UNIVERSIDADE D
COIMBRA

Taís Rigor Soares

PREVISÃO DE RETORNOS DE ATIVOS NO MERCADO FINANCEIRO ATRAVÉS DE MÉTODOS DE OTIMIZAÇÃO

Seminário em Estatística, Otimização e Matemática Financeira no âmbito do Mestrado em Matemática orientado pelo Professor Doutor José Luís Esteves dos Santos.

setembro de 2023

Previsão de Retornos de Ativos no Mercado Financeiro Através de Métodos de Otimização

Taís Rigor Soares

UNIVERSIDADE D
COIMBRA

Dissertação de Mestrado | MSc Dissertation

Mestrado em Estatística, Otimização e Matemática Financeira
Master in Statistics, Optimization and Mathematics in Finance

September 2023

Agradecimentos

A Dissertação de mestrado apresentada não era possível sem o apoio e incentivo de pessoas importantes, aos quais presto o meu agradecimento. Ao meu orientador, Professor Doutor José Luís Esteves dos Santos, pela orientação prestada e pelos conhecimentos transmitidos fundamentais para a realização deste trabalho. Agradeço o facto de, ao longo deste ano, instigar a estruturar e aprofundar as minhas ideias e a desafiar a mim mesma.

À minha família, nomeadamente ao meu pai, à minha mãe e aos meus irmãos, que sempre me apoiaram incondicionalmente e que ajudaram a moldar a pessoa que sou hoje. O vosso esforço, carinho e preocupação foram essenciais nesta minha caminhada.

Aos meus amigos que fizeram com que esta viagem fosse especial e com que cada dia fosse uma aventura inesquecível. Grata por me terem acompanhado ao longo do meu percurso académico.

À minha amiga Inês, a pessoa que sempre me motivou e festejou as minhas conquistas ao meu lado nos últimos 5 anos. Obrigada por estares comigo nos momentos certos e por acreditares em mim. Esta caminhada não teria sido a mesma sem ti.

À Associação Académica de Coimbra, em especial ao Núcleo de Estudantes de Matemática, que a cada dia me desafiou a fazer mais e melhor e a partir da qual aprendi muito do que sei. As experiências vividas no associativismo juntamente com as pessoas que por aqui passaram, enriqueceram o meu percurso tanto profissional como pessoalmente.

Por fim, um obrigada à cidade que me viu viver e crescer. À cidade que agora me diz tudo e que em cada espaço conta uma história que permanece na minha memória.
Coimbra tem mais encanto, mas não é só na hora da despedida.

O trabalho apresentado neste documento foi parcialmente realizado no âmbito do projecto RiskBigData: Gestão Complexa de Risco no Regime de Utilização de Dados Massiva (PTDC/MAT-APL/1286/2021), financiado pela Fundação Portuguesa para a Ciência e Tecnologia (FCT).

Cofinanciado por:

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

COMPETE 2020

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Europeu
de Desenvolvimento Regional

Resumo

Esta dissertação tem como objetivo prever os retornos de ativos no mercado financeiro através de métodos de otimização. O modelo de *Markowitz* foi dos primeiros modelos a surgir no estudo da previsão de retornos, mas foi-se verificando que este não tinha capacidade para tratar de um grande conjunto de dados históricos. Assim, foram surgindo os algoritmos de *Machine Learning*. Após alguma pesquisa, foi decidido abordar o algoritmo de *Random Forest* pois, na área das finanças, é notório o seu bom desempenho tanto na previsão do sinal dos retornos (problema de classificação) como dos seus valores numéricos (problema de regressão).

Desta forma, foram analisados os conceitos deste algoritmo e todo o seu procedimento de maneira a entender o método e a importância que este tem na previsão dos preços das ações. Numa fase final do trabalho é feita uma seleção de ativos a partir de observações de dados reais diários, provenientes da recolha de dados do PSI. Com isto, foi possível fazer a respetiva análise dos comportamentos destes dados com o método da regressão linear e com o algoritmo de *Random Forest* para problemas de classificação e de regressão para cada ativo, e posteriormente verificar o desempenho e precisão dos métodos quanto à previsão dos retornos.

É importante ressalvar que a utilização de *Machine Learning* e de algoritmos como o *Random Forest* na previsão de retornos de ativos financeiros é uma abordagem relevante e atual, visto que o mercado financeiro é altamente dinâmico e complexo, e métodos tradicionais muitas vezes não conseguem captar todos os padrões e informações disponíveis nos dados históricos.

Os resultados experimentais mostraram que o algoritmo de *Random Forest* foi o que obteve melhores resultados, principalmente na previsão dos valores dos retornos.

Assim, acreditamos que esta dissertação contribui para o avanço do conhecimento na área de previsão de retornos financeiros e destaca a importância do uso de algoritmos de *Machine Learning*, como o *Random Forest*, como uma ferramenta eficaz para análise e tomada de decisões no mercado financeiro.

Palavras Chave: retorno, *Machine Learning*, regressão linear, *Random Forest*.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Durante muito tempo, a ideia de prever a direção da evolução dos preços no mercado financeiro era praticamente impossível, a qual era suportada pelas Hipóteses do *Random Walk* ([24]) e do Mercado Eficiente ([19]). A primeira sugere que os preços dos ativos no mercado financeiro seguem um padrão com um caminho aleatório, o que significa que os preços futuros são imprevisíveis de acordo com dados históricos. Por sua vez, a segunda hipótese afirma que um mercado é eficiente quando os reflete de forma rápida e precisa qualquer informação atual nos preços dos ativos, não oferecendo a oportunidade de obtenção de ganhos anormais, ou seja, retornos que tragam lucros extraordinários aos investidores. Estas hipóteses levaram a acreditar que as mudanças nos preços das ações não eram previsíveis.

Sendo um ativo qualquer tipo de investimento financeiro, este é caracterizado pela relação entre o retorno e risco, que mostra que quanto maior for o nível de risco, maior será a possibilidade de retorno. Por retorno entende-se os ganhos, positivos ou negativos, associados a uma carteira de ações, resultando da soma dos ganhos de cada um dos ativos que constituem o portefólio em questão. O risco, por sua vez, está associado à incerteza dos dados e funciona como uma medida para quantificar a possibilidade de ocorrência de perdas e, tal como o retorno, está diretamente relacionado com a carteira.

Com o tempo, foram surgindo investidores individuais, bem como institucionais, que quebraram a crença e foram capazes de vencer o mercado financeiro para obter lucros ([3]). A falta de eficiência na previsão de retornos é devido a uma variedade de fatores que nem sempre são possíveis de controlar: variáveis macroeconômicas, fatores setoriais e específicos da empresa, sentimentos dos investidores em relação a uma determinada empresa, eventos geopolíticos, entre outros. Consequentemente, os mercados de ações são suscetíveis a rápidas alterações, que muitas vezes se transformam em movimentos aleatórios nos preços dos ativos.

Com o intuito de encontrar padrões/tendências num conjunto imenso de dados ao longo de um período de tempo, o *Machine Learning* (ML) provou ser vantajoso no mundo das finanças, quanto à previsão dos retornos de ativos. ML é definido por Alpaydin (2010) como:

> "programar computadores para otimizar um critério de desempenho usando dados de exemplo ou experiência passada" [8].

O uso de técnicas de ML para a previsão de preços de ações tem sido muito explorada e estas podem ser utilizadas em problemas de regressão, mas também em problemas de classificação.

Assim, vários investigadores têm aplicado o ML para prever retornos, por exemplo numa empresa, e assim obter melhores *insights* - descobertas ou conclusões importantes extraídas de um grande conjunto de dados através de algoritmos, que ajudam a tomar decisões, aprimorar produtos e serviços e ainda personalizar experiências de um cliente. De notar que a aplicação de modelos ML no mercado financeiro é relativamente recente ([21]). Esta abordagem mais tecnológica, teve como objetivo encontrar alternativas aos métodos tradicionais para a previsão e difusão de preços de ativos. Normalmente os modelos usados nesta previsão envolvem métodos estatísticos, como modelagem de séries temporais e análise multivariada ([14]), onde os movimentos dos preços dos ativos são geralmente tratados em função do tempo e resolvidos como um problema de regressão ([4]).

Dado o contexto apresentado, o presente trabalho tem como objetivo apresentar e demonstrar como é possível estudar o movimento dos preços de ações e prever os retornos de ativos com a ajuda de métodos de otimização, nomeadamente com a análise do comportamento dos dados no estudo da regressão linear e com o uso de um algoritmo de *Machine Learning*. Os resultados deste estudo poderão ser utilizados no futuro na otimização de portefólios, com o intuito de melhorar as abordagens clássicas existentes.

Capítulo 2

Machine Learning na previsão de retornos

2.1 Revisão da literatura

Um dos avanços mais significativos na otimização de carteiras de investimento ocorreu com a publicação do artigo *Portfolio Selection* de Markowitz, em 1952, onde foi apresentado a Teoria Moderna do portefólio [26]. Este modelo ajuda na seleção do portefólio mais eficiente analisando várias carteiras possíveis através da relação entre o retorno e o risco. Neste caso o risco é medido pela variância do retorno da carteira e a solução será o portefólio de menor risco dado um nível de retorno.

Por mais que os modelos baseados na teoria moderna do portefólio sejam complexos e sofisticados, todos eles sofrem de um ou mais dos seguintes pressupostos:

1. os mercados são eficientes;
2. os investidores são racionais;
3. um dos índices importantes do mercado pode ser usado como *proxy* (procurador) de uma carteira;
4. os investidores são tomadores de preços, ou seja devem aceitar o preço definido pelo mercado e não afetam os preços dos títulos fazendo pedidos;
5. há liquidez e capital ilimitado;
6. as adoções generalizadas da teoria de Markowitz e as posteriores técnicas de gestão do risco com base nele, não têm consequências sobre as próprias técnicas de gestão do risco [6].

A teoria de Markowitz foi estendida por muitos investigadores para melhorar as suas principais limitações, como a alta sensibilidade dos preços históricos. Esta sensibilidade da solução de Markowitz aos preços faz com que os portefólios otimizados com os dados históricos não tenham, em geral, bons desempenhos com os dados futuros.

Na tentativa de melhorar a previsão dos preços, foram surgindo algoritmos de ML que têm vindo a tentar identificar padrões nas evoluções dos preços dos ativos que permitam uma melhor estimativa.

Estes algoritmos foram sofrendo alterações constantemente e, por isso, existem inúmeros estudos sobre os seus desempenhos.

Alguns investigadores, dos quais descrevemos de seguida os mais relevantes, analisaram algoritmos aplicados ao modelo do "dia seguinte", que prevê o resultado do preço da ação no dia seguinte, e ao modelo de "longo prazo", que prevê o resultado do preço da ação para os próximos *n* dias, para preverem a tendência dos preços das ações. Dai e Zhang (2013) [9] aplicaram a estes modelos os algoritmos de Regressão Logística, Análise Discriminante Gaussiana (GDA) e *Support Vector Machine* (SVM). Os dados que foram utilizados para o estudo continham dados diários das ações de uma empresa denominada *3M Stock* de 01/09/2008 até 11/08/2013 e daqui foram extraídas dezasseis características (também chamadas de variáveis preditoras, recursos ou atributos). O modelo do "dia seguinte" apresentou resultados de precisão que variou de 44,52% a 58,2%. Dai e Zhang justificaram os seus resultados afirmando que o mercado financeiro dos EUA é eficiente, o que significa que nem a análise fundamental nem a técnica podem ser utilizadas para obter ganhos superiores. No entanto, resultados mostraram que a longo prazo, o modelo que apresentou melhores resultados foi o algoritmo SVM com a maior precisão de 79,3%.

Em Xinjie (2014) [10], os investigadores aplicaram várias características incluindo indicadores técnicos, como o Índice de Força Relativa (RSI - Relative Strength Index) e a taxa de variação, aos modelos para tentar melhorar a sua precisão. Os autores usaram três ações com intervalo de tempo disponível de 01/04/2010 a 12/10/2014 e, com isto, eles tinham um conjunto de 84 características. A estas características foi implementado um algoritmo de árvore extremamente aleatório [15], para a seleção das características mais relevantes. Após isso, essas características foram treinadas com o algoritmo de SVM e os resultados mostraram uma precisão acima de 70%.

Li et ai.(2014) [23] analisaram modelos lineares e SVM e levaram em consideração a sensibilidade dos preços das ações a fatores externos. As condições externas consideradas foram cotações diárias de cinco contratos futuros de *commodities* (ouro, petróleo bruto, gás natural, milho e algodão), duas moedas estrangeiras (EUR, JPY) e uma taxa de juros (taxa do tesouro de 10 anos). Para além disso, também foram introduzidos dados diários de 2.666 ações dos EUA de 01/01/2000 a 10/11/2014. Este conjunto de dados inclui preço de abertura e de fecho, preço máximo e mínimo em cada dia e volume. Com os resultados obtidos, verificou-se que, dos algoritmos analisados, a regressão logística revelou ser o melhor modelo na previsão da tendência de preços, com uma taxa de sucesso de 55,65%.

2.2 Algoritmos de *Machine Learning*

O desenvolvimento de técnicas de *Machine Learning* (ML) começa no final do ano 1950. Este termo foi usado pela primeira vez em 1959 por Arthur Samuel, um engenheiro do MIT (Instituto de Tecnologia de Massachusetts), que desenvolveu um programa de computador capaz de jogar damas (game of checkers) [31]. Ele utilizou um método de *Reinforcement Learning* (método de aprendizagem por reforço), onde o programa ia melhorando o seu desempenho ao longo do tempo através da análise das jogadas e da adaptação das estratégias. O programa foi feito para ajustar automaticamente os pesos de avaliação das posições do tabuleiro de acordo com o *feedback* recebido das suas próprias partidas. O estudo e desenvolvimento deste programa teve como base o trabalho realizado por Claude Shannon, em 1950, que, apesar de não envolver a adaptação do computador por ML, explorou a ideia

de como um computador poderia ser programado para jogar xadrez de maneira eficaz, apenas com cálculos de posição simples.

Na década de 1960, surgiram alguns dos primeiros algoritmos de ML, incluindo o *perceptron* desenvolvido por Frank Rosenblatt. Este algoritmo consiste num classificador linear lidando apenas com problemas de classificação, onde o conjunto de dados seja linearmente separável. Na década de 1970, surgiu o algoritmo de *backpropagation*, desenvolvido por Paul Werbos, que faz o treino eficiente de redes neurais. Este algoritmo possui duas fases: o *forward pass* (passo para a frente), que é onde os dados de entrada são passados através da rede e obtém-se as previsões de saída, e o *backward pass* (passo para trás), que é onde se calcula o gradiente da função de perda na camada de previsão da rede e usamos esse mesmo gradiente para aplicar recursivamente a regra da cadeia (Chain *rule*) para atualizar os pesos da nossa rede.

Na otimização de portefólios, as redes neurais treinadas com *backpropagation* são usadas para encontrar a melhor alocação de ativos para um determinado perfil de risco. Estas podem receber dados de retornos históricos, volatilidades e correlações entre ativos.

