.5714	0.1048	0.0625	0.5571	0.4287	0.3108	0.6073	0.1887	0.8935	0.7898
yeast-0-2-5-6_vs_3-7-8-9	FS	0.5571	0.5143	0.1085	0.0696	0.7286	0.3508	0.3680	0.5032	0.1458	0.9233	0.9233
yeast-0-2-5-7-9_vs_3-6-8	original	0.2143	0.2429	0.0511	0.0327	0.3429	0.3771	0.1649	0.3675	0.1126	0.8707	0.7486
yeast-0-2-5-7-9_vs_3-6-8	FS	0.2286	0.1714	0.0494	0.0256	0.3143	0.1926	0.1459	0.2730	0.1022	0.8168	0.8168

Table 3.20: Complexity measure values before and after Feature Selection for the group II datasets

Like it was expected the value of metrics like F4 increased drastically, on the other hand F1 and F3 seems to remain the same or decrease slightly. As for the remaining families some values seemed to increase slightly while others decreased, but these variations were not significative enough and most measures remained the same before and after preprocessing.

3.3.3 Classifier Performance

This section details the performance of the Random Forest classifier used before and after all the preprocessing algorithms (SMOTE-ENN, NB-Basic, NB-Tomek, MWMOTE and Feature Selection) were used as well as the difference between both scenarios. These results are presented in Tables 3.21-3.24 except NB-Tomek which can be found in appendix C (Table C.3) due to obtaining similar results to NB-Basic. These tables are sorted from the highest difference to lowest, thus it is expected that for MWMOTE, SMOTE-ENN, NB-Basic and NB-Tomek datasets from group I rank among the highest. While for Feature Selection it is expected that some datasets show slight increases due to the reduction of dimensionality, despite the increase in feature overlap.

Dataset	Difference	original	SMOTE-ENN
glass-0-1-6_vs_5	0.7778	0.0222	0.8000
glass-0-1-6_vs_2	**0.5401**	**0.0222**	**0.5623**
cleveland-0_vs_4	0.4833	0.0167	0.5000
glass-0-1-4-6_vs_2	**0.4286**	**0.0000**	**0.4286**
glass2	**0.3079**	**0.0889**	**0.3968**
glass5	0.2778	0.4000	0.6778
ecoli3	0.2356	0.4587	0.6943
yeast-0-5-6-7-9_vs_4	0.2281	0.3446	0.5727
ecoli-0-3-4-6_vs_5	0.1508	0.7883	0.9390
yeast-1-4-5-8_vs_7	**0.1006**	**0.0000**	**0.1006**
ecoli-0-3-4_vs_5	0.0874	0.9066	0.9939
glass-0-1-5_vs_2	**0.0369**	**0.0000**	**0.0369**
yeast3	0.0368	0.6719	0.7087
glass-0-4_vs_5	0.0333	0.9667	1.0000
vehicle1	0.0291	0.5114	0.5405
ecoli-0-1_vs_5	0.0282	0.8146	0.8428
yeast-0-2-5-6_vs_3-7-8-9	0.0223	0.5845	0.6068
ecoli-0-1-4-7_vs_5-6	0.0209	0.6415	0.6624
ecoli-0_vs_1	0.0120	0.9880	1.0000
newthyroid2	0.0039	0.8947	0.8986
vehicle2	0.0008	0.9914	0.9921
glass-0-1-2-3_vs_4-5-6	0.0001	0.8552	0.8553
glass-0-6_vs_5	0.0000	1.0000	1.0000
iris0	0.0000	1.0000	1.0000
shuttle-c2-vs-c4	0.0000	1.0000	1.0000
vowel0	-0.0012	0.9824	0.9812
ecoli1	-0.0057	0.7398	0.7342
ecoli-0-1-4-6_vs_5	-0.0122	0.6970	0.6848
vehicle0	-0.0139	0.9326	0.9187
ecoli2	-0.0146	0.8119	0.7973
yeast-2_vs_4	-0.0191	0.8310	0.8120
page-blocks-1-3_vs_4	-0.0238	0.9441	0.9203
new-thyroid1	-0.0283	0.9474	0.9191
yeast-0-2-5-7-9_vs_3-6-8	-0.0354	0.8532	0.8177
ecoli-0-6-7_vs_3-5	-0.0365	0.8298	0.7933
vehicle3	-0.0411	0.5390	0.4980
ecoli-0-4-6_vs_5	-0.0414	0.9055	0.8641
ecoli-0-1-4-7_vs_2-3-5-6	-0.0443	0.8983	0.8540
ecoli-0-1_vs_2-3-5	-0.0460	0.6925	0.6466
ecoli-0-1-3-7_vs_2-6	-0.0478	0.5778	0.5300
ecoli-0-6-7_vs_5	-0.0480	0.8322	0.7842
pima	-0.0892	0.6512	0.5620
glass0	-0.0906	0.7171	0.6265
ecoli-0-2-6-7_vs_3-5	-0.0990	0.8727	0.7737
yeast-2_vs_8	-0.1032	0.4786	0.3753
glass6	-0.1094	0.9333	0.8239
ecoli-0-3-4-7_vs_5-6	-0.1166	0.9974	0.8808
glass4	-0.1365	0.7302	0.5937
yeast-1_vs_7	**-0.1368**	**0.4949**	**0.3581**
glass1	-0.1376	0.7201	0.5825
yeast-0-3-5-9_vs_7-8	-0.1583	0.4626	0.3043
ecoli4	-0.2559	0.8848	0.6289

Table 3.21: Performance of Random Forest before and after preprocessing using SMOTE-ENN. The datasets from group I are marked in bold

For the results with SMOTE-ENN (Table 3.21), most datasets from group I are on the top of the table presenting the highest differences. The standout from group I is yeast-1_vs_7 which seemed to get worse after preprocessing showing up at the bottom of the list. Furthermore, some other datasets such as cleveland-0_vs_4, glass5, ecoli3, yeast-0-5-6-7-9_vs_4 and ecoli-0-3-4-6_vs_5 also have with high differences, even above yeast-1-4-5-8_vs_7, a dataset which is part of group I.

The datasets from group II show up mostly on the bottom or middle of the table as they seem to remain unchanged after the preprocessing, which is in accordance to what was expected. The exception to this rule is the aforementioned yeast-0-5-6-7-9_vs_4 which shows differences almost as large as other datasets in group II.

A similar pattern is observed with the datasets preprocessed with NB-Basic (Table 3.22), the members of group I mostly find themselves at the top of the table, except for yeast-1_vs_7 which shows almost no change. Furthermore, the same datasets seen in the previous preprocessing method that did not belong to group I are once again at the top of the table indicating that potentially, these datasets have some characteristic that causes them to show high differences.

As for the results with MWMOTE (Table 3.23), once more the same datasets from group I show up at the top of the table, indicating a very consistent behaviour when a technique is used that reduced their high structural and instance overlap. Similarly, the same outlier datasets that do not belong to group I are also found again among the top differences, solidifying the consistent behaviour found previously.

Dataset	Difference	original	NB-Basic
cleveland-0_vs_4	0.6889	0.0167	0.7056
glass-0-1-5_vs_2	**0.3900**	**0.0000**	**0.3900**
glass-0-1-4-6_vs_2	**0.3642**	**0.0000**	**0.3642**
glass-0-1-6_vs_2	**0.3359**	**0.0222**	**0.3581**
ecoli-0-1-3-7_vs_2-6	0.3222	0.5778	0.9000
glass2	**0.3189**	**0.0889**	**0.4078**
glass-0-1-6_vs_5	0.3094	0.0222	0.3316
yeast-1-4-5-8_vs_7	**0.2346**	**0.0000**	**0.2346**
ecoli-0-1_vs_2-3-5	0.1601	0.6925	0.8527
yeast-0-5-6-7-9_vs_4	0.1105	0.3446	0.4551
yeast3	0.0854	0.6719	0.7573
newthyroid2	0.0770	0.8947	0.9717
ecoli1	0.0602	0.7398	0.8000
new-thyroid1	0.0463	0.9474	0.9937
glass-0-1-2-3_vs_4-5-6	0.0405	0.8552	0.8957
glass-0-4_vs_5	0.0333	0.9667	1.0000
ecoli-0-1_vs_5	0.0188	0.8146	0.8334
yeast-2_vs_8	0.0140	0.4786	0.4926
glass-0-6_vs_5	0.0000	1.0000	1.0000
iris0	0.0000	1.0000	1.0000
shuttle-c2-vs-c4	0.0000	1.0000	1.0000
ecoli-0-4-6_vs_5	-0.0012	0.9055	0.9043
ecoli-0-1-4-6_vs_5	-0.0100	0.6970	0.6871
vehicle1	-0.0180	0.5114	0.4934
ecoli3	-0.0197	0.4587	0.4390
glass6	-0.0274	0.9333	0.9059
ecoli2	-0.0442	0.8119	0.7677
yeast-1_vs_7	**-0.0621**	**0.4949**	**0.4327**
yeast-0-2-5-7-9_vs_3-6-8	-0.0660	0.8532	0.7872
ecoli-0-6-7_vs_5	-0.0667	0.8322	0.7656
ecoli4	-0.0720	0.8848	0.8128
ecoli-0-3-4_vs_5	-0.0729	0.9066	0.8337
glass5	-0.0731	0.4000	0.3269
vehicle3	-0.0776	0.5390	0.4614
ecoli-0-1-4-7_vs_5-6	-0.0785	0.6415	0.5630
ecoli-0-3-4-6_vs_5	-0.0807	0.7883	0.7076
ecoli-0-6-7_vs_3-5	-0.0807	0.8298	0.7491
ecoli-0_vs_1	-0.0917	0.9880	0.8963
yeast-0-2-5-6_vs_3-7-8-9	-0.0936	0.5845	0.4909
vowel0	-0.1066	0.9824	0.8758
yeast-2_vs_4	-0.1097	0.8310	0.7213
glass0	-0.1121	0.7171	0.6049
ecoli-0-2-6-7_vs_3-5	-0.1167	0.8727	0.7560
pima	-0.1256	0.6512	0.5257
glass1	-0.1333	0.7201	0.5868
yeast-0-3-5-9_vs_7-8	-0.1964	0.4626	0.2663
vehicle0	-0.1999	0.9326	0.7327
glass4	-0.2557	0.7302	0.4744
ecoli-0-1-4-7_vs_2-3-5-6	-0.3048	0.8983	0.5935
page-blocks-1-3_vs_4	-0.3506	0.9441	0.5935
ecoli-0-3-4-7_vs_5-6	-0.4168	0.9974	0.5806
vehicle2	-0.6665	0.9914	0.3248