Em 1980, apareceram os algoritmos de ML não supervisionados, incluindo o algoritmo de agrupamento *k-means* - usado para agrupar um conjunto de dados em *clusters* ou grupos baseados em suas características - e o algoritmo de análise de componentes principais (PCA - Principal Component Analysis).[1] Já em 1990, surgiram algoritmos baseados em árvores, incluindo C4.5 e CART (*Classification and Regression Tree*). Estes dois algoritmos são muito semelhantes pois ambos têm como objetivo construir árvores de decisão que dividem os dados em conjuntos cada vez mais homogéneos com base nas características (atributos) desses dados. As maiores diferenças entre estes dois métodos está na parte da seleção de atributos sendo que CART utiliza uma abordagem mais simples em comparação com o C4.5. Ele avalia todos os atributos disponíveis em cada nó e escolhe o atributo que apresenta melhor pureza após a divisão. Outra diferença é o facto de CART poder ser aplicado a problemas de classificação e regressão enquanto que C4.5 só pode ser aplicado a problemas de regressão.

Na década de 2000, o ML começou a ter um papel importante em várias áreas incluindo na visão computacional, processamento de linguagem natural e reconhecimento de fala. Posteriormente a sua popularidade continuou a crescer, e em 2018, apareceram os primeiros algoritmos de *deep learning*, incluindo o algoritmo de rede neural conhecida como o modelo de linguagem GPT-1 [17].

Este conjunto de algoritmos tem-se revelado uma ferramenta valiosa na área das finanças, devido à sua capacidade de processar e analisar grandes volumes de dados e identificar padrões. Este pode ser aplicado para prever tendências e movimentos dos mercados financeiros, o que pode ser útil para investidores e gestores de carteiras. Também pode ser usado para a otimização de carteiras de investimento, encontrando a combinação ótima de ativos para maximizar o retorno esperado e minimizar o risco. Neste contexto, os algoritmos mais utilizados na área das finanças são os seguintes:

[1] técnica de redução de dimensionalidade frequentemente utilizada em análise de dados e ML e tem como objetivo transformar um conjunto de dados original em um novo conjunto de dados com menos dimensões (chamados de componentes principais), enquanto tenta reter a maior parte da variabilidade presente nos dados originais, sendo que isso é alcançado pela identificação dos vetores que apresentam a melhor variação nos dados.

- *K-nearest neighbors* (KNN) [1] - tem como objetivo classificar cada amostra de um conjunto de dados avaliando a sua distância em relação aos vizinhos mais próximos. Se estes vizinhos forem maioritariamente de uma classe, a amostra em questão será classificada nesta categoria. Na área das finanças, este algoritmo pode ser usado para previsão do sinal de preços de ativos, classificação de tendências de mercado, análise de risco e deteção de anomalias (por exemplo, identificar movimentos de preços que são muito diferentes do comportamento histórico);
- *Artificial neural networks* [12] - são amplamente utilizadas para prever o comportamento dos preços de ações e identificar padrões em séries temporais financeiras;
- *Support vector machine* (SVM) [28] - estes algoritmos podem ser usados para classificação de ativos em categorias ou para previsões de retorno de ativos;
- *Random Forest* (RF) [33] e *Gradient Boosting* [18] - utilizados na área das finanças devido à sua capacidade de lidar com dados complexos e não lineares, bem como à sua capacidade de lidar com problemas de previsão e otimização.

Estes algoritmos apresentam vantagens e desvantagens, não existindo um que seja "melhor" para todos os casos de previsão dos retornos (ou do seu sinal) e/ou otimização de portefólios. A escolha do algoritmo depende de vários fatores, como o tipo de dados disponíveis, a função objetivo e as restrições do problema.

Os algoritmos de ML são frequentemente utilizados em problemas de regressão e classificação. Um problema de regressão é um tipo de problema de ML supervisionado onde o objetivo é prever uma saída contínua, como o preço ou retorno de um ativo. Um problema de classificação, é também do tipo supervisionado e tem como objetivo prever uma saída categórica. A título de exemplo, como saber se o retorno será positivo ou não. Alguns dos algoritmos podem ser usados tanto para problemas de regressão como para problemas de classificação como é o caso do *Random Forest* que será explorado nesta dissertação.

O objetivo deste trabalho é prever os retornos dos ativos no mercado financeiro, a partir da análise dos dados históricos. Neste contexto, os algoritmos de ML mais utilizados nesta temática são as KNN [1], Redes Neurais Artificiais [12], RF [33], *Gradient Boosting* [18] e SVM [28], como referido anteriormente.

Alguns estudos apresentam razões onde o algoritmo de *Random Forest*, para um grande conjunto de dados históricos, pode ser uma escolha eficaz para prever retornos de ativos. A título de exemplo refere-se o estudo realizado por Indu Kumar (2018) [22] onde o objetivo era comparar métodos de ML na previsão da tendência de preços no mercado de ações. Os algoritmos comparados foram: SVM, RF, KNN, *Naive Bayes* e *Softmax*. Escolheram trabalhar com um conjunto de dados diários de cinco ativos, nomeadamente a *Amazon, Cipla, Eicher, Bata* e *Bosch*, num período de cinco a dez anos. Estes conjuntos de dados tinham características como preço de abertura, preço de fecho, preço máximo e mínimo diário e volume. Na previsão do modelo foram usados doze indicadores técnicos, entre os

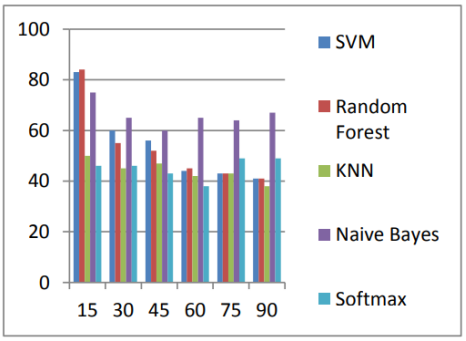
quais *moving average*, *relative strength index* (RSI), *stochastic oscillator* e *volatility*. O conjunto de dados inicial foi dividido em dois conjuntos onde 70% dos dados foram para o conjunto de treino e 30% para o conjunto de teste. Para o estudo comparativo dos algoritmos de ML na previsão do mercado de ações, foram utilizados o *accuracy* e *F-measure* que são dados pelas seguintes fórmulas:

$$accuracy = \frac{TP+TN}{TP+TN+TP+TN} \tag{2.1}$$

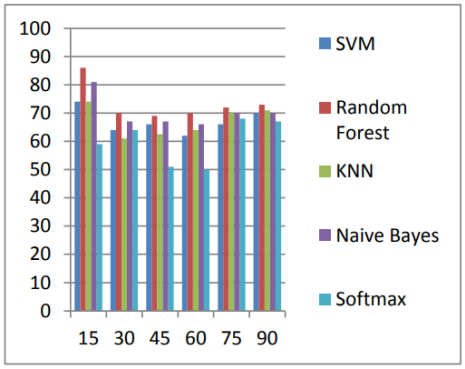
$$\text{F-measure} = 2 \times \frac{\frac{TP^2}{(TP+FP)\times(TP+FN)}}{(\frac{TP}{TP+FP})+(\frac{TP}{TP+FN})} \tag{2.2}$$

sendo que TP e TN são os valores positivos e negativos verdadeiros e FP e FN os valores positivos e negativos falsos.

Com isto, foram feitos dois estudos em que no primeiro os algoritmos foram testados com doze indicadores e no segundo com seis. Fazendo a comparação, para um conjunto de dados pequeno, verificaram que o algoritmo com melhor desempenho foi *Naive Bayes*, enquanto que para um grande conjunto de dados, o RF obteve melhores resultados, como podemos ver pelos gráficos a seguir:



(a) Algoritmos para um conjunto pequeno de dados.



(b) Algoritmos para um conjunto grande de dados.

Fig. 2.1 Comparação entre algoritmos (retirado de [22]).

De acordo com o estudo apresentado, vimos que o algoritmo de RF consegue lidar com dados de alta dimensão e ruído, o que é útil quando temos conjuntos de dados financeiros complexos. É um algoritmo que trata de dados tanto categóricos como numéricos de problemas com várias classes, sendo ideal se quisermos prever retornos de diversas empresas em simultâneo. Também é capaz de suportar a variação dos dados, ou seja, a extensão em que os valores do conjunto de dados se afastam da média ou do valor central, o que torna menos propício ao *overfitting*.

O classificador de RF gera múltiplas árvores de decisão, utilizando subconjuntos selecionados aleatoriamente de amostras e variáveis de treino [20]. Cada árvore funciona como um classificador e, no fim, combinam-se as decisões de cada uma das árvores. De acordo com Wu et al. (2020) [35],

este algoritmo, de uma forma geral, tem sido bem sucedido como um método de classificação. No seu artigo, Wu analisou as mudanças de tendência de dois índices de ações na China usando ações diárias e fez uma previsão da tendência no mercado financeiro, ou seja foi ver se os preços subiram ou desceram, para orientar a análise de investimento. Para este estudo, ele utilizou quatro algoritmos tradicionais de ML como *Logistic Regression* (LOGREG), *k-Nearest Neighbours* (KNN), *Random Forest* e SVM e comparou-os com dois modelos de *deep learning* tais como *long short-term memory* (LSTM) e *gated recurrent unit* (GRU). Foram introduzidas nos testes algumas variáveis métricas: *accuracy* (Acc) - avalia o percentual de acertos, isto é, ela pode ser obtida pela razão entre a quantidade de acertos e o total de entradas; *precision* (P) - quantidade de verdadeiros positivos sobre a soma de todos os valores positivos; *recall* (R) - mede a quantidade de vezes que o seu modelo acerta em relação ao total de vezes que ele deveria ter acertado - e *F1_score* (F1) - média ponderada da precisão e do *recall*. Focando nos testes realizados com os modelos tradicionais de ML, foi possível verificar que nas três experiências, o *Random Forest* apresentou bons resultados, como podemos verificar na Tabela 2.1.

Teste	Precision				Recall			
	KNN	LOGREG	RF	SVM	KNN	LOGREG	RF	SVM
E	0.7050	0.6617	0.6856	0.6759	0.6919	0.8234	0.6469	0.7833
C1	0.5542	0.5306	0.5482	0.5339	0.4583	0.9446	0.4152	0.9431
C3	0.5402	0.5209	0.5377	0.5269	0.4688	0.8829	0.4196	0.8015
Teste	Accuraty				F1_Score			
	KNN	LOGREG	RF	SVM	KNN	LOGREG	RF	SVM
E	0.6752	0.6752	0.6469	0.6781	0.6984	0.7337	0.6657	0.7256
C1	0.5172	0.5273	0.5083	0.5330	0.5017	0.6795	0.4725	0.6818
C3	0.5187	0.5192	0.5130	0.5250	0.5020	0.6552	0.4714	0.6358

Tabela 2.1 Valores das métricas dos resultados da classificação dos conjuntos de teste de KNN, LOGREG, RF e SVM (amostra de resultados obtidos no artigo [35]).

A maioria dos estudos que aborda a previsão dos retornos diários de índices ou outros ativos financeiros, utilizam o preço de fecho como base para tal [34]. Todavia, Gorenc Novak e Velušček (2016) [25] propõem o uso dos preços máximos diários para isso, uma vez que conclui que estes são menos voláteis do que os preços de fecho diários. Por isso, optaram por classificar os preços máximos diários, a fim de prever se estes preços iriam subir ou descer. Os resultados de classificação bem-sucedidos no conjunto de dados de teste foram, em média, 60%.

Após a análise destes estudos, decidimos que o algoritmo que deveria ser abordado nesta dissertação, seria o algoritmo de *Random Forest*.

Capítulo 3

Modelo *Random Forest*

O *Random Forest* (RF) é um algoritmo de ML desenvolvido por Leo Breiman [7] e Adele Cutler, em 2001. Trata-se de um algoritmo que constitui um conjunto de árvores de decisão, desenvolvidas para obter um melhor desempenho de previsão em comparação com uma única árvore de decisão. Este algoritmo é capaz de trabalhar com dados desequilibrados, ou seja quando a classificação de dados não é igual, e com alta dimensão, o que o torna uma boa opção para a previsão de retornos de ativos.

3.1 Árvores de decisão

Segundo Maimon e Rokach (2014) [11], a árvore de decisão é o conjunto dos possíveis resultados de uma série de alternativas relacionadas. Neste sentido, a cada passo é escolhida uma variável que melhor divide o conjunto de amostras, podendo usar diferentes medidas ou critérios de divisão como o *Gini impurity* e o *information gain* que serão explicados mais à frente, os quais originam ramificações. Trata-se de uma hierarquia de perguntas do tipo "sim/não", na qual as questões realizadas dependem das respostas dadas às perguntas anteriores, com os ramos a serem construídos a partir da questão original até que uma resposta apropriada seja obtida [16].

Uma árvore de decisão, ou preditor, é um algoritmo usado para resolver problemas de classificação e regressão, estabelecendo regras para tomada de decisão. Ele criará uma estrutura de árvore com vários nós e ramos, onde cada nó representa uma decisão a ser tomada e cada ramo uma possível consequência dessa decisão. A árvore é construída de tal forma que as características mais importantes são colocadas nos nós superiores da árvore. As características também conhecidas como atributos, recursos ou variáveis preditoras, são medidas que se utiliza para descrever os exemplos ou amostras de dados e podem ser de diferentes tipos como numéricos, categóricos, textuais, etc. A árvore será construída usando uma técnica de divisão recursiva, onde o algoritmo segue cada ramo até chegar a um nó final (nó-folha) que representa a previsão final.

Vejamos um exemplo de uma árvore de decisão no contexto das finanças. Imaginemos que estamos a construir uma árvore de decisão para prever se um ativo terá retorno diário positivo no dia seguinte. Sendo que r_t corresponde ao retorno do dia t, pretendemos prever o sinal de r_{t+1}.

A Figura 3.1 representa uma árvore de decisão simples para prever o sinal de r_{t+1} a partir do sinal de r_t. No primeiro nó (nó-raiz) temos a decisão "r_t é maior ou igual que 0?". Se a resposta for sim, esta informação seguirá por um ramo, indicando que o retorno no dia seguinte também deverá ser maior ou igual a 0. Caso contrário, a informação seguirá o outro ramo e teremos que r_{t+1} é menor que 0.

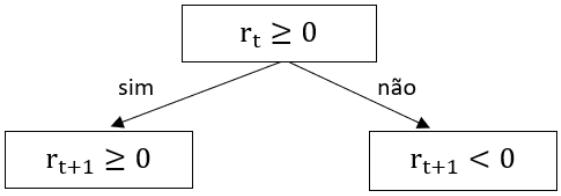


Fig. 3.1 Árvore de decisão simples para prever o sinal de r_{t+1}.

Aumentando o número de variáveis preditoras, temos a possibilidade de desenvolver árvores de decisão mais complexas. A Figura 3.2 é um exemplo de uma árvore de decisão que permite prever o sinal de r_{t+1} a partir de duas variáveis preditoras: r_t e r_{t-1}.

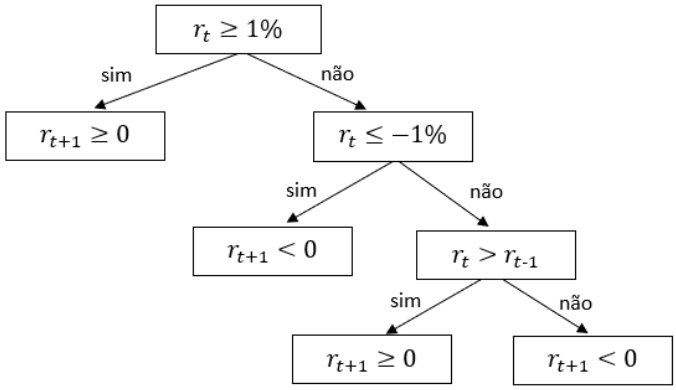


Fig. 3.2 Árvore de decisão mais complexa para prever o sinal de r_{t+1}.

Para ser feito o estudo completo dos dados iniciais, é necessário dividir o conjunto de dados em três partes: o conjunto de treino, o conjunto de teste e o conjunto de validação. O conjunto de treino é o conjunto de dados reais usado para treinar o modelo, isto é, faz a calibração dos seus parâmetros. Aqui está envolvido um algoritmo de otimização onde se ajusta os pesos das variáveis preditoras por forma à previsão se ajustar o melhor possível aos resultados reais nesses dados. Depois que o modelo é treinado, este é testado com o conjunto de dados de teste para ver o que é feito e qual é a interação com os parâmetros do modelo (aqui podem ser feitos ajustes até encontrar o conjunto de características que gere o melhor desempenho). Finalizada esta parte, será feita uma avaliação ao desempenho do modelo com o conjunto de validação. Nesta etapa, é feita uma avaliação *out-of-sample* (dados não utilizados na calibração do modelo) por forma a medir o desempenho real do algoritmo. Normalmente, a divisão aleatória destes dados pelos conjuntos é 50%, 25% e 25%, respetivamente.

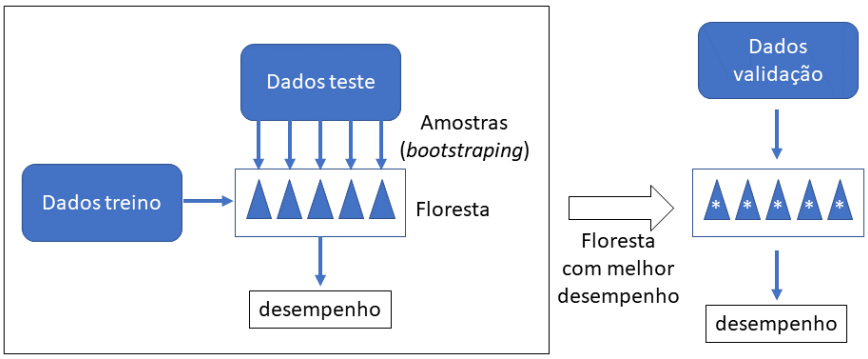


Fig. 3.3 Esquema do processo de divisão dos dados iniciais e construção do modelo.

3.2 Descrição do algoritmo RF

A ideia por trás do RF é construir várias árvores de decisão, formando uma floresta aleatória, cada uma baseada numa amostra aleatória do conjunto de dados de treino e numa amostra aleatória das características. As previsões de cada árvore são combinadas para produzir uma previsão final o que ajuda a reduzir a variação das previsões individuais das árvores e a aumentar a precisão.

De seguida, observamos uma representação ilustrativa básica do algoritmo de *Random Forest*.

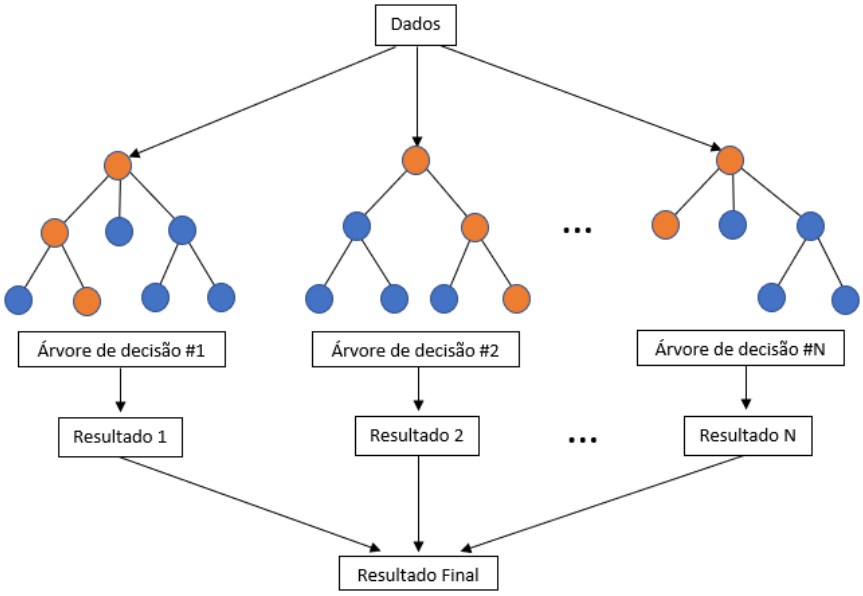


Fig. 3.4 Esquema que representa o algoritmo *Random Forest* com *N* árvores de decisão.

Inicialmente é necessário definir o primeiro nó da árvore, que será a primeira condição verificada, dando origem aos ramos necessários. De acordo com um critério de seleção definido (os quais serão descritos na 3.2.1), será escolhida a melhor variável para compor o nó. Para a escolha da variável do próximo nó, serão escolhidas duas ou mais variáveis, excluindo as selecionadas anteriormente, e o processo de escolha se repetirá. Este procedimento será feito até ser construído o último nó da árvore.

Na construção da próxima árvore, o que foi descrito anteriormente será refeito, levando à criação de uma nova árvore de decisão. Provavelmente essa árvore será diferente da primeira, pois a seleção das amostras será feita de maneira aleatória. No fim, será necessário combinar os resultados das várias árvores produzidas numa única resposta. Existem vários métodos para o efeito, os quais serão descritos na secção 3.2.2.

Podemos construir quantas árvores quisermos, sendo que quanto mais árvores forem criadas, melhor serão os resultados do modelo. Contudo, quando o número de árvores já é muito elevado, essa melhoria é pouco significativa e requer longos tempos de treino, aumentando a necessidade de recursos computacionais (especialmente quando o conjunto de dados é de grande dimensão).

3.2.1 Critérios de seleção de características para cada nó

Cada nó vai estar associado a um subconjunto da amostra de modo a que o nó-raiz contenha toda essa amostra e à medida que se procede à divisão dos nós, realiza-se uma partição desse subconjunto e atribuem-se a cada nó-filho um dos elementos da partição. No fim do procedimento, a reunião dos subconjuntos associados aos nós sem filhos (folhas) deve constituir uma partição disjunta da amostra inicial.

Para o tratamento de grandes amostras de dados, foi necessário criar técnicas de seleção para a divisão dos dados pelos diferentes nós. Os objetivos desta seleção são principalmente evitar o *overfitting*, também conhecido como sobreajuste, e melhorar o desempenho e a precisão do modelo. Esta situação acontece quando um modelo se ajusta excessivamente aos dados de treino e depois não tem um bom desempenho com dados que não fizeram parte desse conjunto.

Nas árvores de decisão, estes critérios ou métricas de divisão pretendem dividir o conjunto de dados a partir de um nó-pai. Será escolhida a característica que minimize o grau de impureza dos nós-filhos. Se num determinado nó a impureza for nula, significa que todos os dados nele pertencem à mesma classe, $\hat{\pi}_{k,t} = 1$, e se esta for máxima indica que todas as classes estão igualmente presentes naquele nó, $\hat{\pi}_{k,t} = \frac{1}{K}$, $k = 1, ..., K$ [2].

Entre as principais medidas de impureza estão o *Gini impurity* (índice de Gini) e o *information gain ratio* (razão de ganho de informação) que descrevemos de seguida. Para caracterizar cada uma destas medidas, primeiro denotemos as seguintes variáveis (adaptado de [13] e [27]):

- φ - conjunto de treino;
- K - número de classes;
- X - espaço de características;
- γ - vetor de características;
- $n_{\bullet,t}$ - número de elementos de φ incidentes no nó t (amostra *bootstrap*);
- $n_{k,t}$ - número de elementos de φ incidentes no nó t e pertencente à classe k;
- $\pi_{k,t}$ - probabilidade de um elemento $\gamma \in X$ incidente sobre o nó t e pertencente à classe k;

- $\hat{\pi}_{k,t}$ - estimador pontual de $\pi_{k,t}$ representando a frequência relativa da classe k sobre o nó t, ou seja:

$$\hat{\pi}_{k,t} = \frac{n_{k,t}}{n_{\bullet,t}}$$

Gini impurity

Esta medida de impureza foi desenvolvida e proposta por Conrado Gini, em 1912, e normalmente é utilizada em análises económicas e sociais. Este critério tem como objetivo selecionar a melhor característica para dividir os nós da árvore de decisão. A expressão desta medida é dada por

$$G(t) = 1 - \sum_{k=1}^{K} \hat{\pi}_{k,t}^2 \tag{3.1}$$

Sob essa medida, a característica escolhida é a que apresenta o maior decréscimo de *Gini impurity* (menor impureza de *Gini*) na divisão do nó, e é obtido por:

$$\Delta G(t) = G(t) - \sum_{z=1}^{Z} \frac{N_{t_z}}{N_t} \times G(t_z) \tag{3.2}$$

onde Z é o número de nós-filhos, N_t é o número total de elementos do nó-pai e N_{t_z} é o número de elementos do nó-filho t_z. Portanto, o decréscimo de *Gini* é calculado subtraindo a média ponderada das *Gini impurities* dos nós-filhos com a *Gini impurity* do nó-pai.

Entropia

Em algoritmos de ML, a entropia é uma medida de impureza ou incerteza de informação. O cálculo da entropia num determinado nó numa árvore de decisão é dado por

$$\varepsilon(t) = -\sum_{k=1}^{K} \hat{\pi}_{k,t} \times log_2[\hat{\pi}_{k,t}] \tag{3.3}$$

A característica que apresentar a maior diferença entre a entropia do nó-pai e a média ponderada das entropias dos nós-filhos, será a selecionada. Essa diferença é conhecida como *information gain* (ganho de informação) e a sua expressão é da forma

$$\Delta\varepsilon(t) = \varepsilon(t) - \sum_{z=1}^{Z} \frac{N_{t_z}}{N_t} \varepsilon(t_z) \tag{3.4}$$

Partindo da entropia, o algoritmo verifica o ganho de informação de cada característica. Aquela que apresentar maior ganho de informação será a característica escolhida para o nó.

Information gain ratio

De acordo com as afirmações de Quinlan [29], o critério de *information gain* tende a dar maior preferência às características com muitas divisões possíveis. A medida de *information gain ratio* foi proposta para contornar este problema. Esta é utilizada para medir a quantidade de informação ganha

ao dividir os dados com base numa determinada característica. A razão de ganho de informação, representada por $GR(t)$, é conceitualmente o ganho de informação ponderado:

$$SP(t) = -\sum_{z=1}^{Z} \frac{N_{t_z}}{N_t} \times log_2(\frac{N_{t_z}}{N_t}) \tag{3.5}$$

$$GR(t) = \frac{\Delta\varepsilon}{SP(t)} \tag{3.6}$$

Devido ao número elevado de dados para análise e a necessidade de construir árvores de decisão muito grandes, este pode levar ao *overfitting*.

3.2.2 Métodos *ensemble - Bagging*

Existem várias formas de os resultados das diferentes árvores individuais (modelos) serem combinadas. Esta junção de resultados de modelos ML, neste caso as árvores de decisão, fazem com que o próprio algoritmo de RF seja considerado um modelo *ensemble*. Os modelos *ensemble* são construídos da mesma forma que algoritmos mais básicos, como, por exemplo, a regressão linear, mas possuem uma característica que os diferem que é a combinação de diferentes modelos para se obter um único. Podemos dizer que um modelo *ensemble* é um conjunto de modelos de ML que trabalham juntos para melhorar o desempenho do modelo final e são normalmente utilizados em problemas de classificação e regressão. Ou seja, em vez de usar apenas um modelo para fazer previsões, eles combinam as previsões de vários modelos para obter uma previsão final mais precisa.

Um dos métodos de *ensemble* utilizado no algoritmo de RF é o *bagging*, também conhecido como *bootstrap aggregation*. Inicialmente existe um conjunto de dados e o *bagging* vai escolher uma amostra aleatória desse mesmo conjunto. Cada modelo (árvore) é gerado a partir das amostras (amostras *bootstrap*) que são fornecidas pelos dados iniciais com reposição, o que significa que cada vez que um item é selecionado, ele é adicionado de volta ao conjunto de dados antes da próxima seleção, sendo que esta etapa é conhecida como *row sampling*. Esta fase de *row sampling* com reposição é chamada de *bootstrap* que é uma técnica com o objetivo de reduzir o alto viés e a alta variância de toda a floresta. Nesta fase, cada modelo será treinado de forma independente e após isso, é possível apresentar novos dados aos modelos e obter o resultado da previsão. No caso de um problema de classificação, como podemos ver na Figura 3.5, o resultado final é baseado na votação por maioria após a combinação de todos os resultados. Por outro lado, num problema de regressão, o resultado final será a média dos resultados de cada uma das árvores. Esta etapa é designada como *aggregation* [30].

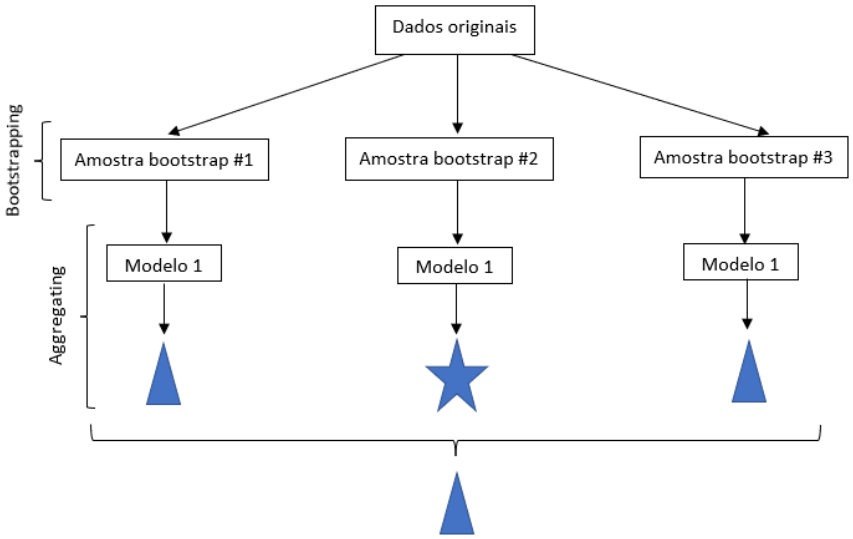


Fig. 3.5 Exemplo de esquema que representa o método *bagging* num problema de classificação (adaptado de [30]).

Usando o exemplo anteriormente apresentado para a previsão de r_{t+1}, suponhamos que temos os seguintes dados iniciais:

t	1	2	3	4	5	6	7	8	9	10
$r_t(\%)$	-4.1	0.55	-0.36	1.18	2.17	0.39	-0.21	5.32	-0.1	0.9
$r_{t+1}(\%)$	-2.5	1.8	0.6	-0.9	4.13	0.21	-0.89	1.4	3.3	1.2

Tabela 3.1 Conjunto de dados iniciais com a variável de retorno r_t.

Estes dados serão divididos em três conjuntos onde $t = 1,...,6$ fazem parte do conjunto de treino, $t = 7,8$ fazem parte do conjunto de teste e $t = 9,10$ estão inseridos no conjunto de validação.

O algoritmo, de forma aleatória, escolhe três amostras *bootstrap* a partir dos dados de treino, como podemos ver pela Tabela 3.2. Vamos trabalhar com 15 dados apesar de inicialmente só termos 8. Essa é uma das vantagens do *bootstrap* pois permite a repetição de elementos na mesma amostra.