Table 3.22: Performance of Random Forest before and after preprocessing using NB-Basic. The datasets from group I are marked in bold

Dataset	Difference	original	MWMOTE
glass-0-1-6_vs_5	0.9629	0.0222	0.9851
cleveland-0_vs_4	0.4733	0.0167	0.4900
glass-0-1-6_vs_2	**0.4441**	**0.0222**	**0.4663**
glass2	**0.3239**	**0.0889**	**0.4128**
glass5	0.2726	0.4000	0.6726
glass-0-1-4-6_vs_2	**0.2017**	**0.0000**	**0.2017**
ecoli3	0.1908	0.4587	0.6495
glass-0-1-5_vs_2	**0.1109**	**0.0000**	**0.1109**
ecoli-0-3-4-6_vs_5	0.0993	0.7883	0.8875
vehicle1	0.0974	0.5114	0.6088
ecoli-0-1-4-7_vs_5-6	0.0880	0.6415	0.7294
ecoli-0-1_vs_2-3-5	0.0822	0.6925	0.7747
yeast-0-5-6-7-9_vs_4	0.0817	0.3446	0.4263
vehicle3	0.0747	0.5390	0.6138
glass-0-4_vs_5	0.0333	0.9667	1.0000
yeast3	0.0281	0.6719	0.7000
ecoli-0-4-6_vs_5	0.0203	0.9055	0.9257
yeast-2_vs_4	0.0194	0.8310	0.8505
pima	0.0191	0.6512	0.6703
ecoli-0-1-4-6_vs_5	0.0185	0.6970	0.7155
glass-0-1-2-3_vs_4-5-6	0.0179	0.8552	0.8731
ecoli-0-3-4_vs_5	0.0164	0.9066	0.9229
yeast-1-4-5-8_vs_7	**0.0075**	**0.0000**	**0.0075**
ecoli-0-2-6-7_vs_3-5	0.0065	0.8727	0.8792
yeast-1_vs_7	**0.0019**	**0.4949**	**0.4968**
newthyroid2	0.0014	0.8947	0.8962
glass-0-6_vs_5	0.0000	1.0000	1.0000
iris0	0.0000	1.0000	1.0000
shuttle-c2-vs-c4	0.0000	1.0000	1.0000
vehicle0	-0.0017	0.9326	0.9309
glass1	-0.0018	0.7201	0.7182
glass0	-0.0021	0.7171	0.7149
yeast-0-2-5-6_vs_3-7-8-9	-0.0023	0.5845	0.5821
ecoli-0-1_vs_5	-0.0036	0.8146	0.8111
yeast-0-2-5-7-9_vs_3-6-8	-0.0042	0.8532	0.8490
vehicle2	-0.0056	0.9914	0.9858
page-blocks-1-3_vs_4	-0.0074	0.9441	0.9368
new-thyroid1	-0.0088	0.9474	0.9386
ecoli1	-0.0099	0.7398	0.7300
vowel0	-0.0134	0.9824	0.9690
ecoli-0-6-7_vs_5	-0.0175	0.8322	0.8147
ecoli2	-0.0180	0.8119	0.7939
ecoli-0_vs_1	-0.0195	0.9880	0.9685
ecoli-0-6-7_vs_3-5	-0.0196	0.8298	0.8102
ecoli-0-1-4-7_vs_2-3-5-6	-0.0421	0.8983	0.8562
yeast-0-3-5-9_vs_7-8	-0.0491	0.4626	0.4135
yeast-2_vs_8	-0.0547	0.4786	0.4238
ecoli-0-1-3-7_vs_2-6	-0.0587	0.5778	0.5190
glass6	-0.1145	0.9333	0.8188
glass4	-0.1419	0.7302	0.5883
ecoli-0-3-4-7_vs_5-6	-0.1835	0.9974	0.8139
ecoli4	-0.2632	0.8848	0.6216

Table 3.23: Performance of Random Forest before and after preprocessing using MWMOTE. The datasets from group I are marked in bold

Finally, the results for Feature Selection (Table 3.24) are within expectation. While the values of the feature metrics increased, the performance of the classifier in most datasets tended to remain the same. There were some exceptions in the case of glass5 and glass-0-1-6_vs_5 which saw a increase in 0.3 and ecoli-0-1-4-7_vs_2-3-5-6 which decreased 0.4. But in comparison to the remaining algorithms most datasets tended to maintain the same

Dataset	Difference	original	FS	Dataset	Difference	original	FS
glass5	0.3000	0.0000	0.3000	new-thyroid1	0.0000	0.9474	0.9474
glass-0-1-6_vs_5	0.3000	0.0000	0.3000	newthyroid2	0.0000	0.8889	0.8889
glass-0-1-6_vs_2	0.2857	0.0000	0.2857	shuttle-c2-vs-c4	0.0000	1.0000	1.0000
glass-0-1-4-6_vs_2	0.1333	0.0000	0.1333	yeast-1-4-5-8_vs_7	0.0000	0.0000	0.0000
ecoli4	0.1231	0.8000	0.9231	yeast-2_vs_8	0.0000	0.5000	0.5000
ecoli-0-1-4-6_vs_5	0.0606	0.6667	0.7273	glass0	-0.0393	0.7222	0.6829
yeast-2_vs_4	0.0419	0.7857	0.8276	pima	-0.0465	0.6216	0.5752
ecoli-0-6-7_vs_5	0.0333	0.8000	0.8333	ecoli1	-0.0472	0.7556	0.7083
page-blocks-1-3_vs_4	0.0078	0.9333	0.9412	yeast3	-0.0493	0.6829	0.6337
cleveland-0_vs_4	0.0000	0.0000	0.0000	vowel0	-0.0532	0.9818	0.9286
ecoli-0-1-3-7_vs_2-6	0.0000	0.6667	0.6667	ecoli-0-3-4-7_vs_5-6	-0.0769	1.0000	0.9231
ecoli-0-1_vs_2-3-5	0.0000	0.7273	0.7273	ecoli2	-0.0800	0.8000	0.7200
ecoli-0-1_vs_5	0.0000	0.9091	0.9091	glass-0-1-2-3_vs_4-5-6	-0.1108	0.8966	0.7857
ecoli-0-2-6-7_vs_3-5	0.0000	0.9091	0.9091	yeast-0-2-5-7-9_vs_3-6-8	-0.1164	0.8571	0.7407
ecoli-0-3-4-6_vs_5	0.0000	0.8000	0.8000	ecoli-0-1-4-7_vs_5-6	-0.1273	0.7273	0.6000
ecoli-0-3-4_vs_5	0.0000	0.9091	0.9091	yeast-1_vs_7	-0.1282	0.4615	0.3333
ecoli-0-4-6_vs_5	0.0000	0.9091	0.9091	vehicle0	-0.1286	0.9421	0.8136
ecoli-0-6-7_vs_3-5	0.0000	0.8333	0.8333	yeast-0-5-6-7-9_vs_4	-0.1688	0.4545	0.2857
ecoli-0_vs_1	0.0000	0.9787	0.9787	yeast-0-2-5-6_vs_3-7-8-9	-0.1733	0.5333	0.3600
ecoli3	0.0000	0.5000	0.5000	vehicle3	-0.1768	0.5000	0.3232
glass-0-1-5_vs_2	0.0000	0.0000	0.0000	glass-0-6_vs_5	-0.2000	1.0000	0.8000
glass-0-4_vs_5	0.0000	1.0000	1.0000	vehicle1	-0.3072	0.5000	0.1928
glass2	0.0000	0.3333	0.3333	yeast-0-3-5-9_vs_7-8	-0.3211	0.3211	0.0000
glass4	0.0000	0.8000	0.8000	glass1	-0.3288	0.6692	0.3404
glass6	0.0000	0.9333	0.9333	vehicle2	-0.3698	0.7848	0.4151
iris0	0.0000	1.0000	1.0000	ecoli-0-1-4-7_vs_2-3-5-6	-0.4000	0.4000	0.0000

Table 3.24: Performance of Random Forest before and after preprocessing using FS

performance before and after preprocessing which is precisely what was expected.

Outlier Group

When analysing the performance of the classifiers in the previous section some datasets which didn't belong to group I seemed to consistently show up with high differences between the original and the preprocessed dataset.

This section will go into some detail analysing those datasets trying to understand why values observed were obtained, by looking at their complexity metrics before and after preprocessing.

Table 3.25 shows the complexity measures for the outlier datasets divided by instance, structural and feature overlap metrics.

dataset	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
cleveland-0_vs_4	0.4000	0.9000	0.0704	0.0720	0.5000	0.5052	0.5109	0.3346	0.0535	0.7760	0.0000
ecoli-0-1-3-7_vs_2-6	0.4000	0.4000	0.0386	0.0355	0.6000	0.3609	0.3260	0.3845	0.0395	0.2081	0.0000
glass-0-1-6_vs_5	0.1429	0.1429	0.0477	0.0308	0.7143	0.4232	0.2427	0.3819	0.1542	0.1462	0.0000
glass5	0.4286	0.1429	0.0517	0.0397	0.7143	0.4232	0.2427	0.4328	0.1054	0.1391	0.0000
ecoli3	0.6000	0.1600	0.1102	0.0975	0.6800	0.5684	0.3698	0.3661	0.1623	0.4449	0.2161

Table 3.25: Instance overlap values for the outlier datasets

When analysing the instance overlap metrics the only high value obtained is N4 for cleveland-0_vs_4, which could help explain why this dataset always showed very high differences when a preprocessing algorithm that addressed instance overlap was used. As for the remaining datasets none show high values for any of the metrics, so their high differences can not be justified by the amount of instance overlap present.

As for structural metrics cleveland-0_vs_4 has average values for N1 and N2 and while its

ONB value is also in the middle of the scale, it is worth noting that values above 0.5 were rarely found indicating that for this metric a value such as this could already indicate a very high structural overlap.

Furthermore, the remaining datasets have at least one of the structural complexity metrics above 0.5. In some cases such as ecoli3 both N1 and N2 are above this threshold.

Except for cleveland-0_vs_4 which has high instance and a somewhat high structural overlap, the remaining 4 datasets all have average instance overlap and relatively high structural overlap.

Finally, for the feature overlap, the values are more or less the same as the datasets found in group I with overall low F1v and F4 scores. In most cases all the datasets in this group also have very low F1 and F3 scores except for cleveland-0_vs_4 which has a relatively high F3.