Amostra 1			**Amostra 2**			**Amostra 3**		
t	r_t	r_{t+1}	t	r_t	r_{t+1}	t	r_t	r_{t+1}
1	-4.14	-2.5	2	0.55	1.8	5	2.17	4.13
3	-0.36	0.6	4	1.18	-0.9	7	-0.21	-0.89
4	1.18	-0.9	7	-0.21	-0.89	2	0.55	1.8
5	2.17	4.13	1	-4.1	-2.5	4	1.18	-0.9
8	5.32	1.4	3	-0.36	0.6	6	0.39	0.21
6	0.39	0.21	8	5.32	1.4	1	-4.1	-2.5

Tabela 3.2 Dados relativos a cada amostra *bootstrap*.

Como estamos a trabalhar apenas com uma variável, não há o processo de seleção para encontrar a melhor variável para o nó. Iremos treinar e testar uma árvore de decisão com a amostra 1. Primeiramente, definimos que a primeira decisão a tomar é ver se o dado é maior ou igual a 0. Se este for maior ou igual a 0, então r_{t+1} será maior ou igual a 0. Caso contrário, r_{t+1} será menor que 0. Assim, teremos os resultados da previsão do sinal de r_{t+1} apresentados na tabela seguinte:

t	r_t	r_{t+1}
1	< 0	< 0
3	< 0	< 0
4	≥ 0	≥ 0
5	≥ 0	≥ 0
8	≥ 0	≥ 0
6	≥ 0	≥ 0

Tabela 3.3 Resultados da previsão do sinal de r_{t+1} correspondente à amostra 1.

Testemos a árvore de decisão agora com os dados do conjunto de teste. r_7 é menor que 0, portanto r_8 também é menor que 0. r_8 é maior ou igual a 0 e, por isso, r_9 será maior ou igual a 0. Podemos observar pelos dados da tabela que a árvore está bem calibrada pois "acertou" nas duas previsões. Posteriormente, avaliamos o desempenho da árvore com o conjunto de dados de validação, t = 9,10. r_9 é menor que 0, portanto r_{10} também será menor que 0. Finalmente, r_{10} é maior ou igual a 0, e consequentemente r_{11} também o será. Analisando o desempenho deste modelo, podemos ver que a taxa de sucesso foi de 50%. Fazendo o mesmo processo para as amostras 2 e 3, teremos os resultados de cada uma das árvores. Finalizando, o resultado da previsão final do RF será a votação por maioria após a combinação de todos os resultados obtidos pelas árvores.

Resumindo, entendemos que o RF é um conjunto de k árvores de decisão treinadas com M amostras *bootstrap* (k e M não necessitam de ser iguais). No modelo de RF será aplicado o método de *bagging*

que é a junção da fase *bootstrap* com a fase *aggregation*. O *bootstrap* é aplicado para calibrar (treinar) a árvore de decisão e a *aggregation* é para juntar os vários resultados num só (resultado do RF).

É de realçar que o método de treino de florestas aleatórias não é tão simples e direto quanto aplicar o método de *bagging* a várias árvores de decisão individuais e simplesmente agregar os resultados de cada uma. O procedimento para treinar uma floresta aleatória é o seguinte:

1. Escolher aleatoriamente algumas variáveis preditoras entre as disponíveis, de maneira a evitar o *overfitting*;
2. Para o nó em estudo, selecione aleatoriamente p dados do conjunto de dados inicial, D - *row sampling*;
3. Calcular o melhor ponto de divisão para a árvore i usando o critério de seleção (técnica de divisão) adequado, divida o nó atual em nós-filhos e reduza o número de dados de D deste nó em diante;
4. Repetir as etapas 2 e 3 até que a profundidade máxima da árvore seja atingida ou a técnica de divisão atinja algum extremo;
5. Aplicar o conjunto de dados de treino na árvore i e testá-la com o conjunto de dados de teste, sendo que cada árvore irá fazer esse procedimento de forma independente;
6. Repetir os passos de 2 a 5 para cada uma das árvores da floresta;
7. Votar ou agregar os resultados de todas as árvores.

Comparando com as árvores de decisão, as florestas aleatórias são divididas selecionando várias variáveis em vez de apenas uma variável em cada ponto de divisão. Como as florestas aleatórias são formadas por várias árvores de decisão, o método de seleção de variáveis (critérios de divisão) e o procedimento de treino, teste e validação são feitas de forma análoga.

3.2.3 Classificação *versus* regressão

O algoritmo de RF permite trabalhar com problemas tanto de classificação como de regressão. A grande diferença entre estes dois tipos de problemas está relacionado com o tipo de dados de treino nomeadamente na parte dos exemplos de entrada e nas respetivas saídas desejadas. No caso de problemas de classificação, os dados de entrada são geralmente características ou atributos, enquanto que as saídas desejadas são as classes ou rótulos associados a esses exemplos. No caso de problemas de regressão, as saídas desejadas são normalmente valores numéricos. Consequentemente, o método de *aggregation* utilizado para obter a previsão final varia com o tipo de problema. Em problemas de classificação o resultado que mais se repete será o escolhido para ser o resultado final enquanto que nos problemas de regressão será utilizada a média dos valores.

Nas Figuras 3.1 e 3.2 foram apresentados dois pequenos exemplos de árvores de decisão para o problema onde se pretende prever o sinal do retorno no dia seguinte. Por outro lado, poderíamos tentar prever o retorno do ativo em vez de apenas o seu sinal. Nesse caso, teríamos um problema de regressão. A Figura 3.6 mostra um exemplo de uma árvore de decisão simples para esse problema.

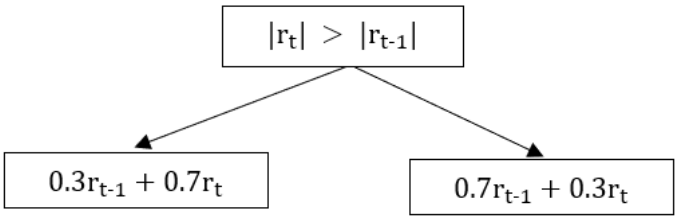


Fig. 3.6 Árvore de decisão para um problema de regressão.

As folhas darão resultados "contínuos", isto é, deverá ser uma fórmula que depende dos *inputs*. Geralmente o aparecimento destes resultados significa que estamos perante um problema de regressão, onde o objetivo é prever valores numéricos em vez de classes.

3.3 Análise matemática do algoritmo

Esta secção baseia-se no trabalho de Leo Breiman (2001) [7], no trabalho de Erwan Scornet [32] e no livro de Richard Berk [5] sobre a consistência do *Random Forest*. Desse modo, serão seguidas as notações e nomenclaturas utilizadas pelos autores.

Breiman (2001) originalmente definiu *Random Forest* como sendo um classificador, quando é usado variáveis de resposta categóricas, formado por uma coleção de árvores de decisão $\{h(x,\Theta_k), k = 1,...,M\}$, sendo Θ_k um vetor aleatório que é independente dos vetores aleatórios passados $\Theta_1,...,\Theta_{k-1}$, mas possuem a mesma distribuição (conjunto dos pesos/parâmetros da árvore k). Este conjunto de árvores é denominado por "floresta aleatória" e a intenção é atribuir classes a observações usando informações contidas em um conjunto de preditores. Cada árvore é criada a partir de um conjunto de treino, conjunto selecionado aleatoriamente a partir do conjunto de dados inicial, e por Θ_k, resultando assim num classificador $h(x,\Theta_k)$ onde x é um vetor de entrada de P, valores preditores usados para atribuir uma classe e k um índice para uma determinada árvore.

Definição 3.3.1. *O RF é um classificador que consiste em um conjunto de classificadores estruturados em árvores $h(x,\Theta_k), k = 1,...,M$ onde Θ_k são vetores aleatórios independentes e identicamente distribuídos e cada árvore obtém um resultado único para a classe mais popular na entrada x.*

A saída de um classificador é uma classe atribuída para cada observação, determinada pelos valores de entrada x. Nas florestas aleatórias, a classe atribuída a cada observação é determinada por uma votação sobre o conjunto de classificadores de árvores para os quais essa observação não fazia parte do conjunto de dados de treino. É importante saber distinguir entre a classe concedida pelo k classificador e a classe final concedida pela votação.

3.3.1 Condições de convergência - margem e erro de generalização

Temos definido um conjunto de classificadores $h_1(x), h_2(x), ..., h_k(x)$ e um conjunto de dados de treino de variáveis aleatórias independentes e identicamente distribuídas, criado a partir do vetor

aleatório X,Y. No vetor aleatório X,Y teremos que o conjunto de valores do preditor P poderá ser representado pela variável aleatória X e a classe real para esse ponto de dados será representada pela variável aleatória Y, definimos a função de margem como

$$mg(X,Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j), \tag{3.7}$$

onde $I(.)$ é a função indicadora, ou seja, função que indica se o elemento pertence ao conjunto, assumindo neste caso o valor 1, e 0 em caso contrário, j é uma classe incorreta e av_k, *average* da k árvore, denota a média sobre o conjunto de classificadores para um único ponto de dados realizado. A margem é o número médio de votos para a classe observada correta menos o máximo do número médio de votos para qualquer outra classe incorreta. Quanto maior for a margem, mais confiança teremos na classificação. O erro de generalização é dado por

$$g = P_{X,Y}(mg(X,Y) < 0), \tag{3.8}$$

onde X e Y indicam que a probabilidade está no espaço X,Y.

Para um grande número de árvores, é seguida a lei dos grandes números:

Teorema 3.3.1. *À medida que o número de árvores aumenta, o erro de generalização estimado converge para o erro de generalização da população, que é*

$$P_{X,Y}(P_\Theta(h(X,\Theta) = Y) - \max_{j \neq Y} P_\Theta(h(X,\Theta) = j) < 0) \tag{3.9}$$

sendo que

$$P_\Theta(h(X,\Theta) = j) < 0) = \frac{1}{N} \sum_{n=1}^{N} I(h(X,\Theta_n) = j)$$

Este resultado e a sua importância na convergência explica o porquê das florestas aleatórias não se ajustarem à medida que mais árvores são adicionadas, evitando assim o *overfitting*. Assim, o estimado erro de generalização converge para a probabilidade sobre X e Y será uma votação anulada.

Especificando, os dados da amostra *bootstrap* (amostra com dados selecionados aleatoriamente do conjunto de treino) serão usados numa determinada árvore. Assim, temos que P_Θ está associado à amostra *bootstrap* e $P_{X,Y}$ está associado ao conjunto de dados de treino.

3.3.2 A força do *Random Forest*

A função de margem para um ponto de dados realizado em floresta aleatória também pode ser escrita da seguinte forma:

$$mr(X,Y) = (P_\Theta(h(X,\Theta) = Y) - \max_{j \neq Y} P_\Theta(h(X,\Theta) = j), \tag{3.10}$$

Neste caso teremos que, considerando todas as árvores possíveis e o ponto de dados realizado, a função de margem será a diferença entre a probabilidade de uma classificação estar correta e a probabilidade máxima de estar incorreta. Esta função para uma determinada floresta aleatória

considera os dados do conjunto de treino fixos. Se tomarmos o valor esperado de 3.10 sobre as realizações do ponto de dados, a força de uma floresta aleatória é da forma

$$s = E_{X,Y} mr(X,Y). \tag{3.11}$$

Com isto, a força de uma floresta aleatória é essencialmente a margem média sobre os dados do conjunto de treino aleatórios e quanto maior for esse valor, melhor.

3.3.3 Estimativa da função de regressão com florestas finitas e infinitas

Biau e Scornet (2013) introduziram matematicamente o conceito de RF aplicado a problemas de regressão. Inicialmente, um vetor aleatório de entrada é observado e o objetivo é prever a resposta aleatória Y associado à variável aleatória X, através da estimativa da função de regressão:

$$m(x) = \mathbb{E}[Y|X = x] \tag{3.12}$$

Para isso, assumimos que dado um conjunto de dados de treino de variáveis aleatórias, independentes e identicamente distribuídas, onde $\mathbb{E}[Y^2] < \infty$, o qual será denotado por:

$$D_n = ((X_1, Y_1), ..., (X_n, Y_n)) \tag{3.13}$$

Neste contexto, o objetivo é usar o *Random Forest* para construir uma estimativa $m_n : X \to \mathbb{R}$ da função m, com base no conjunto de dados D_n.

Novamente Biau e Scornet (2016) cita que RF é um algoritmo de previsão baseado numa coleção de M árvores de regressão aleatórias onde, para a j-ésima árvore da floresta, o resultado da previsão de uma entrada x é definida por $m_n(x, \Theta_j, D_n)$ onde $\Theta_1, ..., \Theta_M$ são variáveis aleatórias independentes, e identicamente distribuídas com Θ, e são independentes do conjunto de dados D_n. Para facilitar a notação, podemos simplesmente escrever $m_n(x)$ em vez de $m_n(x; D_n)$.

Temos que a combinação das diferentes árvores de decisão pode ser representada da seguinte forma:

$$m_{M,n}(x, \Theta_1, ..., \Theta_M) = \frac{1}{M} \sum_{j=1}^{M} m_n(x, \Theta_j) \tag{3.14}$$

Pela lei dos grandes números, para qualquer x fixo, condicional a D_n, a estimativa de uma floresta finita tende para a estimativa de uma floresta infinita.

Esta segunda fórmula é mais comum já que a quantidade de árvores escolhidas na parametrização do método pode ser escolhida arbitrariamente e será limitada apenas pela disponibilidade computacional:

$$m_{\infty,n}(x) = \mathbb{E}_\Theta[m_n(x, \Theta)] \tag{3.15}$$

onde $\mathbb{E}_\Theta$ representa a expectativa em relação ao parâmetro aleatório Θ. Isto é justificado pela lei dos grandes números, que afirma que,

$$\lim_{M\to\infty} m_{M,n}(x,\Theta_1,...,\Theta_M) = m_n(x) \tag{3.16}$$

O risco de $m_{\infty,n}$ é definido por

$$R(m_{\infty,n}) = \mathbb{E}[m_{\infty,n}(X) - m(X)]^2 \tag{3.17}$$

enquanto que o risco de $m_{M,n}$ é igual a

$$R(m_{M,n}) = \mathbb{E}[m_{M,n}(X,\Theta_1,...,\Theta_M) - m(X)]^2 \tag{3.18}$$

Ambos os riscos são determinísticos, uma vez que a expectativa de 3.17 depende de X e D_n e a expectativa de 3.18 depende de X, D_n e $\Theta_1,...,\Theta_M$.

Dizemos que uma estimativa da função $m_{\infty,n}$ ($m_{M,n}$) é consistente se $R(m_{\infty,n}) \to 0$ ($R(m_{M,n} \to 0)$), quando $n \to \infty$.

É possível criar várias florestas aleatórias dependendo de como as árvores são criadas e de como a seleção aleatória de Θ influencia a construção das árvores. Pretendo utilizar o procedimento original de Breiman onde as divisões dependem de toda a amostra e são executadas para minimizar a variação dentro das duas células resultantes.

Capítulo 4

Análise de Ativos

4.1 Descrição da base de dados

O PSI é um índice que regula e traduz toda a evolução da cotação das maiores empresas cotadas na Bolsa de Lisboa, isto é, no mercado de capitais. Este índice reflete a evolução dos preços das ações dessas empresas. A liquidez de cada empresa cotada é medida pelo volume de transações em bolsa.

Os pesos dos constituintes do índice PSI são ajustados de acordo com o *free float* isto é, a parcela de capital negociada em bolsa e que não se encontra fixa na mão de acionistas de referência. Uma regra estabelecida a estes pesos é que nenhuma emissão (ou seja, nenhuma ação de uma empresa) pode ter uma ponderação superior a 15% no índice PSI. Isso é feito para evitar que uma única empresa domine excessivamente o índice, garantindo uma diversificação adequada.