The outlier datasets found, except for cleveland-0_vs_4 which could be considered part of group I, can be joined in a third group of high structural overlap, average instance overlap and low feature overlap. It is expected that after preprocessing that the structural metrics will go down, which will justify why such high differences were found between the original and the preprocessed datasets.

In the appendix C the values before and after preprocessing for MWMOTE, SMOTE-ENN, NB-Tomek and NB-Basic (Tables C.4-C.7) can be seen. These results effectively reflect what was already seen in the dataset from group I, where notable reductions were seen for instance and structural overlap for every algorithm. Furthermore, the feature overlap values also lowered significantly for the cleaning approaches also similar to what was observed with the group I datasets.

3.3.4 Discussion

RQ3: What is the validity of the complexity measure taxonomy?

At first glance it might seem apparent that the instance and structural metrics are linked, since in a lot of cases it is notable that when the instance overlap increases/decreases, the structural overlap follows the same behaviour. This fact could be indicative of two possibilities: (i) There is a lack of variety of datasets openly available in online repositories in regard to the different families of measures; (ii) The instance and structural overlap metrics are inherently linked and as such when the instance overlap is high, the structural overlap follows to have high values as well.

After careful observation an outlier group was found that presents relatively high structural overlap with lower values for instance overlap, indicating that it may be possible to find one without the other. Although the difference between the values of these two families is not as noticeable as the difference found between instance and feature overlap for datasets in group I. Which still indicates the possibility that there is a limit to how much structural metrics can vary given a value for instance metrics. Even if it is possible to find cases where these two families of complexity exist separately there is a need for more datasets to be made available that distinguish not only between these two families, but, if possible, but between the four families of the taxonomy. That way more experiments could be done isolating each of the families a priori which could better help understand the behaviour of these metrics and preprocessing algorithms.

As far as the sensibility of the metrics is concerned, it is important to note that even though all metrics are on a scale from 0 to 1, where in theory 0.5 would indicate an average complexity scenario, some metrics do not behave this way. For example, ONB presents its highest values at around 0.5, which is in theory average complexity, but these values are usually accompanied by high values for N1 and N2. This indicates that 0.5 is already a high value for ONB, which might be due to the fact that this metric gives a singular value for both classes. Even though this value is weighted to account for class imbalance the majority class can still heavily influence the final result. Meaning values higher than 0.5 can only be achieved if most samples in the majority class are also overlapped, which is not the case in a lot of scenarios.

A similar argument can be used for the kDN and AugR metrics which do not seem to achieve high values overall, only obtaining results around 0.2-0.3. These values are usually accompanied by high results for N3 and N4. Since kDN and AugR also show a singular value for both classes they can be heavily influenced by the majority class samples that are not overlapped, not allowing the metric to achieve high results. This implies that values around 0.2-0.3 already potentially indicate a high overlap scenario. Furthermore, both of these metrics seemed to be more resilient to the use of preprocessing algorithms unlike N3 and N4 which always lowered. While this could be indicative that kDN and AugR are in the wrong overlap family, this is most likely due to the effects of imbalance, as explained previously. As such it is safe to assume that the four instance measures analysed belong to the correct family of the taxonomy. The same argument can be made for the structural metrics (N1, N2 and ONB) as datasets in group I saw a consistent reduction for the intended preprocessing algorithms and had a matching high performance with the RF classifier. Although, considering the possibility that the instance and structural family are linked, some of these metrics may be measuring both types of overlap simultaneously.

As for the feature metrics, most values consistently increased or decreased, after the use of the preprocessing algorithms. Even for cases where a decrease/increase was not expected, it could be justified by the choice of preprocessing algorithm and not by virtue of the metrics' properties. These results indicate their correct placement in the taxonomy. Despite this, the feature level metrics seemed to provide a very lacking picture of how complex the problem truly was, as even when there were increases to their values, the classification algorithm still achieved higher performances after preprocessing, like in the case of MWMOTE and Feature Selection preprocessing algorithms.

Finally, it is worth mentioning again that the multi-resolution metrics were not experimented with, due to the lack of consistent values obtained, caused by the difficulty in parameterization. While in theory these measures could provide a better picture of a dataset's complexity by using both global and local information, they are not very practical to use in studies where a lot of datasets are used at once, as each dataset needs a specific set of parameters.

Knowing this the main takeaways from these experiments were:

- (i) The structural and instance level overlap metrics might be linked, as both families seem to behave in the same way;
- (ii) There is a lack of variety of datasets available representative of the 3 families of overlap analysed;
- (iii) Augmented R-Value and kDN seemed to behave differently from N3 and N4. This is not necessarily indicative that they measure different types of overlap, as it is most likely due to the effects of imbalance;
- (iv) While in theory only values above 0.5 should indicate high overlap, metrics like ONB already indicate this scenario with values between 0.4 - 0.5. This is most likely due to the existence of imbalance which is known to hide the effects of overlap. This shows the usefulness of splitting the metrics into two values, one for the majority class and one for the minority;
- (v) Feature level metrics appear to not be as relevant as the remaining two families studied. This was especially notable with the case of the Feature Selection algorithm which saw these metrics increase, but the performance of the classifier remained mostly the same before and after preprocessing.

RQ4: How effective are preprocessing algorithms in what concerns to overlap reduction?

Another point to analyse is how the preprocessing methods were expected to behave and how they actually worked in practice. Out of the 5 preprocessing techniques used (NB-Basic, NB-Tomek, and SMOTE-ENN, MWMOTE and Feature Selection) it was expected that techniques that only consider a local neighbourhood such as NB-Basic, NB-Tomek and SMOTE-ENN would mainly reduce instance level overlap [13] and MWMOTE which also considers structural information would have a bigger effect on the reduction of structural metrics as well.

The observed results showed that, when compared to the cleaning algorithms, MWMOTE reduced instance overlap somewhat similarly, for the datasets in group I. While when looking at the values of N3 it might seem that MWMOTE performed better, no differences are seen for kDN. Furthermore, AugR and N4 show a higher reduction when the cleaning approaches are used. So overall no approach seems superior to the other when only observing these values.

On the other hand, like it was expected MWMOTE was able to reduce structural overlap much more than the cleaning algorithms. However, when compared to SMOTE-ENN both seem to reduce structural overlap significantly. In many cases SMOTE-ENN even surpasses MWMOTE in reducing the value of these metrics. This last result is unexpected as SMOTE-ENN does not actively deal with structural issues as MWMOTE attempts to. It is clear that even algorithms like NB-Basic and NB-Tomek which do not consider structural overlap still address it just by virtue of removing samples from the dataset, removing complexity from the structure. Therefore, it is not impossible that SMOTE-ENN was also able to address structural problems present in the dataset.

It is also interesting to note that while cleaning approaches were only meant to reduce instance overlap a fair reduction was also seen in the feature overlap metrics. While it is true that this algorithm only cares about local information, by removing invasive samples each feature becomes more discriminative. As such, it follows that the feature metrics also lower, since their values are tied to the discriminative power of the dataset features. On the other hand, when looking at an oversampling algorithm like MWMOTE, which inserts samples in the dataset, the opposite effect was seen and a lot of the values for the feature metrics increased as a consequence.

When analysing the performance increase of the classification algorithms after preprocessing we notice that MWMOTE and SMOTE-ENN provide higher performance increases when compared to the cleaning algorithms. So even though the cleaning algorithms reduce feature overlap, this reduction is not reflected in the performance of the classification algorithms. This confirms that out of the 3 families studied feature overlap is the one that gives the least amount of information on the complexity of datasets. This idea makes sense as feature metrics consider entire features as a whole, completely ignoring local information, making them heavily affected by noisy samples and small disjuncts.

It was also expected that datasets from group I would rank among the top when either of the cleaning algorithms, SMOTE-ENN or MWMOTE were used. These algorithms would address instance and structural overlap present in these datasets which would cause a large performance boost. This was more or less what was observed, while these datasets ranked mostly at the top showing very large performance boosts, there were some among group I which consistently ranked towards the bottom showing almost no increase, like yeast-1_vs_7. When analysing the results of the complexity measures for this dataset before and after preprocessing it is clear to see why, the values of the instance and structural metrics did not decrease nearly enough when compared to the remaining datasets in group I. As such the performance barely changes which is consistent with the results obtained for the classifier.

On the other hand the outlier group ranked among the top datasets showing the highest difference alongside the datasets of group I. However, when analysing these datasets we can see most of them have a very strong structural overlap component that decreases after preprocessing. This would explain why there is such a high performance boost in the first place.

To summarize, the main takeaways from these experiments were:

- (i) Cleaning approaches are very effective at reducing feature overlap since by removing noisy samples from the dataset each feature becomes much more discriminative. On the other hand, oversampling approaches have the opposite effect, where they increase feature overlap;
- (ii) Neighbourhood based approaches are very effective at reducing instance overlap

as stated in [13]. However, they are also surprisingly effective at reducing structural overlap even with algorithms that do not take into account structural properties of the dataset.

This page is intentionally left blank.

Chapter 4

Conclusion

As some works previously have identified, the presence of imbalance in conjunction with other problems like dataset shift and overlap can greatly degrade the performance of data analysis algorithms. This work was particularly motivated on trying to understand the relationship between an algorithm's performance degradation and dataset shift in imbalanced contexts. Furthermore, this study was also motivated on tying to get a better understanding of how the current set of overlap complexity measures behaves in the presence of imbalance.

The experiments done with dataset shift confronts the results obtained in [12], where four cross validation algorithms which induce different degrees of dataset shift are analysed. The same algorithms are tested, but this research expands on the mentioned experiments with a larger dataset pool, analysing different performance metrics, experimenting with oversampling algorithms to measure the impact of imbalance and by backing up the results with dataset complexity measures, providing a more complete coverage.

The results obtained somewhat agree with the ones in [12], where partitioning methods that induce more dataset shift like MSSCV and SCV cause the classification algorithms to obtain worse performances than partitioning algorithms that more actively combat dataset shift such as DBSCV and DOBSCV. Where this study disagrees with the results in [12] is where DOBSCV does not outperform DBSCV like it was expected. Results showed, in fact, that the two algorithms do not appear to have any statistical differences, potentially indicating that DOBSCV does not reduce dataset shift when compared to DBSCV like intended. This idea is supported by the analysis done using the complexity measures, where datasets partitioned with DBSCV and DOBSCV show very similar values between the training and testing partitions, which indicates both combat dataset shift similarly.

Finally, the experiments done with the oversampling algorithms showed that between imbalance and dataset shift, the factor that most degrades the performance of classification algorithms is dataset shift. While, imbalance does cause a degradation in performance in some algorithms such as the C4.5 decision tree and SVM, the kNN and Naive Bayes are not so much affected by the use of an oversampling algorithms, however, dataset shift always degrades an algorithms' performance regardless of the classification algorithm used.

Future work in this area could consist in extending this study for more classifiers similar to what was done in [12] and analysing in more detail the differences between DBSCV and DOBSCV, to better understand why no statistical differences were found between the two algorithms. Furthermore, a new CV algorithm could also be proposed that attempts to decrease dataset shift even further.

The experiments with overlap started by creating a new complexity measure package *py-col* which incorporates multiple complexity measures including the most common ones introduced in [16] to newer ones which were not implemented in any complexity measure package. The package also implements the imbalanced version of some metrics proposed in the literature along with one vs one and one vs all approaches for all metrics.

Experiments were performed using this package in order to validate the taxonomy proposed in [13]. A total of 52 imbalanced datasets from the KEEL repository were selected with varying imbalance ratios. And multiple pre-processing methods were chosen that try to reduce the different types of overlap according to the taxonomy.

These experiments helped in validating the taxonomy proposed when it came to feature overlap metrics as most of them had a similar behaviour. However, an interesting behaviour was seen for the instance and structural metrics that appear to be somewhat connected. It is unclear if these two families are always linked or if it is due to the datasets used. It does make sense that the structural metrics increase when the instance overlap grow as most of them are built on the concept of distance and nearest neighbour approaches which is the same concept the instance metrics are built upon.

A better understanding of the preprocessing algorithms was also obtained. The cleaning approaches which in theory only tackle instance overlap ended up being able to reduce the 3 families studied. Oversampling algorithms seem to work for both instance and structural overlap suggesting once again the connection between these two families. Finally, these experiments confirmed that feature overlap is not the most important out of the families as even in scenarios where it increases, the performance of classification algorithms can increase as well, showing the need for more local approaches rather than global. Eventually the multi resolution metrics could work better for this, but experiments showed that these metrics are very sensitive to parametrization and are not easy to use in scenarios like this one where multiple datasets are tested, as each dataset would need a different set of parameters.

Future work in this area could be done in trying to better understand the parameterization needed for multi resolution metrics. Experiments can also be done testing with measures in the *pycol* package which were not used in this study. Furthermore, more preprocessing algorithms can be experimented with, to further understand the relationship between instance and structural metrics. If these metrics are independent, more datasets should be made available distinguishing between these types of overlap. Even if they are linked there must be an effort to publish datasets where it is made clear what type of overlap is present.

Despite the lack of datasets representative of each overlap family, the *pycol* package can make experimentation much easier now, so that a further understanding of overlap and overlap complexity measures can be obtained.

References

[1] Apple is now worth more than the combined value of the oil and gas majors, https://qz.com/1870823/apple-is-worth-more-than-the-oil-and-gas-majors-combined/. Accessed: 2022-01-22.

[2] Microsoft passes apple to become the world's most valuable company repository https://www.cnbc.com/2021/10/29/microsoft-passes-apple-to-become-the-worlds-most-valuable-company-.html. Accessed: 2022-01-22.

[3] Krzysztof J Cios, Roman W Swiniarski, Witold Pedrycz, and Lukasz A Kurgan. The knowledge discovery process. In *Data Mining*, pages 9–24. Springer, 2007.

[4] Pedro García Laencina, Pedro Henriques Abreu, Miguel Henriques Abreu, and Noémia Afonoso. Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values. *Computers in Biology and Medicine*, 59:125–133, 2015.

[5] Adriana Costa, Miriam Santos, Jastin Soares, and Pedro Henriques Abreu. *Missing Data Imputation via Denoising Autoencoders: The Untold Story: 17th International Symposium, IDA 2018, 's-Hertogenbosch, The Netherlands, October 24–26, 2018, Proceedings*, pages 87–98. 2018.

[6] Vicente García, Roberto Alejo, Josep Sánchez, José Sotoca, and Ramón Mollineda. Combined effects of class imbalance and class overlap on instance-based classification. pages 371–378, 2006.

[7] Misha Denil and Thomas Trappenberg. Overlap versus imbalance. pages 220–231, 2010.

[8] Victor H. Barella, Luís P. F. Garcia, Marcilio P. de Souto, Ana C. Lorena, and André de Carvalho. Data complexity measures for imbalanced classification tasks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[9] Ramón Mollineda, Josep Sánchez, and José Sotoca. A meta-learning framework for pattern classification by means of data complexity measures. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, ISSN 1137-3601, Nº. 29, 2006 (Ejemplar dedicado a: Minería de Datos), pags. 31-38*, 10, 2006.

[10] Victoria López, Alberto Fernández, and Francisco Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1–13, 2014.

[11] Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.

[12] Jose García Moreno-Torres, José A. Saez, and Francisco Herrera. Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1304–1312, 2012.

[13] Miriam Santos, Pedro Henriques Abreu, Nathalie Japkowicz, Alberto Fernández, Carlos Soares, Szymon Wilk, and Joao Santos. On the joint-effect of class imbalance and overlap: a critical review. *Artificial Intelligence Review*, pages 1–69, 03 2022.

[14] Victor H. Barella, Luís P.F. Garcia, Marcilio C.P. de Souto, Ana C. Lorena, and André C.P.L.F. de Carvalho. Assessing the data complexity of imbalanced datasets. *Information Sciences*, 553:83–109, 2021.

[15] Ana C. Lorena, Luís P. F. Garcia, Jens Lehmann, Marcilio C. P. Souto, and Tin Kam Ho. How complex is your classification problem? a survey on measuring classification complexity. *ACM Comput. Surv.*, 52(5), 2019.

[16] Tin Kam Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

[17] Enrique Leyva, Antonio González, and Raúl Pérez. A set of complexity measures designed for applying meta-learning to instance selection. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):354–367, 2015.

[18] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

[19] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

[20] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(35):985–1005, 2007.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[22] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from Imbalanced Data Sets.* 2018.

[23] Jose G. Moreno-Torres, Xavier Llorà, David E. Goldberg, and Rohit Bhargava. Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis. *Information Sciences*, 222:805–823, 2013. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.

[24] Rocío Alaiz-Rodríguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. Improving classification under changes in class and within-class distributions. In Joan Cabestany, Francisco Sandoval, Alberto Prieto, and Juan M. Corchado, editors, *Bio-Inspired Systems: Computational and Ambient Intelligence*, pages 122–130, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[25] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(75):2137–2155, 2009.

[26] Xinchuan Zeng and Tony R. Martinez. Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental & Theoretical Artificial Intelligence*, 12(1):1–12, 2000.

[27] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[28] Vicente García, Ramón Mollineda, Josep Sánchez, Roberto Alejo, and José Sotoca. When overlapping unexpectedly alters the class imbalance effects. pages 499–506, 2007.

[29] Ronaldo C. Prati, Gustavo E. A. P. A. Batista, and Maria Carolina Monard. Class imbalances versus class overlapping: An analysis of a learning system behavior. In *MICAI*, 2004.

[30] Vicente García, R. Mollineda, and José Sánchez. On the k-nn performance in challenging scenario of imbalance and overlapping. *Formal Pattern Analysis Applications*, 11:269–280, 2008.

[31] Vicente García, Ramón Mollineda, Josep Sánchez, Roberto Alejo, and José Sotoca. When overlapping unexpectedly alters the class imbalance effects. pages 499–506, 2007.

[32] L. Cummins. Combining and choosing case base maintenance algorithms. phd thesis. UCC, 2013.

[33] Barnard E Van der Walt, Christiaan M. Measures for the characterization of pattern-recognition data sets. 2007.

[34] José Daniel Pascual-Triana, David Charte, Marta Andrés Arroyo, Alberto Fernández, and Francisco Herrera. Revisiting data complexity metrics based on morphology for overlap and imbalance: Snapshot, new overlap number of balls metrics and singular problems prospect. *CoRR*, abs/2007.07935, 2020.

[35] Christiaan Maarten Van der Walt. Data measures that characterise classification problems. 2008.

[36] Sejong Oh. A new dataset evaluation method based on category overlap. *Computers in Biology and Medicine*, 41(2):115–122, 2011.

[37] Zalán Borsos, Camelia Lemnaru, and Rodica Potolea. Dealing with overlap and imbalance: A new metric and approach. *Pattern Anal. Appl.*, 21(2):381–395, 2018.

[38] Marta Mercier, Miriam Santos, Pedro Henriques Abreu, Carlos Soares, Jastin Soares, and Joao Santos. *Analysing the Footprint of Classifiers in Overlapped and Imbalanced Contexts: 17th International Symposium, IDA 2018, 's-Hertogenbosch, The Netherlands, October 24–26, 2018, Proceedings*, pages 200–212. 2018.

[39] Chris Thornton. Separability is a learner's best friend. In John A. Bullinaria, David W. Glasspool, and George Houghton, editors, *4th Neural Computation and Psychology Workshop, London, 9–11 April 1997*, pages 40–46, London, 1998. Springer London.

[40] John Greene. Feature subset selection using thornton's separability index and its applicability to a number of sparse proximity-based classifiers. 2001.

[41] Martinez Tony Smith, Michael R. An instance level analysis of data complexity. pages 225–256, 2014.

[42] José Sotoca, Ramón Mollineda, and Josep Sánchez. A meta-learning framework for pattern classification by means of data complexity measures. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 10:31–38, 2006.

[43] Jerzy Napierala, Krystyna Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46:563–597, 2016.

[44] Jerzy Wilk S. Napierala, Krystyna Stefanowski. Learning from imbalanced data in presence of noisy and borderline examples. page 158–167, 2010.

[45] G. Armano and E. Tamponi. Experimenting multiresolution analysis for identifying regions of different classification complexity. *Pattern Anal. Appl.*, 19(1):129–137, 2016.

[46] Sameer Singh. Multiresolution estimates of classification complexity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25:1534 – 1539, 2004.

[47] S. Singh. Prism – a novel framework for pattern recognition. *Pattern Analysis and Applications*, 6:134–149, 2003.

[48] Miriam Seoane Santos, Pedro Henriques Abreu, Szymon Wilk, and João Santos. How distance metrics influence missing data imputation with k-nearest neighbours. *Pattern Recognition Letters*, 136:111–119, 2020.

[49] Artificial dataset generator repository, https://github.com/sysmon37/datagenerator. Accessed: 2022-01-22.

[50] Pattaramon Vuttipittayamongkol and Eyad Elyan. Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. *Information Sciences*, 509:47–70, 2020.