Para o desenvolvimento deste trabalho, utilizei várias ferramentas como o *Excel* e o MATLAB para realizar o tratamento dos dados, executar os procedimentos de previsão e produzir gráficos e tabelas para posteriormente serem expostos os resultados.

A base de dados das empresas foram extraídas do portal 'yahoofinance' acessível em https://finance.yahoo.com/quote/PSI20.LS?p=PSI20.LS&.tsrc=fin-srch. Estes dados estão ordenados cronologicamente, do mais antigo para o mais recente, com a apresentação dos seguintes parâmetros para cada observação da periodicidade selecionada:

- preço de abertura e fecho: primeiro e último preço do ativo a cada dia;
- preço mais alto e mais baixo: maior e menor preço apresentado pelo ativo em cada dia;
- preço de fecho ajustado: valor que é ajustado após o preço da ação ter sido afetado por eventos corporativos, como por exemplo pela divisão de ações (*split*) ou distribuição de dividendos aos acionistas. Esta variável permite aos investidores e analistas terem uma visão mais precisa do desempenho do ativo ao longo do tempo, eliminando a influência dos eventos corporativos, permitindo uma comparação mais justa entre diferentes períodos;
- volume: quantidade total de unidades desse ativo que foram negociadas em um determinado período de tempo.

Nesta análise, considerou-se os dados das empresas que estiveram presentes no PSI no ano 2020 e têm dados históricos de 4 de janeiro de 2000 a 30 de dezembro de 2022, sendo que nem todos os dias têm dados associados devido, por exemplo, aos feriados. Ao todo, foram escolhidas 5 empresas de setores diferentes de maneira a ser possível ver o comportamento dos resultados nos diversos setores: Altri (ALTR) , Banco Comercial Português (BCP), NOS, *Navigator* (NVG) e *SOANE* (SON). Foram considerados dados com uma periodicidade diária.

ALTR	**BCP**	**NOS**	**NVG**	**SON**
5919	5850	5921	5825	5903

Tabela 4.1 Número de observações em cada ativo.

O primeiro passo é dividir os nossos dados iniciais em dois grupos: *in-sample* e *out-of-sample*. Os dados *in-sample* são utilizados para o treino do modelo e incluem os dados de treino e dados de teste. Eles correspondem ao período de 4 de janeiro de 2000 a 31 de dezembro de 2021. Uma das variáveis preditoras que iremos considerar é a média móvel dos últimos 28 dias. Deste modo, iremos considerar o início do estudo a partir de 11/02/2000, e utilizar os dados anteriores apenas para inicialização das variáveis preditivas. O dia 31 de dezembro será o dia em que nós vamos trabalhar como presente, ou seja, será o nosso "hoje" - dia em que se pretende fazer o investimento (no fim do dia). Os dados *out-of-sample* contêm os dados de validação e correspondem ao período 3 de janeiro de 2022 a 30 de dezembro de 2022. Este último conjunto é utilizado para avaliar o desempenho dos métodos em dados diferentes daqueles que foram utilizados para calibração dos modelos. Podemos ver, de seguida, a representação cronológica desta divisão de dados.

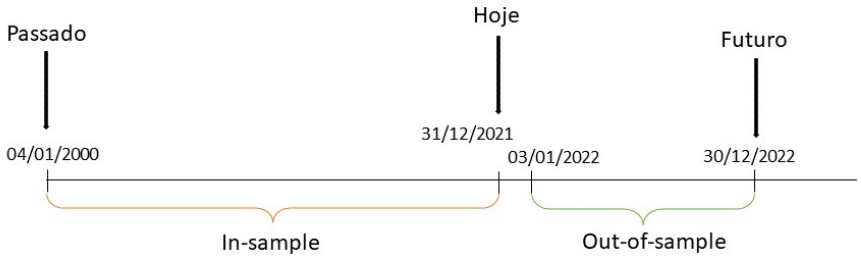


Fig. 4.1 Representação cronológica e divisão dos dados recolhidos de cada um dos ativos.

No caso dos retornos diários, observa-se que a distribuição dos retornos é próxima de uma log-normal, pelo que são considerados os retornos logarítmicos, isto é, determina-se os retornos dos logaritmos dos preços. Deste modo, sendo X_t o preço de fecho ajustado do dia t, o retorno r_t é dado por

$$r_t = ln(X_t) - ln(X_{t-1}) = ln(X_t/X_{t-1}) \tag{4.1}$$

Ativo	*in-sample*		*out-of-sample*	
	MR	DP	MR	DP
ALTR	6.14E-05	4.140E-02	-2.476E-04	3.084E-02
BCP	-8.633E-04	2.601E-02	1.673E-04	3.060E-02
NOS	-3.116E-04	1.976E-02	6.786E-04	1.200E-02
NVG	3.657E-04	1.620E-02	4.736E-04	1.561E-02
SON	-2.14E-04	2.073E-02	-7.52E-04	1.425E-02

Tabela 4.2 Valor médio dos retornos observados (MR) e desvio padrão (DP), em cada ativo, para os períodos *in-sample* e *out-of-sample*.

4.2 Descrição das variáveis preditoras

Alguns trabalhos [22] nesta temática apontam que é possível ter bons resultados na previsão da tendência do sinal do retorno a partir dos dados históricos considerando, por exemplo, a média móvel, mínimo e máximo, volume, etc.

Neste capítulo pretendemos construir um modelo para prever o sinal (e eventualmente o valor) do retorno diário do dia seguinte a partir dos dados históricos dos ativos considerando os seguintes parâmetros:

- y: retorno logarítmico diário;
- $x1, x2, x3$ e $x4$: média móvel do retorno logarítmico nos últimos 7, 14, 21 e 28 dias, respetivamente;
- u e p: dados relativos ao último e penúltimo dia dos retornos logarítmicos;
- $volume$: quantidade de negociação/número de transações de um ativo num determinado período de tempo. A variação no volume pode indicar mudanças no interesse do mercado por parte dos investidores e pode ser usado para confirmar tendências de preços. A título de exemplo, se o preço de uma ação está a subir e o volume está a aumentar, isso pode sugerir que a tendência de alta é sustentável. Da mesma forma, uma queda nos preços com aumento no volume pode indicar uma tendência de baixa mais forte;
- hl: diferença entre o valor mais alto (máximo) e o valor mais baixo (mínimo) ocorrido em cada dia, geralmente referida como "intervalo diário";
- co: diferença entre o preço de fecho e o preço de abertura ocorrido em cada dia, conhecida como "variação diária".

A inclusão das variáveis hl e co deve-se à possibilidade delas poderem trazer vantagens ao estudo. Estas variáveis são medidas diretas da volatilidade diária - medida de instabilidade ou incerteza associada ao ativo -, sendo que ativos com, por exemplo, variações diárias maiores são geralmente

considerados mais voláteis, o que pode afetar a estratégia de investimento. Com o intervalo diário e a variação diária é possível adquirir informações sobre a direção que o mercado está a tomar num determinado dia, ajuda a identificar tendências e padrões a curto prazo nos movimentos dos preços e ainda são relevantes para a avaliação de risco e retorno, pois podem ser usados para estabelecer metas de lucro com base na volatilidade.

4.2.1 Regressão linear

Neste trabalho iremos utilizar o método de regressão linear e comparar o seu desempenho com o do método *Random Forest*. Uma das vantagens do método de regressão linear é que os seus resultados são mais interpretáveis uma vez que temos acesso aos pesos atribuídos a cada variável preditiva.

Este método analisa a correlação entre as variáveis preditoras e a variável dependente (retorno do dia seguinte), atribuindo um peso a cada uma das variáveis nessa previsão. Deste modo, podemos fixar esses pesos e utilizá-los nos dados *out-of-sample* (os quais não foram utilizados para a calibração do modelo) e analisar o desempenho das previsões realizadas. O objetivo será encontrar os pesos que minimizem a soma dos quadrados dos erros (diferença entre os valores previstos e os valores reais dos retornos). A interpretação dos pesos do modelo ajuda a entender como cada variável afeta os retornos do ativo. Pesos positivos indicam uma relação positiva, enquanto que pesos negativos indicam uma relação negativa.

A utilização de todas as variáveis preditoras poderia causar *overfitting* e dificultar a interpretação dos pesos obtidos (muitos deles seriam baixos e "misturados" com outras variáveis). Assim, decidimos segmentar o conjunto de variáveis a estudar. Inicialmente pretendemos analisar se os retornos do último (u) e penúltimo (p) dia (separados ou em conjunto) poderão ter uma grande influência na previsão do retorno do dia seguinte, completando-a com a informação de uma média móvel. Assim, a primeira fase do processo consiste em considerar a variável $x1$ e analisar o ganho obtido quando se anexas as restantes médias móveis ($x2, x3$ e $x4$) se deveriam considerar. Desta forma teremos 12 casos a serem analisados. Na segunda fase, pretende-se introduzir as variáveis *volume*, *hl* e *co* na melhor estratégia obtida na fase 1.

De forma a ser possível fazer uma comparação entre a regressão linear e o RF, iremos utilizar outros métodos preditores mais simples (heurísticos) de forma a conseguirmos comparar os seus desempenhos:

- cada uma das médias móveis: $x1, x2, x3$ e $x4$ (aqui a previsão é feita apenas com o valor da média móvel, não havendo calibração dos pesos associados a cada uma);
- p: a previsão é feita apenas com o valor do último dia;
- média *in-sample*: considera-se a mesma previsão em todo o período *out-of-sample* e corresponde ao valor da média *in-sample*.

O método da regressão linear também será utilizado como um classificador, indicando que o retorno do dia seguinte terá o sinal correspondente ao da sua previsão.

O código utilizado em MATLAB para desenvolver este método encontra-se no Anexo A.

4.2.2 *Random Forest*

O outro método a ser trabalhado para a previsão dos retornos futuros das empresas selecionadas é então o *Random Forest*, explicado ao pormenor anteriormente, sendo este o método fulcral na análise deste trabalho. O objetivo será, com os dados históricos de cada ativo, formar amostras aleatórias, criar árvores e treiná-las. No caso de um problema de regressão, estas árvores serão treinadas com o conjunto de dados de treino, posteriormente estas serão testadas com o conjunto de dados de teste para a sua calibração e, por fim, o melhor conjunto de árvores será avaliado com conjunto de dados de validação para se obter o modelo com o melhor desempenho. Para os conjuntos mencionados, iremos considerar do *in-sample* 75% dos dados para treino e 25% para teste e os dados de *out-of-sample* farão parte do conjunto de dados de validação.

Com este algoritmo foram elaborados dois códigos MATLAB em que um deles é para um problema de classificação e o outro para um problema de regressão.

Random Forest por classificação

Com o problema de classificação pretendemos entender qual será o sinal das previsões de cada ativo no mercado, ou seja, se estes vão ter retornos positivos (1) ou negativos (0). Neste algoritmo iremos trabalhar com dados *out-of-bag*. Uma amostra *out-of-bag* é uma amostra que contém todos os dados que não foram escolhidos para certa amostra *bootstrap*. O *OOB_Score* é uma técnica de validação muito poderosa usada especialmente para o algoritmo *Random Forest* pois apresenta o modelo com menor variância. Este é calculado como sendo o número de linhas previstas corretamente da amostra *out-of-bag*. Com o sentido inverso temos *OOB Error* que é o número de classificações incorretas da amostra OOB.

O objetivo do *out-of-bag* é ver quais os dados que ficaram fora das amostras *bootstrap* e fazer a previsão nas árvores de decisão em que estão em falta. De notar que nenhuma destas árvores tem conhecimentos prévios dos dados *out-of-bag*. Por fim, a previsão final será por voto de maioria de acordo com os resultados das amostras *out-of-bag*.

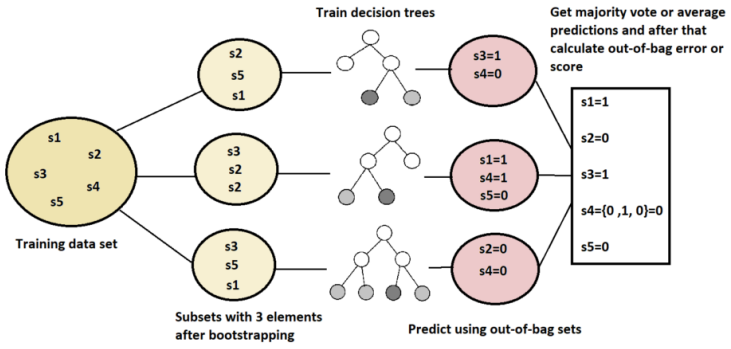


Fig. 4.2 Esquema completo do processo do algoritmo *Random Forest* por classificação (figura retirada de [36]).

Concluindo este método, com o *OOB Labels* serão previstos os retornos fora do *out-of-bag*.

O código utilizado em MATLAB para desenvolver este método encontra-se no Anexo A.

***Random Forest* por regressão**

No método de *Random Forest* por regressão, queremos obter os valores da previsão dos retornos dos ativos. Os dados iniciais serão divididos em três conjuntos: conjunto de treino, conjunto de teste e conjunto de validação. Posteriormente, as árvores de decisão serão criadas e treinadas com os dados do conjunto de treino. Comparando os resultados do erro médio quadrático obtido com cada floresta, será escolhida a floresta com o menor erro. De seguida, as previsões serão calculadas na floresta escolhida com o conjunto de dados de teste. O conjunto de dados de validação não faz parte do processo de treino do RF, mas é uma parte importante no processo de avaliação do desempenho do modelo, pois tem como objetivo avaliar o modelo e generalizá-lo para novos dados ainda não vistos. Por fim, como estamos perante um problema de regressão, a previsão final é baseada na média das previsões de cada árvore.

O código utilizado em MATLAB para desenvolver este método encontra-se no Anexo A.

4.3 Resultados e Análise

Depois de devidamente selecionados todos os dados e variáveis para o estudo de cada ativo com cada método de otimização, vamos avaliar o seu desempenho. Para isso, começamos por testar quais as variáveis preditoras que melhor se ajustam aos dados de cada um dos ativos. As variáveis que obtiverem os melhores resultados no método da regressão linear, serão usadas igualmente no método de *Random Forest* pois, apesar do próprio RF realizar a sua seleção de variáveis preditoras, desta forma podemos fazer uma comparação mais justa com a regressão linear.

4.3.1 Resultados da regressão linear

Com o código MATLAB analisado anteriormente relativo ao método da regressão linear, fomos testar quais as combinações de variáveis preditoras que melhor se ajustam a cada um dos ativos. O melhor conjunto será aquele que teve melhor desempenho (menor erro no *out-of-sample*). Assim, com base nas Tabelas B.1 e B.2 que se encontram no Anexo B, construímos o resumo apresentado na Tabela 4.3.

Ativo	**Conjunto variáveis preditoras**	**Erro *out-of-sample***
ALTR	$x1, x2$ e p	1.03211E-03
BCP	$x1$ e p	9.20287E-04
NOS	$x1, u$ e p	1.53104E-04
NVG	$x1, u$ e p	2.41546E-04
SON	$x1, u$ e p	2.09792E-04

Tabela 4.3 Resultados no *out-of-sample* dos melhores conjuntos de variáveis preditoras em cada um dos ativos (fase 1).

De seguida, iremos ver se o comportamento destes conjuntos de variáveis da fase 1 com as variáveis *volume*, *hl* e *co* melhora os resultados. Com os resultados obtidos pelas Tabelas B.3 e B.4 obtivemos o resumo apresentado na Tabela 4.4.