[51] Sukarna Barua, Md. Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2014.

This page is intentionally left blank.

Appendix A

Dataset shift

Exploratory Analysis Figures

This section contains the remaining figures used for exploratory analysis for the SVM and Naive Bayes classifier, analysing both the impact of the cross validation and preprocessing method chosen.

Figure A.1 and Figure A.2 show the box plots of the oversampling techniques vs performance for these two classifiers. Similarly, Figure A.3 and Figure A.4 show the box plots for the cross validation algorithms vs performance.

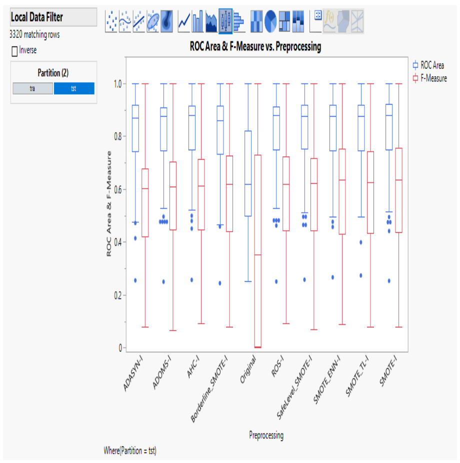


Figure A.1: Oversampling Algorithm vs Performance for SVM

Similarly to what was shown in the results section, depending on the classifier the reducing the imbalance through the use of an oversampling method has different effects, sometimes increasing the performance, like the case of the SVM, and in the case of the Naive Bayes the results are inconsistent.

On the other hand, when we observe the cross validation methods (Figure A.3 and Figure A.4), we see the same effect shown before. Where datasets split with MSSCV cause the worse performance for the classifiers and DBSCV and DOBSCV are the best performing out of the four tested.

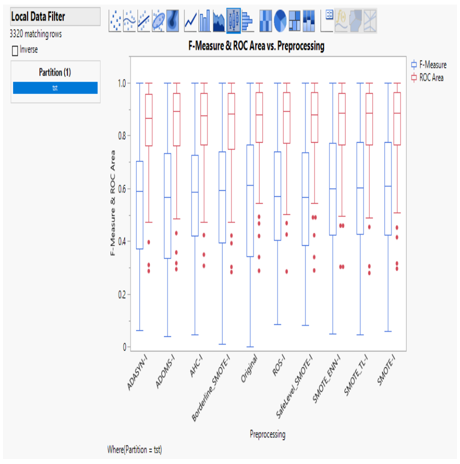


Figure A.2: Oversampling Algorithm vs Performance for Naive Bayes

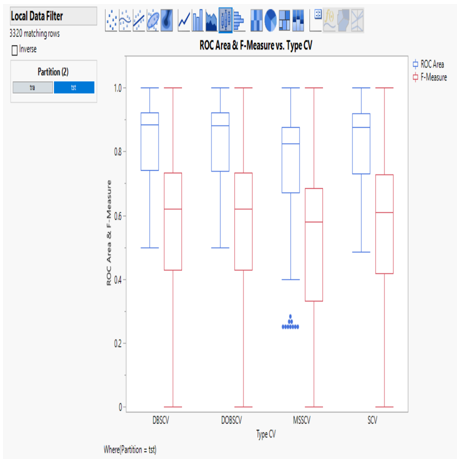


Figure A.3: Cross Validation Algorithm vs Performance for SVM

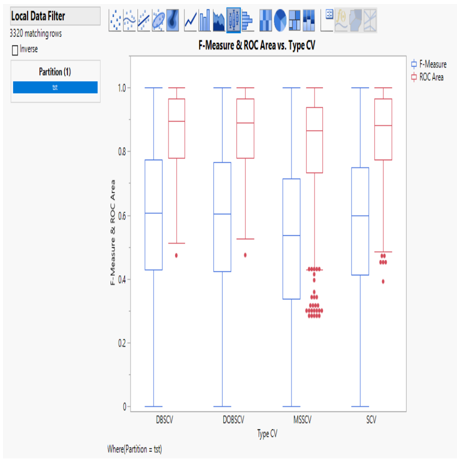


Figure A.4: Cross Validation vs Performance for Naive Bayes

Statistical Analysis Tables

In this section, the remaining tables used for the statistical analysis of dataset shift are show, Tables A.1-A.4 show the averages for the F-Measure, Precision, Recall and AUC, for all the classification algorithms. MSSCV is highlighted, showing consistently worse performances, like how it was described in the Results and Discussion Section.

Algorithm	CV	F-Measure	Precision	Recall	AUC
ADASYN-I	DBSCV	0.5642	0.4981	0.7948	0.8661
	DOBSCV	0.5606	0.4972	0.7899	0.865
	MSSCV	0.5154	0.5018	0.6958	0.8161
	SCV	0.5532	0.4921	0.7784	0.8569
ADOMS-I	DBSCV	0.5494	0.4855	0.8067	0.871
	DOBSCV	0.547	0.4851	0.8019	0.8699
	MSSCV	0.5021	0.4915	0.727	0.8157
	SCV	0.538	0.48	0.7885	0.8618
AHC-I	DBSCV	0.5748	0.5033	0.8037	0.8693
	DOBSCV	0.573	0.5029	0.8004	0.868
	MSSCV	0.524	0.5125	0.7112	0.817
	SCV	0.565	0.4979	0.7897	0.8595
Borderline_SMOTE-I	DBSCV	0.5791	0.5414	0.7355	0.8569
	DOBSCV	0.5747	0.5381	0.7289	0.856
	MSSCV	0.5159	0.52	0.6496	0.8085
	SCV	0.5656	0.5278	0.7209	0.8478
Original	DBSCV	0.5812	0.5931	0.6496	0.8741
	DOBSCV	0.5781	0.5892	0.6463	0.8732
	MSSCV	0.4994	0.5519	0.5698	0.82
	SCV	0.5681	0.5835	0.6362	0.8653
ROS-I	DBSCV	0.5678	0.4867	0.8178	0.8756
	DOBSCV	0.567	0.4876	0.815	0.8744
	MSSCV	0.5244	0.5093	0.7184	0.8209
	SCV	0.5593	0.4837	0.803	0.8648
SMOTE-I	DBSCV	0.598	0.5393	0.805	0.8726
	DOBSCV	0.5946	0.5376	0.7993	0.8712
	MSSCV	0.5322	0.5294	0.7024	0.8196
	SCV	0.5835	0.5291	0.7846	0.8625
SMOTE_ENN-I	DBSCV	0.5987	0.5498	0.7988	0.8732
	DOBSCV	0.5954	0.5496	0.7928	0.8717
	MSSCV	0.531	0.5319	0.6922	0.8205
	SCV	0.5839	0.54	0.7774	0.8624
SMOTE_TL-I	DBSCV	0.5952	0.5371	0.8143	0.8728
	DOBSCV	0.5918	0.5371	0.8082	0.8715
	MSSCV	0.5359	0.5265	0.7154	0.8213
	SCV	0.5818	0.5283	0.7948	0.863
Safe-Level_SMOTE-I	DBSCV	0.5666	0.4831	0.8228	0.8744
	DOBSCV	0.5655	0.483	0.8193	0.8734
	MSSCV	0.5266	0.5057	0.7336	0.8223
	SCV	0.5598	0.4811	0.8098	0.8657

Table A.1: Performance Measures for Naive Bayes

Algorithm	CV	F-Measure	Precision	Recall	AUC
ADASYN-I	DBSCV	0.5642	0.4981	0.7948	0.8661
	DOBSCV	0.5606	0.4972	0.7899	0.865
	MSSCV	0.5154	0.5018	0.6958	0.8161
	SCV	0.5532	0.4921	0.7784	0.8569
ADOMS-I	DBSCV	0.5494	0.4855	0.8067	0.871
	DOBSCV	0.547	0.4851	0.8019	0.8699
	MSSCV	0.5021	0.4915	0.727	0.8157
	SCV	0.538	0.48	0.7885	0.8618
AHC-I	DBSCV	0.5748	0.5033	0.8037	0.8693
	DOBSCV	0.573	0.5029	0.8004	0.868
	MSSCV	0.524	0.5125	0.7112	0.817
	SCV	0.565	0.4979	0.7897	0.8595
Borderline_SMOTE-I	DBSCV	0.5791	0.5414	0.7355	0.8569
	DOBSCV	0.5747	0.5381	0.7289	0.856
	MSSCV	0.5159	0.52	0.6496	0.8085
	SCV	0.5656	0.5278	0.7209	0.8478
Original	DBSCV	0.5812	0.5931	0.6496	0.8741
	DOBSCV	0.5781	0.5892	0.6463	0.8732
	MSSCV	0.4994	0.5519	0.5698	0.82
	SCV	0.5681	0.5835	0.6362	0.8653
ROS-I	DBSCV	0.5678	0.4867	0.8178	0.8756
	DOBSCV	0.567	0.4876	0.815	0.8744
	MSSCV	0.5244	0.5093	0.7184	0.8209
	SCV	0.5593	0.4837	0.803	0.8648
SMOTE-I	DBSCV	0.598	0.5393	0.805	0.8726
	DOBSCV	0.5946	0.5376	0.7993	0.8712
	MSSCV	0.5322	0.5294	0.7024	0.8196
	SCV	0.5835	0.5291	0.7846	0.8625
SMOTE_ENN-I	DBSCV	0.5987	0.5498	0.7988	0.8732
	DOBSCV	0.5954	0.5496	0.7928	0.8717
	MSSCV	0.531	0.5319	0.6922	0.8205
	SCV	0.5839	0.54	0.7774	0.8624
SMOTE_TL-I	DBSCV	0.5952	0.5371	0.8143	0.8728
	DOBSCV	0.5918	0.5371	0.8082	0.8715
	MSSCV	0.5359	0.5265	0.7154	0.8213
	SCV	0.5818	0.5283	0.7948	0.863
Safe-Level_SMOTE-I	DBSCV	0.5666	0.4831	0.8228	0.8744
	DOBSCV	0.5655	0.483	0.8193	0.8734
	MSSCV	0.5266	0.5057	0.7336	0.8223
	SCV	0.5598	0.4811	0.8098	0.8657

Table A.2: Performance Measures for kNN

Algorithm	CV	F-Measure	Precision	Recall	AUC
ADASYN-I	DBSCV	0.5856	0.4849	0.8684	0.8498
	DOBSCV	0.5839	0.4845	0.8649	0.8477
	MSSCV	0.5421	0.4919	0.7852	0.7828
	SCV	0.5802	0.485	0.8568	0.8424
ADOMS-I	DBSCV	0.5901	0.4982	0.848	0.8479
	DOBSCV	0.5909	0.5002	0.8471	0.8475
	MSSCV	0.5461	0.5039	0.7722	0.781
	SCV	0.5843	0.4979	0.837	0.8411
AHC-I	DBSCV	0.6006	0.5086	0.8505	0.8514
	DOBSCV	0.6016	0.5095	0.851	0.8519
	MSSCV	0.5574	0.5136	0.7794	0.7901
	SCV	0.5957	0.5084	0.8427	0.8461
Borderline_SMOTE-I	DBSCV	0.6105	0.5326	0.8271	0.843
	DOBSCV	0.6117	0.5352	0.8247	0.8425
	MSSCV	0.5467	0.5123	0.7439	0.7745
	SCV	0.603	0.5294	0.813	0.8356
Original	DBSCV	0.4215	0.5819	0.3686	0.678
	DOBSCV	0.4187	0.5735	0.3678	0.6773
	MSSCV	0.3402	0.418	0.3376	0.6478
	SCV	0.4092	0.5583	0.3613	0.6734
ROS-I	DBSCV	0.606	0.5177	0.855	0.852
	DOBSCV	0.6051	0.5171	0.8521	0.8512
	MSSCV	0.5508	0.5084	0.7741	0.7855
	SCV	0.5978	0.5135	0.8425	0.8448
SMOTE-I	DBSCV	0.6149	0.5303	0.8573	0.854
	DOBSCV	0.6147	0.5315	0.8554	0.8529
	MSSCV	0.5614	0.5211	0.7786	0.7893
	SCV	0.6085	0.5289	0.8463	0.847
SMOTE_ENN-I	DBSCV	0.614	0.5309	0.8453	0.8512
	DOBSCV	0.6141	0.5323	0.844	0.8506
	MSSCV	0.5547	0.5183	0.7657	0.7853
	SCV	0.6077	0.5295	0.8351	0.8449
SMOTE_TL-I	DBSCV	0.604	0.507	0.8848	0.8505
	DOBSCV	0.6035	0.5077	0.8814	0.8491
	MSSCV	0.559	0.5075	0.8025	0.7886
	SCV	0.599	0.507	0.8716	0.8442
Safe-Level_SMOTE-I	DBSCV	0.6055	0.5142	0.8588	0.8538
	DOBSCV	0.6048	0.5146	0.8568	0.8526
	MSSCV	0.5531	0.5098	0.7763	0.7867
	SCV	0.5991	0.5139	0.8467	0.8464

Table A.3: Performance Measures for SVM

Algorithm	CV	F-Measure	Precision	Recall	AUC
ADASYN-I	DBSCV	0.5856	0.4849	0.8684	0.8498
	DOBSCV	0.5839	0.4845	0.8649	0.8477
	MSSCV	0.5421	0.4919	0.7852	0.7828
	SCV	0.5802	0.485	0.8568	0.8424
ADOMS-I	DBSCV	0.5901	0.4982	0.848	0.8479
	DOBSCV	0.5909	0.5002	0.8471	0.8475
	MSSCV	0.5461	0.5039	0.7722	0.781
	SCV	0.5843	0.4979	0.837	0.8411
AHC-I	DBSCV	0.6006	0.5086	0.8505	0.8514
	DOBSCV	0.6016	0.5095	0.851	0.8519
	MSSCV	0.5574	0.5136	0.7794	0.7901
	SCV	0.5957	0.5084	0.8427	0.8461
Borderline_SMOTE-I	DBSCV	0.6105	0.5326	0.8271	0.843
	DOBSCV	0.6117	0.5352	0.8247	0.8425
	MSSCV	0.5467	0.5123	0.7439	0.7745
	SCV	0.603	0.5294	0.813	0.8356
Original	DBSCV	0.4215	0.5819	0.3686	0.678
	DOBSCV	0.4187	0.5735	0.3678	0.6773
	MSSCV	0.3402	0.418	0.3376	0.6478
	SCV	0.4092	0.5583	0.3613	0.6734
ROS-I	DBSCV	0.606	0.5177	0.855	0.852
	DOBSCV	0.6051	0.5171	0.8521	0.8512
	MSSCV	0.5508	0.5084	0.7741	0.7855
	SCV	0.5978	0.5135	0.8425	0.8448
SMOTE-I	DBSCV	0.6149	0.5303	0.8573	0.854
	DOBSCV	0.6147	0.5315	0.8554	0.8529
	MSSCV	0.5614	0.5211	0.7786	0.7893
	SCV	0.6085	0.5289	0.8463	0.847
SMOTE_ENN-I	DBSCV	0.614	0.5309	0.8453	0.8512
	DOBSCV	0.6141	0.5323	0.844	0.8506
	MSSCV	0.5547	0.5183	0.7657	0.7853
	SCV	0.6077	0.5295	0.8351	0.8449
SMOTE_TL-I	DBSCV	0.604	0.507	0.8848	0.8505
	DOBSCV	0.6035	0.5077	0.8814	0.8491
	MSSCV	0.559	0.5075	0.8025	0.7886
	SCV	0.599	0.507	0.8716	0.8442
Safe-Level_SMOTE-I	DBSCV	0.6055	0.5142	0.8588	0.8538
	DOBSCV	0.6048	0.5146	0.8568	0.8526
	MSSCV	0.5531	0.5098	0.7763	0.7867
	SCV	0.5991	0.5139	0.8467	0.8464

Table A.4: Performance Measures for C4.5

The following table shows the results from the non-parametric Friedman test done for the 4 cross validation algorithms, for every oversampling algorithm as well as the original datasets. The tables show if the test was below or above the threshold of 0.05 significance level:

Algorithm	F Value	Lower Thres.
ADASYN-I	162.3151	Yes
ADOMS-I	138.4347	Yes
AHC-I	147.2931	Yes
Borderline_SMOTE-I	149.8662	Yes
Original	108.859	Yes
ROS-I	172.039	Yes
SMOTE-I	167.5986	Yes
SMOTE_ENN-I	162.9456	Yes
SMOTE_TL-I	150.0558	Yes
Safe-Level_SMOTE-I	161.719	Yes

Table A.5: Results of the Friedman test comparing the CV Algorithms for each Oversampling Algorithm

The following tables show the post hoc tests done for the C4.5 Decision tree, for every pre-processing algorithm used, the DBSCV and DOBSCV algorithms are highlighted to show that among the CV methods these two were always statistical similar:

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.4159**	0.001	0.001
DOBSCV	**0.4159**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.6: Post Hoc test results for ADASYN-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.9**	0.001	0.001
DOBSCV	**0.9**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.7: Post Hoc test results for ADOMS-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.734**	0.001	0.001
DOBSCV	**0.734**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.8: Post Hoc test results for AHC-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.6421**	0.001	0.001
DOBSCV	**0.6421**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.9: Post Hoc test results for Borderline SMOTE-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.3964**	0.001	0.001
DOBSCV	**0.3964**	1.0	0.001	0.0042
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.0042	0.001	1.0

Table A.10: Post Hoc test results for the original datasets

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.3403**	0.001	0.001
DOBSCV	**0.3403**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.11: Post Hoc test results for ROS-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.4948**	0.001	0.001
DOBSCV	**0.4948**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.12: Post Hoc test results for SMOTE-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.5502**	0.001	0.001
DOBSCV	**0.5502**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.13: Post Hoc test results for SMOTE-ENN-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.7156**	0.001	0.001
DOBSCV	**0.715**6	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.14: Post Hoc test results for SMOTE-TL-I

	DBSCV	DOBSCV	SCV	MSSCV
DBSCV	1.0	**0.2733**	0.001	0.001
DOBSCV	**0.2733**	1.0	0.001	0.001
MSSCV	0.001	0.001	1.0	0.001
SCV	0.001	0.001	0.001	1.0

Table A.15: Post Hoc test results for Safe-Level-SMOTE-I

Appendix B

Metric Validation

In this section, the validation process of the complexity measures will be detailed. with that goal in mind, three of the developed measures will be shown. These measures serve as an example for the process of validation which was performed for all measures.

T1

As stated, before, the T1 measure defines boundaries around each sample in the dataset by finding its nearest enemy and drawing hyperspheres around them. After that process is complete, the hyperspheres which are completely contained inside another are removed.

Figure B.1 shows a simple example of the hyperspheres that would be created, before the removal processes.

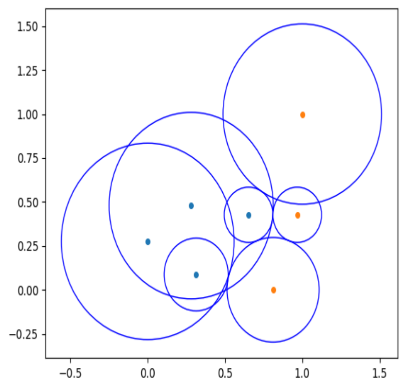


Figure B.1: Simple T1 test

In this example, there are a total of 7 samples which each generate their hypersphere, out of all of them only one is completely contained inside another, which should be removed. Figure B.2 shows the effect of removing the overlapped spheres.

Out of a total of 7 samples, 6 hyperspheres remain after removal, as such using the formula described previously, T1 should be equal to 6/7 which was the value obtained by the pycol package.

Another example can be seen in B.3, in this example no circles are completely contained

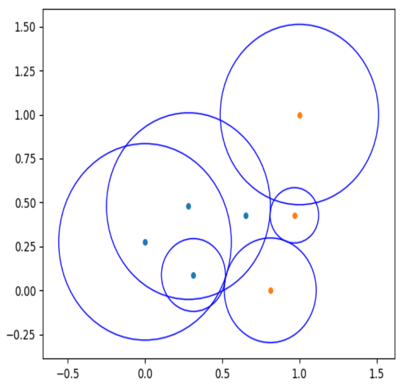


Figure B.2: Simple T1 test, after removal

inside another, as such the none is removed contrary to what happened in the previous example. The expected value for pycol in this case is 1.0, since every sample has a circle surrounding it, which was the value obtained.

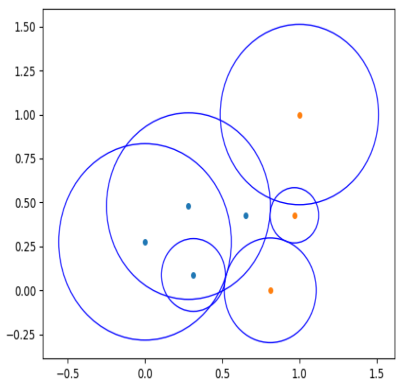


Figure B.3: Second T1 test

Augmented R-Value

The R value measure is a distance based measure which uses the nearest neighbours of all samples to determine how many are in the overlapped region. The parameters picked for this test is a neighbourhood of k=3 and a threshold of 1. In other words, if more than 1 neighbour out of the 3 nearest ones is of a different class, that sample is, according to this measure, in the overlapped region.