Ativo	**Conjunto variáveis preditoras**	**Erro *out-of-sample***
ALTR	$x1, x2, p$ e *volume*	1.03193E-03
BCP	$x1, p$ e *co*	9.20043E-04
NOS	-	-
NVG	-	-
SON	-	-

Tabela 4.4 Resultados dos erros no *out-of-sample* com a adição de novas variáveis preditoras (fase 2).

Após a verificação do melhor conjunto de variáveis preditoras para cada um dos ativos, classificamos os valores previstos, isto é, vimos se estes eram retornos positivos ou negativos (classificação). A Tabela 4.5 reporta a taxa de sucesso na previsão dos retornos no *out-of-sample*.

	ALTR	**BCP**	**NOS**	**NVG**	**SON**
Taxa (%)	46.00	43.68	45.77	45.20	43.34

Tabela 4.5 Taxa de sucesso na previsão dos retornos no *out-of-sample* de cada ativo.

Analisemos o caso específico do ativo ALTR, sendo que a análise dos resultados é feita de forma análoga para os restantes ativos.

Com os dados do *in-sample*, foram obtidos os pesos que minimizavam a soma dos quadrados dos erros associados às variáveis preditoras que melhor se ajustavam aos dados históricos. Posteriormente, esses pesos foram utilizados nos dados *out-of-sample*. Os valores desses pesos estão indicados na tabela seguinte:

Termo independente	$x1$	$x2$	p
0.000032	-0.016433	0.083550	0.043972

Tabela 4.6 Valores dos pesos calculados no *in-sample* do ativo ALTR.

Com estes dados, verificamos que a média móvel dos 14 dias, $x2$, é a variável preditora que maior influência tem na previsão dos retornos do ativo. As variáveis $x2$ e p apresentam pesos positivos o que indica que a correlação com os retornos do ativo é positiva. Por outro lado, $x1$ tem associado um peso negativo o que indica uma correlação negativa.

Observemos os gráficos associados ao ativo ALTR, onde conseguimos ver a influência dos pesos e uma parte do seu comportamento tanto no *in-sample* (Figura 4.3) como no *out-of-sample* (Figura 4.4).

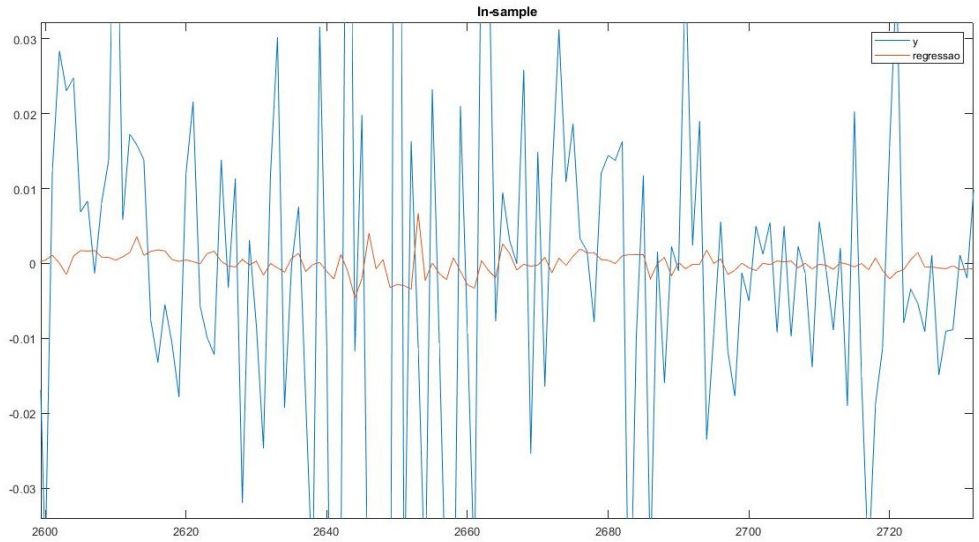


Fig. 4.3 Representação gráfica do estudo feito no *in-sample* de 15/01/2010 a 21/07/2010, do ativo ALTR.

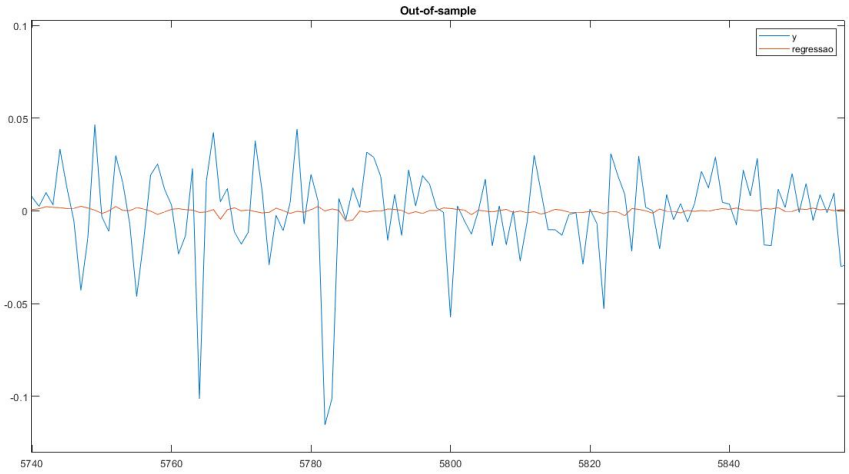


Fig. 4.4 Representação gráfica dos resultados obtidos no *out-of-sample* de 21/04/2022 a 30/09/2022, do ativo ALTR.

Na fase *in-sample* é possível verificar que, com o estudo dos dados e consequentemente o cálculo dos pesos, a linha vermelha associada ao método da regressão tenta ajustar-se bem aos dados históricos. Na fase *out-of-sample*, vemos a comparação entre os dados originais e os dados previstos pelo algoritmo e observamos que a linha vermelha tem a tendência em acompanhar a linha azul. O erro associado é de 1.03193E-03, o que mostra um bom desempenho.

De seguida fomos calcular a média móvel de cada uma das variáveis (neste caso $x1, x2$ e p) e comparamos com a média de todo o *in-sample*, uma vez que é a estimativa clássica utilizada no modelo Markowitz. A Tabela 4.7 apresenta os erros cometidos no período *out-of-sample* quando são utilizadas cada uma das estimativas heurísticas descritas.

Média *in-sample*	$x1$	$x2$	p
0.234366	0.280354	0.253390	0.253390

Tabela 4.7 Resultados relativos ao erro obtido com as médias móveis $x1, x2$ e ao penúltimo dia no ativo ALTR.

Comparando estes valores com o valor da média de todo o *in-sample* percebemos que apesar das alterações, o valor médio do *in-sample* continua a ser melhor.

4.3.2 Resultados do *Random Forest*

Primeiramente fomos trabalhar com o código MATLAB associado ao RF por classificação com o objetivo de prever se o retorno será negativo ou positivo. Inicialmente teremos a classificação dos dados iniciais, ou seja, vemos se o dado é positivo (1) ou negativo (0). Foram novamente utilizados os melhores conjuntos de variáveis preditoras de cada ativo (fase 1 ou fase 2) e, com isto, decidimos construir 20 árvores de decisão de acordo com os conjuntos de dados de treino e teste e obtivemos os resultados de cada uma. Como estamos perante um problema de classificação, a previsão final será o resultado que mais vezes se repete.

Os dados iniciais são divididos de forma aleatória pelos conjuntos de treino e teste, portanto sempre que formos testar, os resultados serão diferentes. A Tabela 4.8 apresenta as taxas de sucesso obtidas com este método em cada um dos ativos analisados.

	ALTR	**BCP**	**NOS**	**NVG**	**SON**
Taxa (%)	45.55	44.54	46.22	45.78	43.63

Tabela 4.8 Taxa de sucesso na previsão dos retornos de cada ativo.

Novamente iremos fazer o estudo mais específico do ativo ALTR. Das 20 árvores construídas, fomos ver o comportamento de uma das árvores.

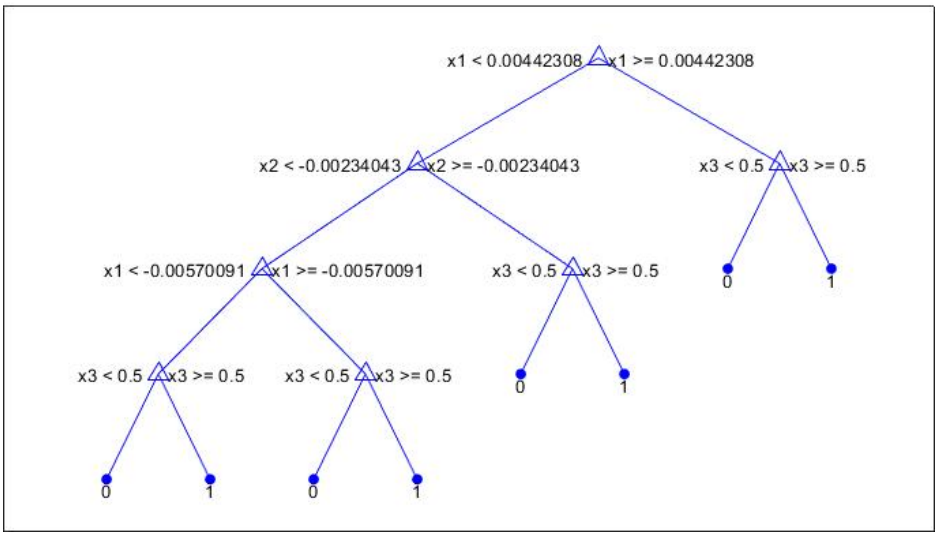


Fig. 4.5 Representação gráfica de uma árvore de decisão do RF por classificação, para os dados do ativo ALTR.

Primeiramente, o algoritmo selecionou a melhor variável preditora para o nó-raiz, que neste caso foi o $x1$. Suponhamos que o dado *input* era menor que 0.00442308. Então ele seguia pelo ramo até chegar ao nó-filho representado pela variável $x2$. Aqui verificamos que o dado de $x2$ é maior ou igual a -0.00234043. Novamente segue pelo ramo consequente. É de realçar que não é assim tão simples interpretar o modelo. Além disso, a primeira variável escolhida nesta árvore foi a variável $x1$ e vimos que na regressão linear ela tem um peso inferior a $x2$. Além disso, no RF são geradas várias árvores (neste caso 20) e nas outras a primeira variável escolhida pode ser diferente desta.

Pela Tabela 4.8, verificamos que a taxa de sucesso obtido com o algoritmo RF usando 20 árvores foi de 45.55%. O gráfico do erro associado à previsão do sinal dos retornos *out-of-bag* conforme a construção das árvores é o seguinte:

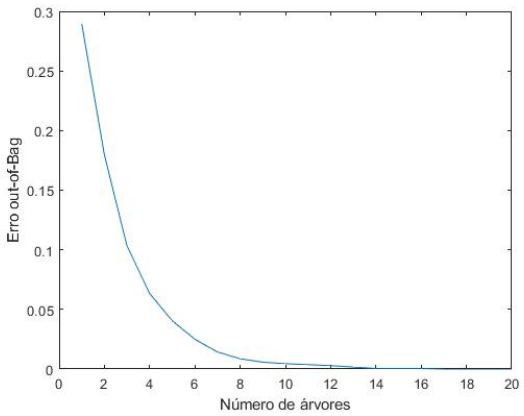


Fig. 4.6 Gráfico que apresenta o erro *out-of-bag* em função do número de árvores construídas.

Por fim, fomos estudar o código *Random Forest* para a regressão. Com este algoritmo pretendemos prever o valor dos retornos de cada ativo. Iniciamos com a divisão dos dados pelos conjuntos de treino, teste e validação e de seguida as árvores foram construídas. Escolhemos testar com 20 árvores de decisão. Na Tabela 4.9 , podemos observar um exemplo dos valores dos erros médios quadráticos e a taxa de sucesso em cada um dos ativos.

	ALTR	**BCP**	**NOS**	**NVG**	**SON**
MSE	0.8800	0.2417	0.2005	0.1359	0.3728
Taxa (%)	90.20	90.37	93.11	85.98	84.33

Tabela 4.9 Erro médio quadrático e taxa de sucesso de cada ativo.

Para o ativo ALTR, temos um gráfico onde pode ser comparado o comportamento dos retornos reais e os previstos.

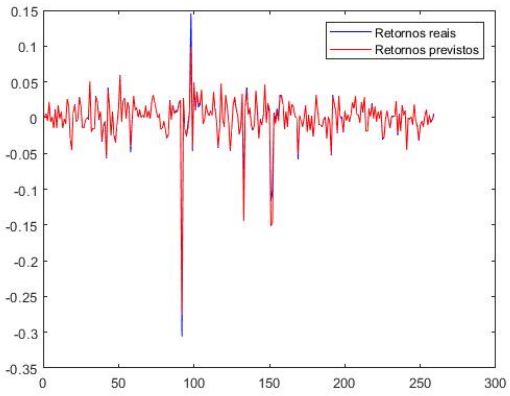


Fig. 4.7 Gráfico de comparação entre os retornos reais e os previstos no período *out-of-sample* com método da regressão, com os dados do ativo ALTR.

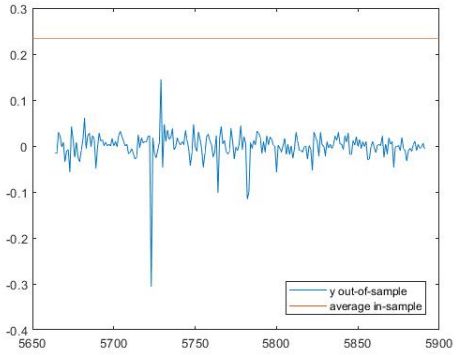
É de notar que as previsões obtidas tiveram ótimos resultados, uma vez que se pode ver pelo gráfico que estas se ajustaram muito aos valores reais.

4.3.3 Comparação da *performance*

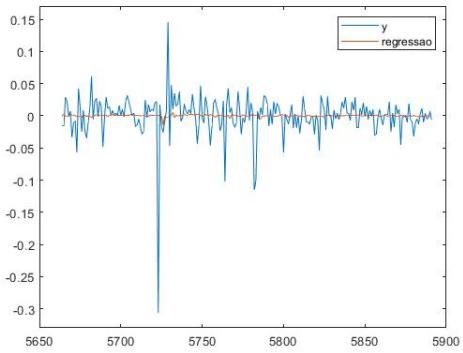
Comparando o desempenho obtido pelo algoritmo da regressão linear quanto à parte da classificação dos retornos (retorno do dia seguinte terá o mesmo sinal que a sua previsão) com o desempenho obtido pelo algoritmo de *Random Forest* por classificação, podemos dizer que as diferenças não são significativas. Pelas Tabelas 4.7 e 4.8, verificamos que as taxas de sucesso são muito idênticas.

Quanto à parte do problema de regressão, conseguimos notar grandes diferenças no desempenho das previsões dos retornos entre a regressão linear e o *Random Forest*.

De maneira a ser mais percetível a diferença entre os três métodos, Markowitz, regressão linear e *random forest*, apresentamos de seguida os gráficos que refletem os seus desempenhos:



(a) Gráfico dos retornos reais do *out-of-sample* e average *in-sample*.



(b) Gráfico dos retornos reais do *out-of-sample* e valores da previsão dos retornos.

Fig. 4.8 Gráficos de desempenho no método de regressão linear.