Figure B.4 shows an example of dataset used for this test. Out of the seven samples present, four of them are in the overlapped region, according to the parameters describes previously. The three orange samples, and the leftmost blue sample.

As such, according to equation 2.35 the $r(C_{min}, C_{maj})$ is 3/3 and $r(C_{maj}, C_{min})$ is 1/4. Considering the imbalance ratio which is 0.75, the augmented r value according to 2.34 should be $1/(1 + 0.75) * (1/4 + 0.75 * 1) = 0.5714$, which is precisely the value obtained by the pycol package.

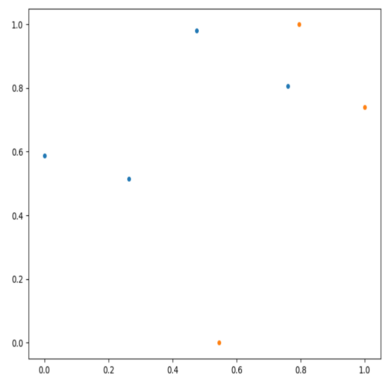


Figure B.4: Simple R Value test

Following the same logic, we can look at the second example in figure B.5, presented below:

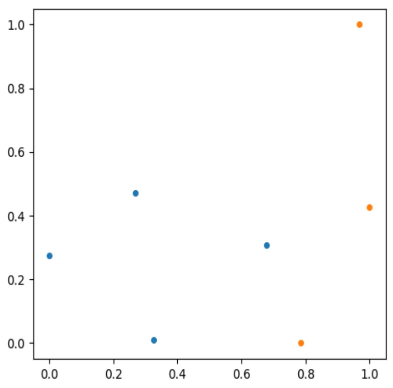


Figure B.5: Second simple R Value test

In this case, the imbalance ratio and the number of samples is the same, if the same parameters for k and theta are maintained, the same number of samples for blue are in the overlapped region, however this time only two of the orange samples are in the overlapped. Following the previously mentioned equations, $r(C_{min}, C_{maj})$ is $2/3$ and $r(C_{maj}, C_{min})$ is $1/4$. As such, the augmented r value for this dataset is $1/(1 + 0.75) * (1/4 + 0.75 * 2/3) = 0.4285$, which is the value obtained by the package.

Purity

For the purity test, we first start by looking at if the divisions made at each resolution are being done correctly, for the sake of this example the dataset used will be a very small one, and the resolution will only go up to 2, which generates nine hypercubes. The results for the resolution of 0 to 2 can be seen in Figures B.6-B.7.

As it is possible to see at each point, the feature space is being split into evenly spaced areas in every dimension, starting with a resolution of 0, where there is no partitioning, and ending with a resolution of 2 where there are 9 cells present.

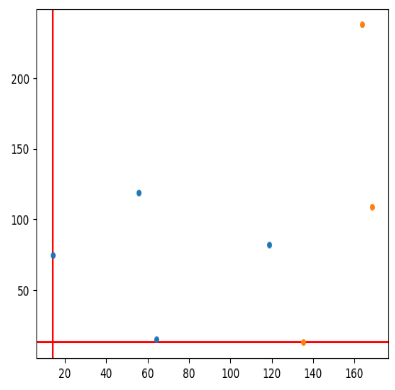


Figure B.6: Purity test with resolution=0

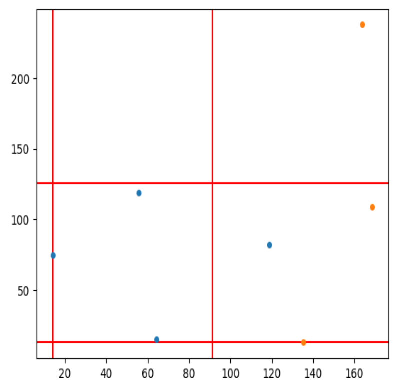


Figure B.7: Purity test with resolution=1

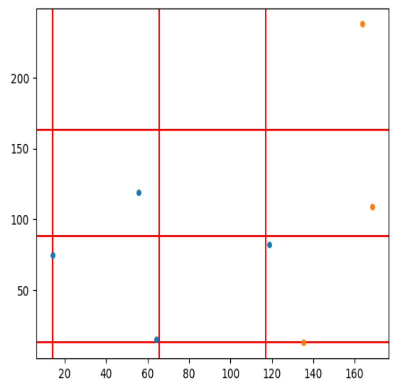


Figure B.8: Purity test with resolution=2

Afterwards, each sample is assigned the cell it is in and a class count for each cell is made. To make this process easier, we assign labels to each cell: Cell 0-0 is the leftmost and downmost, cell 1-0 is the cell directly to the right of 0-0 and 0-1 is the one directly above.

As the figures show, for the resolution of 0, cell 0-0 (the only existing cell) should have the following class counts, 4 blue samples and 3 orange ones, which constitutes the whole dataset. For the resolution of 1, cell 0-0 should have 3 instances of the blue class, cell 1-0 should have 2 of the orange class and 1 of the blue, cell 1-1 should have 1 of the orange, while cell 0-1 should be empty. Finally, for the resolution of 3, cell 0-0 should have 2 of the blue class, 2-0 should have one of each class, cell 0-1 one of the blue class, cells 2-1 and 2-2 one of the orange class, with the remaining cell having none samples. For all resolutions, these were the assignments made by the package.

The next step would be to calculate the purity of each cell according to the equation 2.58. To not extend this example too much further, it will only focus on the resolution of 2. Cells 0-0, 0-1 2-1 and 2-2 should have a purity of 1 (all the samples are the same) while cell 2-0 should have a purity of 0 (has an equal number of samples from each class), which were the values obtained by the package.

These values are averaged taking into account the number of samples in each cell, giving the overall purity values for this resolution of 0.71428, which when weighted according to equation 2.60 gives a weighted purity of 0.17857.

Appendix C

Taxonomy Validation

This section shows the results for the taxonomy validation that were left out of the overlap section. Starting by showing the complexity metrics before and after the use of NB-Tomek and then showing the results of the Random Forest algorithm before and after NB-Tomek was performed. After that the results for the outlier group are shown before and after preprocessing is applied.

NB-Tomek Comparison

dataset	preproc	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
glass-0-1-4-6_vs_2	original	0.9167	0.6667	0.1264	0.0833	0.9167	0.6362	0.4773	0.7568	0.3632	0.5417	0.0000
glass-0-1-4-6_vs_2	NB-Basic	0.4167	0.0833	0.1778	0.1222	0.7500	0.5286	0.1955	0.5517	0.2757	0.4111	0.0000
yeast-1-4-5-8_vs_7	original	0.8571	0.9048	0.0807	0.0494	0.9524	0.6108	0.4900	0.8506	0.5775	0.8313	0.5617
yeast-1-4-5-8_vs_7	NB-Basic	0.4286	0.3810	0.0840	0.0605	0.5714	0.4745	0.3023	0.7237	0.3560	0.7402	0.3523
glass-0-1-6_vs_2	original	0.8333	0.8333	0.1526	0.1037	1.0000	0.5859	0.4736	0.7793	0.3439	0.5704	0.0963
glass-0-1-6_vs_2	NB-Basic	0.4167	0.0000	0.1831	0.1084	0.8333	0.4546	0.1673	0.6003	0.2139	0.4940	0.0000
glass-0-1-5_vs_2	original	0.8333	0.6667	0.1719	0.0992	1.0000	0.6369	0.4855	0.8787	0.3863	0.4959	0.0826
glass-0-1-5_vs_2	NB-Basic	0.5000	0.0833	0.2521	0.1370	0.9167	0.5318	0.2240	0.7563	0.1827	0.3562	0.0000
glass2	original	0.6667	0.6667	0.1333	0.0867	0.8333	0.5877	0.4438	0.7524	0.3892	0.5267	0.0600
glass2	NB-Basic	0.3333	0.0000	0.1579	0.1368	0.5833	0.5053	0.1792	0.5501	0.2249	0.4316	0.0000
yeast-1_vs_7	original	0.7143	0.5714	0.1050	0.0652	0.7619	0.5662	0.4629	0.7116	0.2786	0.7671	0.4627
yeast-1_vs_7	NB-Basic	0.4762	0.2857	0.1010	0.0765	0.6190	0.4649	0.3429	0.6393	0.1626	0.6429	0.2704
yeast-0-3-5-9_vs_7-8	original	0.5429	0.5429	0.1442	0.1014	0.8286	0.5186	0.4069	0.7806	0.2968	0.8310	0.5944

Table C.1: Complexity measure values before and after NB-Tomek for the group I datasets

dataset	preproc	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
pima	original	0.4947	0.4149	0.3193	0.2658	0.6117	0.4966	0.4447	0.6568	0.3463	0.9926	0.9591
pima	NB-Tomek	0.0372	0.0160	0.1196	0.0959	0.1117	0.3379	0.2389	0.3447	0.1751	0.5388	0.3105
yeast-0-5-6-7-9_vs_4	original	0.4722	0.4722	0.1086	0.0838	0.6944	0.5041	0.3460	0.4329	0.1987	0.9189	0.6973
yeast-0-5-6-7-9_vs_4	NB-Tomek	0.1111	0.1111	0.0565	0.0471	0.1944	0.3775	0.1547	0.3063	0.0893	0.8863	0.5569
ecoli-0-1-4-7_vs_2-3-5-6	original	0.4286	0.1905	0.0703	0.0339	0.4762	0.4448	0.2968	0.6395	0.1005	0.8856	0.6822
ecoli-0-1-4-7_vs_2-3-5-6	NB-Tomek	0.1905	0.2381	0.0531	0.0339	0.3333	0.3543	0.2193	0.5569	0.0845	0.8757	0.5989
yeast-0-2-5-6_vs_3-7-8-9	original	0.3286	0.5714	0.1048	0.0625	0.5571	0.4287	0.3108	0.6073	0.1887	0.8935	0.7898
yeast-0-2-5-6_vs_3-7-8-9	NB-Tomek	0.1429	0.3000	0.0794	0.0553	0.2857	0.2801	0.1641	0.5020	0.1482	0.8266	0.4950
yeast-0-2-5-7-9_vs_3-6-8	original	0.2143	0.2429	0.0511	0.0327	0.3429	0.3771	0.1649	0.3675	0.1126	0.8707	0.7486
yeast-0-2-5-7-9_vs_3-6-8	NB-Tomek	0.0571	0.2000	0.0227	0.0195	0.1143	0.2580	0.0753	0.3293	0.0923	0.8425	0.6602