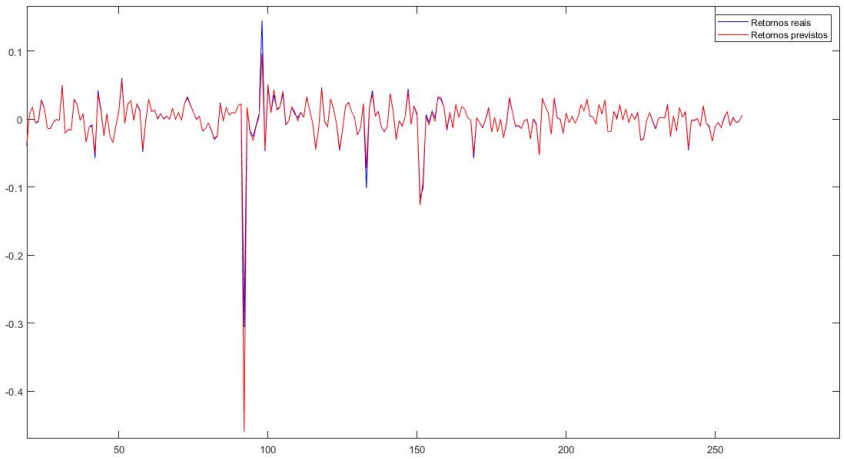


Fig. 4.9 Gráfico dos retornos reais e dos retornos previstos no método de RF por regressão no *out-of-sample*.

Fazendo a comparação das taxas de sucesso entre as Tabelas 4.5 e 4.9 verificamos que o algoritmo de *machine learning* teve resultados muito bons, chegando a ter taxas de sucesso de 90%.

Capítulo 5

Conclusão

Os métodos de otimização, como dito anteriormente, são métodos muito explorados na área das finanças nomeadamente para o estudo da previsão de retornos de ativos. Devido à necessidade de se trabalhar com um grande número de dados históricos e ainda, ter em consideração o impacto de vários fatores externos nos preços dos ativos, a previsão dos retornos tornou-se mais difícil com os métodos tradicionais. Os algoritmos de *machine learning* começaram a ser desenvolvidos e frequentemente utilizados nesta área para a dificuldade ser ultrapassada.

O estudo dos retornos pode ser feito de duas formas sendo elas com um problema de classificação para a previsão do sinal dos retornos e com um problema de regressão, onde é possível prever os valores numéricos dos retornos dos ativos. Com o objetivo de testar o desempenho entre um método de otimização clássico com um método de *machine learning*, fomos estudar o comportamento dos dados históricos de ativos pertencentes ao PSI com os algoritmos de regressão linear, RF por classificação e RF por regressão.

Desta forma, com os resultados obtidos, foi possível verificar que o desempenho e a taxa de sucesso do RF é melhor para a previsão de retornos de ativos. Observamos que todo o procedimento que envolve o algoritmo de *Random Forest* desde a divisão aleatória dos dados históricos em conjuntos de treino, teste e validação, o uso dos critérios de divisão de nó e ainda o uso do método *bagging*, fazem com que este método seja mais preciso e eficaz.

Embora os nossos resultados tenham sido razoáveis nos testes, a diferença da taxa de sucesso entre a regressão linear e o RF por classificação, não foi muito notória. Num próximo estudo, poderíamos começar por alterar o cálculo do erro. Em vez de calcularmos o erro absoluto, iríamos calcular o erro relativo e ver se obtínhamos melhores resultados.

Num trabalho futuro, para um estudo mais rigoroso de cada método, poderão ser introduzidas mais condicionantes devido aos fatores externos que afetam os preços diários e também mais métricas de avaliação. Para além disso, a previsão dos retornos dos ativos poderá ser feito em simultâneo. Assim poderá haver uma comparação mais direta e os ativos poderão ser correlacionados.

Bibliografia

[1] Khalid Alkhatib, Hassan Najadat, Ismail Hmeidi, and Mohammed K Ali Shatnawi. Stock price prediction using k-nearest neighbor (knn) algorithm. *International Journal of Business, Humanities and Technology*, 3(3):32–44, 2013.

[2] Keith D. Copsey Andrew R. Webb. *Statistical Pattern Recognition*. John Wiley Sons, 2011.

[3] Christopher N. Avery, Judith A. Chevalier, and Richard J. Zeckhauser. The "CAPS" Prediction System and Stock Market Returns*. *Review of Finance*, 20(4):1363–1381, 2015. URL https://doi.org/10.1093/rof/rfv043.

[4] Suryoday Basak, Saibal Kar, Snehanshu Saha, Luckyson Khaidem, and Sudeepa Roy Dey. Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance*, 47:552–567, 2019. URL https://www.sciencedirect.com/science/article/pii/S106294081730400X.

[5] Richard A. Berk. *Random Forests*. Springer New York, 2008. URL https://doi.org/10.1007/978-0-387-77501-2_5.

[6] Mehdi Beyhaghi and James P. Hawley. Modern portfolio theory and risk management: assumptions and unintended consequences. *Journal of Sustainable Finance & Investment*, 3(1):17–37, 2013. URL https://doi.org/10.1080/20430795.2012.738600.

[7] Leo Breiman. Random Forest. *Machine Learning*, 45:5–32, 2001. URL https://doi.org/10.1023/A:1010933404324.

[8] Mariana Serrano Lopes da Bernanda. Portfolio optimization: From markowitz to machine learning. Master's thesis, Universidade Nova de Lisboa, Instituto Superior de Estatística e Gestão de Informação, NOVA Information Management School, 2021.

[9] Yuqing Dai and Yuning Zhang. Machine learning in stock price trend forecasting. *Stanford University*, 21, 2013.

[10] Xinjie Di. Stock trend prediction with technical indicators using svm. *Stanford University*, 2011.

[11] Lior Rokach e Oded Maimon. *Data mining with decision trees: theory and applications*, volume 81. World scientific, 2014.

[12] Leandro Fleck, Maria Hermínia Ferreira Tavares, Eduardo Eyng, Andrieli Cristina Helmann, and MA de M Andrade. Redes neurais artificiais: Princípios básicos. *Revista Eletrônica Científica Inovação e Tecnologia*, 1(13):47–57, 2016.

[13] Cláudio Frizzarini. Algoritmo para indução de Árvores de classificação para dados desbalanceados. Master's thesis, Universidade de São Paulo, 2013.

[14] Ramazan Gençay. Linear, non-linear and essential foreign exchange rate prediction with simple technical trading rules. *Journal of International Economics*, 47(1):91–107, 1999. URL https://www.sciencedirect.com/science/article/pii/S0022199698000178.

[15] Pascal Geurts and Gilles Louppe. Learning to rank with extremely randomized trees. *Journal of Machine Learning Research - Proceedings Track*, pages 49–61. URL https://www.researchgate.net/publication/220320879_Learning_to_rank_with_extremely_randomized_trees.

[16] Chenn-Jung Huang, Dian-Xiu Yang, and Yi-Ta Chuang. Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Systems with Applications*, 34(4): 2870–2878, 2008. URL https://www.sciencedirect.com/science/article/pii/S0957417407001819.

[17] Yongkang Huang and Hongzhi Yu. Research on text generation techniques combining machine learning and deep learning. *IPEC '22: Proceedings of the 3rd Asia-Pacific Conference on Image Processing, Electronics and Computers*, 2022. URL https://doi.org/10.1145/3544109.3544168.

[18] Thi Thuy Le Dao Thi Nhat Hung Dang, Ngoc Van Vu. Información contable y rendimiento de acciones en el mercado de valores de vietnam: enfoque de aprendizaje automático. *Contabilidad y Negocios*, 17(33), 2022. URL https://doi.org/10.18800/contabilidad.202201.004.

[19] Michael C. Jensen. Some anomalous evidence regarding market efficiency. *Journal of Financial Economics*, 6(2):95–101, 1978. URL https://www.sciencedirect.com/science/article/pii/0304405X78900259.

[20] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. Predicting the direction of stock market prices using random forest. *ArXiv*, 2016. URL https://arxiv.org/pdf/1605.00003.pdf.

[21] W Khan, MA Ghazanfar, M Asam, A Iqbal, S Ahmad, and Javed Ali Khan. Predicting trend in stock market exchange using machine learning classifiers. *Science International*, 28(2), 2016.

[22] Indu Kumar, Kiran Dogra, Chetna Utreja, and Premlata Yadav. A comparative study of supervised machine learning algorithms for stock market trend prediction. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1003–1007, 2018. doi: 10.1109/ICICCT.2018.8473214.

[23] Haoming Li, Zhijun Yang, and Tianlun Li. Algorithmic trading strategy based on massive data mining. 2014. URL https://api.semanticscholar.org/CorpusID:15348103.

[24] Burton G. Malkiel. The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, 17(1):59–82, 2003. URL https://www.aeaweb.org/articles?id=10.1257/089533003321164958.

[25] Dejan Velušček Marija Gorenc Novak. Prediction of stock price movement based on daily high prices. *Quantitative Finance*, 16(5):793–856, 2016.

[26] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. URL http://www.jstor.org/stable/2975974.

[27] Sergio Reinaldo Marteletto. Técnicas de seleção de atributos através de *Random Forest*: um estudo de caso para detecção de tendências em séries temporais financeiras. Master's thesis, Universidade de São Paulo, Escola de Artes, Ciências e Humanidades, 2022.

[28] Javier M. Moguerza and Alberto Muñoz. Support vector machines with applications. *Statistical Science*, 21(3):322–336, 2006. URL http://www.jstor.org/stable/27645765.

[29] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. C4.5: Programs for Machine Learning, 1993.

[30] Sruthi E R. Understanding random forest. *Data Science Blogathon - Published On June 17 2021 and Last Modified On November 30th 2022*. URL https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

[31] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal*, 3(3):535–554, 1959. URL https://www.cs.virginia.edu/~evans/greatworks/samuel1959.pdf.

[32] Erwan Scornet. On the asymptotics of random forests. *J. Multivar. Anal.*, 146:72–83, 2014. URL https://api.semanticscholar.org/CorpusID:35064889.

[33] Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015. URL http://www.jstor.org/stable/43556658.

[34] Yauheniya Shynkevich, T.M. McGinnity, Sonya A. Coleman, Ammar Belatreche, and Yuhua Li. Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*, 264:71–88, 2017. URL https://www.sciencedirect.com/science/article/pii/S0925231217311074. Machine learning in finance.

[35] Dingming Wu, Xiaolong Wang, Jingyong Su, Buzhou Tang, and Shaocong Wu. A labeling method for financial time series prediction based on trends. *Entropy*, 22(10):1162, 2020.

[36] Enes Zvornicanin. Out-of-bag error in random forests, 2023. https://www.baeldung.com/cs/random-forests-out-of-bag-error consultado em 31/08/2023.

Anexo A

Códigos MATLAB

Código do método da regressão linear

```
function RegressaoLinear()
clc; clear; close all;

% Dados do excel ALTR
% y-retorno ln; x1-duracao de 7 dias; x2-duracao de 14 dias; x3-
    duracao de 21 dias; x4-duracao de 28 dias; hl=high-low;
%co=close-open

dimAmostraIN = 5634; %4/01/2000 ate 31/12/2021 (comecar quando
    houver dados de x4)
ultimaLinhaExcel = 5920;
primeiraLinha = 30;
ultimaLinha = primeiraLinha + dimAmostraIN - 1;
fileName = 'c:\Users\taisr\Desktop\empresasescolhidas\ALTR.LS.
    xlsx';
y = xlsread(fileName,['O',num2str(primeiraLinha),':O',num2str(
    ultimaLinhaExcel)]);
x1 = xlsread(fileName,['K',num2str(primeiraLinha),':K',num2str(
    ultimaLinhaExcel)]);
x2 = xlsread(fileName,['L',num2str(primeiraLinha),':L',num2str(
    ultimaLinhaExcel)]);
x3 = xlsread(fileName,['M',num2str(primeiraLinha),':M',num2str(
    ultimaLinhaExcel)]);
x4 = xlsread(fileName,['N',num2str(primeiraLinha),':N',num2str(
    ultimaLinhaExcel)]);
volume = xlsread(fileName,['G',num2str(primeiraLinha),':G',
    num2str(ultimaLinhaExcel)]);
hl = xlsread(fileName,['Q',num2str(primeiraLinha),':Q',num2str(
    ultimaLinhaExcel)]);
```

```
20 co = xlsread(fileName,['R',num2str(primeiraLinha),':R',num2str(
   ultimaLinhaExcel)]);
21
22 a0 = [0,0,0,0,0,0,0,0,0,0];
23
24     disp(['In-sample'])
25     tempo = 4:dimAmostraIN;
26     X = calcularX(tempo-1);
27     [aOpt,fval] = fminsearch(@funcaoObj,a0);
28     fprintf('aOpt: %f\n', aOpt) %pesos
29     fprintf('fval: %f\n', fval) %valor da funcao objetivo
30     erroInSample = fval/dimAmostraIN;
31     fprintf('erroInSample = %E\n', erroInSample);
32
33     figure(1)
34     %grafico com a apresentacao dos dados historicos e com os
          valores da previsao
35     plot(tempo', [y(tempo), previsao(aOpt)])
36     legend('y', 'regressao')
37     title('In-sample')
38     [mean(y(tempo)) mean(previsao(aOpt))];
39
40
41     disp(['Out-of-Sample'])
42     tempo = ultimaLinha+1:length(y); %03/01/2022 ate 30/12/2022
43     X = calcularX(tempo-1);
44     funcaoObj(aOpt)
45     erroOutOfSample = funcaoObj(aOpt)/(length(y)-ultimaLinha);
46     fprintf('erroOutOfSample = %E\n', erroOutOfSample);
47
48     tempoInSample = 4:dimAmostraIN;
49     %aAux - vetor que ira guardar os valores dos pesos
          calculados
50     aAux = zeros(1, length(aOpt));
51     %aAux (1) - media dos retornos do in-sample associado ao
          termo indepedendente
52     aAux(1) = mean(y(tempoInSample));
53     [mean(y(tempoInSample)) mean(y(tempo))];
54     fprintf('Average InSample: %f\n', funcaoObj(aAux))
55
56     %plot dos retornos historicos do out-of-sample com o average
          do in-sample
```

```
57      figure(2)
58      plot(tempo',y(tempo));
59      hold on;
60      plot(xlim,funcaoObj(aAux)*[1,1]);
61      hold off
62      legend('y out-of-sample','average in-sample')
63
64
65      %aAux(2) - x1 tera o valor de 1 no vetor aAux, enquanto que
           as restantes variaveis terao o valor de 0 (calculo da
           media movel dos 7 dias)
66      aAux = zeros(1, length(aOpt)); aAux(2) = 1;
67      fprintf('Moving average 7 dias: %f\n', funcaoObj(aAux))
68      aAux = zeros(1, length(aOpt)); aAux(3) = 1;
69      fprintf('Moving average 14 dias: %f\n', funcaoObj(aAux))
70      aAux = zeros(1, length(aOpt)); aAux(4) = 1;
71      fprintf('Moving average 21 dias: %f\n', funcaoObj(aAux))
72      aAux = zeros(1, length(aOpt)); aAux(5) = 1;
73      fprintf('Moving average 28 dias: %f\n', funcaoObj(aAux))
74      aAux = zeros(1, length(aOpt)); aAux(end-1) = 1;
75      fprintf('ultimo dias: %f\n', funcaoObj(aAux))
76      aAux = zeros(1, length(aOpt)); aAux(end-2) = 1;
77      fprintf('penultimo dias: %f\n', funcaoObj(aAux))
78
79      %taxa de sucesso
80      taxa = sum(y.*funcaoObj(aAux)>0)/length(y)
81
82      figure(3)
83      plot(tempo',[y(tempo),previsao(aOpt)])
84      legend('y','regressao')
85      title('Out-of-sample')
86
87      function z = funcaoObj(a)
88          %erro entre o valor real e o valor previsto
89          z = norm(y(tempo)-(previsao(a)))^2;
90      end
91
92      function z = previsao(a)
93          %termo independente + pesos*variaveis preditoras
94          z = a(1) + X*a(2:end)';
95      end
96
```

```
97      function X = calcularX(tempo)
98          %conjunto de variaveis preditoras
99          X = [x1(tempo-1) x2(tempo-1) x3(tempo-1) x4(tempo-1)
100         y(tempo-1) y(tempo-2) volume(tempo-1) hl(tempo-1)
101         co(tempo-1)];
102     end
103 end
```

Código do método *Random Forest* para o problema de classificação

```
function RF_classificacao()
clc; clear; close all;

%carregando os dados
dimAmostraIN = 5634; %4/01/2000 ate 31/12/2021
ultimaLinhaExcel = 5920;
primeiraLinha = 30;
fileName = 'c:\Users\taisr\Desktop\empresas escolhidas\ALTR.LS.xlsx';
y = xlsread(fileName,['O',num2str(primeiraLinha),':O',num2str(ultimaLinhaExcel)]);

%passamos todos os valores dos retornos a zero
%os valores de y que forem positivos terao o valor 1 associado. Os valores restantes serao 0
% 1 - retorno positivo; 0 - retorno negativo
y1=y*0;
y1(y>0)=1;
y=y1;

x1 = xlsread(fileName,['K',num2str(primeiraLinha),':K',num2str(ultimaLinhaExcel)]);
x2 = xlsread(fileName,['L',num2str(primeiraLinha),':L',num2str(ultimaLinhaExcel)]);
x3 = xlsread (fileName,['M',num2str(primeiraLinha),':M',num2str(ultimaLinhaExcel)]);
x4 = xlsread(fileName,['N',num2str(primeiraLinha),':N',num2str(ultimaLinhaExcel)]);
volume = xlsread(fileName,['G',num2str(primeiraLinha),':G',num2str(ultimaLinhaExcel)]);
hl = xlsread(fileName,['Q',num2str(primeiraLinha),':Q',num2str(ultimaLinhaExcel)]);
co = xlsread(fileName,['R',num2str(primeiraLinha),':R',num2str(ultimaLinhaExcel)]);


tempo = 4:dimAmostraIN;
X = calcularX(tempo-1);

frac_treino = 0.75; %conjunto de treino
frac_teste = 0.25; %conjunto de teste
```

```
32 if frac_teste+frac_treino~=1
33     error('Soma tem que dar 1');
34 end
35
36 %divisao aleatoria dos dados iniciais pelos conjuntos de treino
      e teste
37 [treinoInd,testeInd] = dividerand(size(X,1),frac_treino,
      frac_teste);
38
39 X_treino = X(treinoInd,:);
40 Y_treino = y(treinoInd,:);
41 X_teste = X(testeInd,:);
42 Y_teste = y(testeInd,:);
43
44 %treino e construcao de 20 arvores com a ferramenta do MATLAB
      TreeBagger para um problema de classificacao
45 %OOBpredict - previsao feita com os dados out-of-bag
46 RFmodel = TreeBagger(20, X_treino, Y_treino, 'Method', '
      classification', OOBPrediction="on");
47 %representacao da arvore 20
48 view(RFmodel.Trees{1},Mode="graph")
49 %lista dos nos e ramos (tomadas de decisoes e consequencias) da
50 %arvore 20
51 view(RFmodel.Trees{1});
52
53 Y_pred = RFmodel.predict(X_teste);
54 %passar as previsoes do tipo char para numero
55 c = str2double(Y_pred);
56
57 taxa = sum(Y_teste.*c>0)/length(Y_teste)
58
59
60 %plot do erro de classificacao do out-of-bag sobre o numero de
      arvores de decisao
61 %Erro OOB e o numero de classificacoes incorretas da amostra OOB
      .
62 figure(1)
63 plot(oobError(RFmodel))
64 xlabel("Numero de arvores")
65 ylabel("Erro Out-of-Bag")
66 %o erro out-of-bag diminui a medida que o numero de arvores
      aumenta
```

```
67 %prever observacoes fora do out-of-bag
68 %serao apresentados resultados para um conjunto aleatorio de 20
      observacoes
69 oobLabels = oobPredict(RFmodel);
70 %vetor com 20 observacoes escolhidas aleatoriamente
71 nArv = 20;
72 ind = randsample(length(oobLabels),nArv);
73 %passagem do tipo celula para string e depois para inteiro
74 aux = str2num(cell2mat(oobLabels));
75 table(y(ind),aux(ind),VariableNames=["Valores verdadeiros" "
      Valores previstos"])
76 suc = aux(ind) - y(ind);
77 %taxa de sucesso
78 taxaFora = sum(suc == 0)/nArv
79
80     function X = calcularX(tempo)
81         %conjunto de variaveis preditoras
82         X = [x1(tempo-1) x2(tempo-1) x3(tempo-1) x4(tempo-1) y(
               tempo-1) y(tempo-2) volume(tempo-1) hl(tempo-1) co(
               tempo-1)];
83     end
84 end
```

Código do método *Random Forest* para o problema de regressão

```
function RF_regressao()
clc; clear; close all;

% Carregando os dados
dimAmostraIN = 5634; %4/01/2000 ate 31/12/2021
ultimaLinhaExcel = 5920;
primeiraLinha = 30;
fileName = 'c:\Users\taisr\Desktop\empresas escolhidas\ALTR.LS.xlsx';
y = xlsread(fileName,['O',num2str(primeiraLinha),':O',num2str(ultimaLinhaExcel)]);
x1 = xlsread(fileName,['K',num2str(primeiraLinha),':K',num2str(ultimaLinhaExcel)]);
x2 = xlsread(fileName,['L',num2str(primeiraLinha),':L',num2str(ultimaLinhaExcel)]);
x3 = xlsread(fileName,['M',num2str(primeiraLinha),':M',num2str(ultimaLinhaExcel)]);
x4 = xlsread(fileName,['N',num2str(primeiraLinha),':N',num2str(ultimaLinhaExcel)]);
volume = xlsread(fileName,['G',num2str(primeiraLinha),':G',num2str(ultimaLinhaExcel)]);
hl = xlsread(fileName,['Q',num2str(primeiraLinha),':Q',num2str(ultimaLinhaExcel)]);
co = xlsread(fileName,['R',num2str(primeiraLinha),':R',num2str(ultimaLinhaExcel)]);

tempo = 4:dimAmostraIN;
X = calcularX(tempo-1);

%dividir os dados em conjuntos de treino e teste
frac_treino = 0.75;
frac_teste = 0.25;
if frac_teste+frac_treino~=1
    error('Soma tem que dar 1');
end

%divisao aleatoria dos dados iniciais pelos conjuntos de treino, teste e validacao
[treinoInd,testeInd] = dividerand(size(X,1),frac_treino,frac_teste);
valInd = (size(X, 1)+1):ultimaLinhaExcel-primeiraLinha;
```

```
31
32 X_treino = X(treinoInd,:);
33 Y_treino = y(treinoInd,:);
34 X_teste = X(testeInd,:);
35 Y_teste = y(testeInd,:);
36
37 %dados de validacao do out-of-sample
38 X = calcularX(3:ultimaLinhaExcel-primeiraLinha+2);
39 X_val = X(valInd,:);
40 Y_val = y(valInd,:);
41
42 MSEOpt = Inf;
43 nArv = 20;
44 for i=1:nArv
45     %treino e construcao com o modelo Random Forest aplicado ao
          problema de regressao de 20 arvores
46     RFmodel = TreeBagger(i, X_treino, Y_treino, 'Method', '
          regression');
47
48     %avalia o modelo com os dados do conjunto de validacao
49     Y_pred = predict(RFmodel,X_teste);
50
51     %erro quadratico medio (MSE)
52     MSE = norm(Y_teste-Y_pred,2);
53
54     if MSE<MSEOpt
55         MSEOpt=MSE;
56         RFmodelOpt = RFmodel;
57     end
58 end
59
60 %representacao e descricao da arvore de decisao 20
61 view(RFmodel.Trees{18},Mode="graph");
62 view(RFmodel.Trees{18});
63
64 %faz as previsoes com o conjunto de teste
65 Y_pred = predict(RFmodel,X_val);
66
67 %coeficiente de determinacao R2
68 R2 = corr(Y_val,Y_pred).^2;
69 %erro quadratico medio (MSE)
70 MSE = norm(Y_val-Y_pred,2)
```

```
71
72 %plot dos resultados
73 plot(Y_val,'b');
74 hold on;
75 plot(Y_pred,'r');
76 legend('Retornos reais','Retornos previstos');
77
78 %taxa de sucesso
79 taxa = sum(Y_val.*Y_pred>0)/length(Y_val)
80
81 function X = calcularX(tempo)
82      %conjunto de variaveis preditoras
83      X = [x1(tempo-1) x2(tempo-1) x3(tempo-1) x4(tempo-1)
84      y(tempo-2) y(tempo-1) volume(tempo-1) hl(tempo-1)
85      co(tempo-1)];
86     end
87 end
```

Anexo B

Tabelas de resultados - regressão linear

Variáveis preditoras	ALTR		BCP		NOS	
	EI	EO	EI	EO	EI	EO
$x1$ e u	1.68996E-03	1.03563E-03	6.76152E-04	9.22798E-04	3.87986E-04	1.53809E-04
$x1$ e p	1.68781E-03	1.03264E-03	6.76603E-04	9.20287E-04	3.89038E-04	1.53605E-04
$x1$, u e p	1.68755E-03	1.03354E-03	6.76152E-04	9.22826E-04	3.88643E-04	1.53104E-04
$x1$,$x2$ e u	1.68952E-03	1.03506E-03	6.76153E-04	9.22593E-04	3.87808E-04	1.53706E-04
$x1$,$x2$ e p	1.68734E-03	1.03211E-03	6.76603E-04	9.20290E-04	3.88286E-04	1.53523E-04
$x1$,$x2$, u e p	1.68708E-03	1.03297E-03	6.76152E-04	9.22914E-04	3.87732E-04	1.53443E-04
$x1$,$x2$,$x3$ e u	1.68945E-03	1.03577E-03	6.75037E-04	9.43152E-04	3.87518E-04	3.87518E-04
$x1$,$x2$,$x3$ e p	1.68727E-03	1.03273E-03	6.75622E-04	9.37277E-04	3.88390E-04	1.53707E-04
$x1$,$x2$,$x3$, u e p	1.68702E-03	1.03370E-03	6.75947E-04	9.30062E-04	3.87513E-04	1.53777E-04
$x1$,$x2$,$x3$,$x4$ e u	1.68937E-03	1.03600E-03	6.74588E-04	9.43249E-04	3.87509E-04	1.53954E-04
$x1$,$x2$,$x3$,$x4$ e p	1.68719E-03	1.03297E-03	6.76598E-04	9.20353E-04	3.88264E-04	1.53502E-04
$x1$,$x2$,$x3$,$x4$, u e p	1.68694E-03	1.03408E-03	6.75076E-04	9.37156E-04	3.87862E-04	1.53272E-04

Tabela B.1 Resultados dos erros no *in-sample* (EI) e no *out-of-sample* (EO) no método da regressão linear dos ativos ALTR, BCP e NOS (fase 1).

Variáveis preditoras	NVG		SON	
	EI	EO	EI	EO
$x1$ e u	2.62461E-04	2.41854E-04	4.29472E-04	2.09793E-04
$x1$ e p	2.62461E-04	2.41860E-04	4.29498E-04	2.10131E-04
$x1$, u e p	2.62359E-04	2.41546E-04	4.29472E-04	2.09792E-04
$x1$,$x2$ e u	2.62461E-04	2.41847E-04	4.29310E-04	2.10207E-04
$x1$,$x2$ e p	2.62287E-04	2.41967E-04	4.29338E-04	2.10557E-04
$x1$,$x2$, u e p	2.62285E-04	2.41986E-04	4.29310E-04	2.10215E-04
$x1$,$x2$,$x3$ e u	2.62461E-04	2.41851E-04	4.29019E-04	2.10105E-04
$x1$,$x2$,$x3$ e p	2.62266E-04	2.42661E-04	4.29049E-04	2.10461E-04
$x1$,$x2$,$x3$, u e p	2.62266E-04	2.42443E-04	4.29019E-04	2.10111E-04
$x1$,$x2$,$x3$,$x4$ e u	2.62461E-04	2.41847E-04	4.29015E-04	2.09943E-04
$x1$,$x2$,$x3$,$x4$ e p	2.62173E-04	2.43398E-04	4.29045E-04	2.10314E-04
$x1$,$x2$,$x3$,$x4$, u e p	2.62241E-04	2.41959E-04	4.29015E-04	2.09951E-04

Tabela B.2 Resultados dos erros no *in-sample* (EI) e no *out-of-sample* (EO) no método da regressão linear dos ativos NVG e SON (fase 1).

Variáveis preditoras	ALTR		BCP		NOS	
	EI	EO	EI	EO	EI	EO
Fase1 e *volume*	1.68734E-03	1.03193E-03	6.76415E-04	9.22139E-04	3.88331E-04	1.55255E-04
Fase1 e *hl*	1.68721E-03	1.03487E-03	6.76496E-04	9.20420E-04	3.87928E-04	1.53694E-04
Fase1 e *co*	1.68734E-03	1.03257E-03	6.76291E-04	9.20043E-04	3.87973E-04	1.53730E-04
Fase1, *volume* e *hl*	1.68720E-03	1.03465E-03	6.76379E-04	9.22123E-04	3.87254E-04	1.55359E-04
Fase1, *volume* e *co*	1.68733E-03	1.03239E-03	6.76078E-04	9.22899E-04	3.88322E-04	1.55418E-04
Fase1, *hl* e *co*	1.68721E-03	1.03483E-03	6.76154E-04	9.20320E-04	3.87920E-04	1.53683E-04
Fase1, *volume*, *hl* e *co*	1.68720E-03	1.03461E-03	6.76049E-04	9.22515E-04	3.88298E-04	1.55264E-04

Tabela B.3 Resultados dos erros no *in-sample* (EI) e no *out-of-sample* (EO) da fase 1 com a adição de novas variáveis preditoras nos ativos ALTR, BCP e NOS (fase 2).

Variáveis preditoras	NVG		SON	
	EI	EO	EI	EO
Fase1 e *volume*	2.62456E-04	2.41888E-04	4.29472E-04	2.10771E-04
Fase1 e *hl*	2.62065E-04	2.42402E-04	4.29199E-04	2.10598E-04
Fase1 e *co*	2.62281E-04	2.42055E-04	4.29280E-04	2.10006E-04
Fase1, *volume* e *hl*	2.62047E-04	2.42996E-04	4.29464E-04	2.10760E-04
Fase1, *volume* e *co*	2.62280E-04	2.42161E-04	4.29469E-04	2.10745E-04
Fase1, *hl* e *co*	2.62017E-04	2.43414E-04	4.29059E-04	2.10840E-04
Fase1, *volume*, *hl* e *co*	2.61954E-04	2.43372E-04	4.29057E-04	2.10620E-04

Tabela B.4 Resultados dos erros no *in-sample* (EI) e no *out-of-sample* (EO) da fase 1 com a adição de novas variáveis preditoras nos ativos NVG e SON (fase 2).