Table C.2: Complexity measure values before and after NB-Tomek for the group II datasets

NB-Tomek Performance

dataset	diff	original	NB-Tomek	Dataset	Difference	original	NB-Tomek
glass-0-1-6_vs_5	0.5000	0.0222	0.5222	glass6	0.0000	0.9333	0.9333
cleveland-0_vs_4	0.4833	0.0167	0.5000	iris0	0.0000	1.0000	1.0000
glass-0-1-4-6_vs_2	0.4677	0.0000	0.4677	shuttle-c2-vs-c4	0.0000	1.0000	1.0000
glass-0-1-6_vs_2	0.4197	0.0222	0.4419	**yeast-1-4-5-8_vs_7**	0.0000	0.0000	0.0000
glass2	0.3863	0.0889	0.4752	ecoli-0-3-4-6_vs_5	-0.0017	0.7883	0.7866
glass-0-1-5_vs_2	0.2702	0.0000	0.2702	ecoli-0-6-7_vs_3-5	-0.0078	0.8298	0.8220
ecoli-0-1-3-7_vs_2-6	0.2444	0.5778	0.8222	ecoli-0-2-6-7_vs_3-5	-0.0098	0.8727	0.8629
glass5	0.1889	0.4000	0.5889	**yeast-1_vs_7**	-0.0197	0.4949	0.4752
ecoli3	0.1260	0.4587	0.5847	glass0	-0.0221	0.7171	0.6950
yeast-0-5-6-7-9_vs_4	0.1177	0.3446	0.4623	yeast-2_vs_4	-0.0248	0.8310	0.8062
yeast3	0.0897	0.6719	0.7616	ecoli-0-1_vs_2-3-5	-0.0255	0.6925	0.6670
newthyroid2	0.0733	0.8947	0.9680	yeast-0-2-5-7-9_vs_3-6-8	-0.0462	0.8532	0.8070
glass-0-1-2-3_vs_4-5-6	0.0638	0.8552	0.9191	ecoli-0_vs_1	-0.0492	0.9880	0.9388
ecoli1	0.0562	0.7398	0.7960	ecoli-0-1-4-7_vs_2-3-5-6	-0.0519	0.8983	0.8464
yeast-0-2-5-6_vs_3-7-8-9	0.0476	0.5845	0.6321	vehicle3	-0.0529	0.5390	0.4861
new-thyroid1	0.0435	0.9474	0.9909	ecoli-0-3-4_vs_5	-0.0551	0.9066	0.8515
ecoli2	0.0371	0.8119	0.8490	yeast-0-3-5-9_vs_7-8	-0.0632	0.4626	0.3994
glass-0-4_vs_5	0.0333	0.9667	1.0000	glass4	-0.0730	0.7302	0.6571
ecoli-0-1_vs_5	0.0333	0.8146	0.8480	pima	-0.0737	0.6512	0.5775
ecoli-0-1-4-7_vs_5-6	0.0326	0.6415	0.6741	vowel0	-0.1067	0.9824	0.8757
ecoli-0-1-4-6_vs_5	0.0323	0.6970	0.7293	vehicle0	-0.1227	0.9326	0.8099
vehicle1	0.0221	0.5114	0.5336	glass1	-0.1486	0.7201	0.5714
yeast-2_vs_8	0.0177	0.4786	0.4963	ecoli4	-0.1489	0.8848	0.7360
ecoli-0-4-6_vs_5	0.0036	0.9055	0.9091	ecoli-0-3-4-7_vs_5-6	-0.1886	0.9974	0.8088
ecoli-0-6-7_vs_5	0.0036	0.8322	0.8359	page-blocks-1-3_vs_4	-0.3535	0.9441	0.5906
glass-0-6_vs_5	0.0000	1.0000	1.0000	vehicle2	-0.9417	0.9914	0.0496

Table C.3: Performance of Random Forest before and after preprocessing using NB-Tomek. The datasets from group I are marked in bold

Outlier group Comparison

dataset	preproc	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
cleveland-0_vs_4	orig	0.4000	0.9000	0.0704	0.0720	0.5000	0.5052	0.5109	0.3346	0.0535	0.7760	0.0000
cleveland-0_vs_4	NB-Basic	0.2000	0.5000	0.0531	0.0313	0.3000	0.2582	0.2778	0.2377	0.0285	0.6719	0.0000
ecoli-0-1-3-7_vs_2-6	orig	0.4000	0.4000	0.0386	0.0355	0.6000	0.3609	0.3260	0.3845	0.0395	0.2081	0.0000
ecoli-0-1-3-7_vs_2-6	NB-Basic	0.4000	0.0000	0.0224	0.0132	0.3000	0.1114	0.2170	0.2775	0.0313	0.1316	0.0000
glass-0-1-6_vs_5	orig	0.1429	0.1429	0.0477	0.0308	0.7143	0.4232	0.2427	0.3819	0.1542	0.1462	0.0000
glass-0-1-6_vs_5	NB-Basic	0.0000	0.1429	0.0175	0.0000	0.1429	0.2666	0.1637	0.2501	0.0197	0.0777	0.0000

Table C.4: Complexity measure values before and after NB-Basic for the outlier group datasets

dataset	preproc	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
glass-0-1-6_vs_5	orig	0.1429	0.1429	0.0477	0.0308	0.7143	0.4232	0.2427	0.3819	0.1542	0.1462	0.0000
glass-0-1-6_vs_5	MWMOTE	0.0000	0.0033	0.0191	0.0203	0.0325	0.0609	0.0449	0.3819	0.1161	0.5486	0.0000
glass5	orig	0.4286	0.1429	0.0517	0.0397	0.5714	0.4898	0.3065	0.4328	0.1054	0.1391	0.0000
glass5	MWMOTE	0.0000	0.0019	0.0284	0.0313	0.0481	0.0742	0.0339	0.3651	0.0935	0.5309	0.0000
cleveland-0_vs_4	orig	0.4000	0.9000	0.0704	0.0720	0.5000	0.5052	0.5109	0.3346	0.0535	0.7760	0.0000
cleveland-0_vs_4	MWMOTE	0.0000	0.1864	0.0450	0.0493	0.0435	0.1388	0.2633	0.2543	0.0476	0.8783	0.0000
ecoli3	orig	0.6000	0.1600	0.1102	0.0975	0.6800	0.5684	0.3698	0.3661	0.1623	0.4449	0.2161
ecoli3	MWMOTE	0.0000	0.2686	0.0733	0.0783	0.3684	0.0108	0.0869	0.3680	0.1293	0.6896	0.5047

Table C.5: Complexity measure values before and after MWMOTE for the outlier group datasets

dataset	**preproc**	**N3**	**N4**	**kDN**	**AugR**	**N1**	**N2**	**ONB**	**F1**	**F1v**	**F3**	**F4**
glass-0-1-6_vs_5	orig	0.1429	0.1429	0.0477	0.0308	0.7143	0.4232	0.2427	0.3819	0.1542	0.1462	0.0000
glass-0-1-6_vs_5	SMOTE-ENN	0.0000	0.0000	0.0187	0.0203	0.0325	0.0472	0.0447	0.3819	0.1286	0.5488	0.0000
glass5	orig	0.4286	0.1429	0.0517	0.0397	0.5714	0.4898	0.3065	0.4328	0.1054	0.1391	0.0000
glass5	SMOTE-ENN	0.0000	0.0000	0.0285	0.0313	0.0486	0.0699	0.0382	0.3907	0.0993	0.5347	0.0000
cleveland-0_vs_4	orig	0.4000	0.9000	0.0704	0.0720	0.5000	0.5052	0.5109	0.3346	0.0535	0.7760	0.0000
cleveland-0_vs_4	SMOTE-ENN	0.0000	0.2347	0.0385	0.0376	0.0510	0.0920	0.1431	0.2851	0.0449	0.8592	0.0000
ecoli3	orig	0.6000	0.1600	0.1102	0.0975	0.6800	0.5684	0.3698	0.3661	0.1623	0.4449	0.2161
ecoli3	SMOTE-ENN	0.0000	0.0962	0.0754	0.0811	0.0913	0.1247	0.0883	0.3518	0.1130	0.6874	0.5036

Table C.6: Complexity measure values before and after SMOTE-ENN for the outlier group datasets

dataset	preproc	N3	N4	kDN	AugR	N1	N2	ONB	F1	F1v	F3	F4
glass-0-1-6_vs_5	orig	0.1429	0.1429	0.0477	0.0308	0.7143	0.4232	0.2427	0.3819	0.1542	0.1462	0.0000
glass-0-1-6_vs_5	NB-Tomek	0.0000	0.4286	0.0353	0.0252	0.3286	0.2905	0.1741	0.3520	0.1058	0.1345	0.0000
glass5	orig	0.4286	0.1429	0.0517	0.0397	0.5714	0.4898	0.3065	0.4328	0.1054	0.1391	0.0000
glass5	NB-Tomek	0.0000	0.0000	0.0500	0.0303	0.3286	0.2862	0.1589	0.3152	0.0443	0.0985	0.0000
cleveland-0_vs_4	orig	0.4000	0.9000	0.0704	0.0720	0.5000	0.5052	0.5109	0.3346	0.0535	0.7760	0.0000
cleveland-0_vs_4	NB-Tomek	0.3000	0.7000	0.0712	0.0548	0.4000	0.4700	0.3556	0.2612	0.0480	0.6849	0.0000
ecoli3	orig	0.6000	0.1600	0.1102	0.0975	0.6800	0.5684	0.3698	0.3661	0.1623	0.4449	0.2161
ecoli3	NB-Tomek	0.0800	0.0000	0.0159	0.0050	0.1200	0.3381	0.0770	0.2301	0.0644	0.3532	0.0000
ecoli-0-1-3-7_vs_2-6	orig	0.4000	0.4000	0.0386	0.0355	0.6000	0.3609	0.3260	0.3845	0.0395	0.2081	0.0000
ecoli-0-1-3-7_vs_2-6	NB-Tomek	0.4000	0.2000	0.0294	0.0245	0.3000	0.3320	0.2253	0.2872	0.0401	0.1595	0.0000

Table C.7: Complexity measure values before and after NB-Tomek for the outlier group datasets