Enhancing a Pairs Trading strategy with the application of Machine Learning

Simão Nava de Moraes Sarmento

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Nuno Cavaco Gomes Horta

Examination Committee

Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Nuno Cavaco Gomes Horta
Member of the Committee: Prof. Helena Isabel Aidos Lopes Tomás

September 2019

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would first like to thank my thesis supervisor, Prof. Nuno Horta, for his guidance during this work. His feedback was fundamental in delineating the aspects this thesis should focus on.

I would also like to thank my friends and my family, for their constant support during the course of this work, even when that implied spending less time with them.

Resumo

Pairs Trading é uma das estratégias mais importantes utilizadas por fundos de investimento. É particularmente interessante por ultrapassar o processo árduo de avaliação de títulos financeiros focando-se no preço relativo. Através da compra de um título financeiro relativamente subvalorizado e da venda de um sobrevalorizado, o investidor pode lucrar após a convergência dos preços do par. No entanto, com a afluência dos dados disponíveis, tornou-se mais difícil encontrar pares recompensadores. Neste trabalho investigamos dois problemas: (i) como encontrar pares rentáveis num espaço de procura limitado e (ii) como evitar longos períodos de queda devido a pares com divergência prolongada. Para gerir estas dificuldades, exploramos a aplicação de técnicas promissoras de Aprendizagem Automática. Propomos a integração de um algoritmo de aprendizagem não supervisionada, OPTICS, para lidar com o problema (i). Os resultados demonstram que a técnica sugerida é capaz de superar as técnicas comuns, sendo que o portfólio alcança um *Sharpe ratio* médio de 3.80, em comparação com 3.58 e 2.59 obtido com técnicas padrão. Para o problema (ii), introduzimos um modelo de investimento baseado em previsão, capaz de reduzir os períodos de queda em 75%, ainda que isso implique uma diminuição da rentabilidade. A estratégia proposta é testada utilizando um modelo ARMA, uma LSTM e um Codificador-Descodificador baseado em LSTM. Os resultados deste trabalho são simulados durante diversos períodos, entre Janeiro de 2009 e Dezembro de 2018, utilizando séries de preços de 5 minutos de um grupo de 208 ETFs relacionadas com *commodities*, e considerando custos transacionais.

Palavras-chave: *Pairs Trading*, Mercado Neutro, Aprendizagem Automática, Aprendizagem Profunda, Aprendizagem Não Supervisionada

Abstract

Pairs Trading is one of the most valuable market-neutral strategies used by hedge funds. It is particularly interesting as it overcomes the arduous process of valuing securities by focusing on relative pricing. By buying a relatively undervalued security and selling a relatively overvalued one, a profit can be made upon the pair's price convergence. However, with the growing availability of data, it became increasingly harder to find rewarding pairs. In this work, we address two problems: (i) how to find profitable pairs while constraining the search space and (ii) how to avoid long decline periods due to prolonged divergent pairs. To manage these difficulties, the application of promising Machine Learning techniques is investigated in detail. We propose the integration of an Unsupervised Learning algorithm, OPTICS, to handle problem (i). The results obtained demonstrate the suggested technique can outperform the common pairs' search methods, achieving an average portfolio Sharpe ratio of 3.79, in comparison to 3.58 and 2.59 obtained by standard approaches. For problem (ii), we introduce a forecasting-based trading model, capable of reducing the periods of portfolio decline by 75%. Yet, this comes at the expense of decreasing overall profitability. The proposed strategy is tested using an ARMA model, an LSTM and an LSTM Encoder-Decoder. This work's results are simulated during varying periods between January 2009 and December 2018, using 5-minutes price data from a group of 208 commodity-linked ETFs, and accounting for transaction costs.

Keywords: Pairs Trading, Market Neutral, Machine Learning, Deep Learning, Unsupervised Learning

Contents

Declaration . . . iii

Acknowledgments . . . v

Resumo . . . vii

Abstract . . . ix

List of Tables . . . xvii

List of Figures . . . xix

List of Acronyms . . . xxi

1 Introduction **1**

1.1 Topic Overview . . . 1

1.2 Objectives . . . 3

1.3 Thesis Outline . . . 4

2 Pairs Trading - Background and Related Work **5**

2.1 Mean-Reversion and Stationarity . . . 5

2.1.1 Augmented Dickey-Fuller test . . . 6

2.1.2 Hurst exponent . . . 6

2.1.3 Half-life of mean-reversion . . . 7

2.1.4 Cointegration . . . 7

2.2 Pairs selection . . . 8

2.2.1 The minimum distance approach . . . 9

2.2.2 The correlation approach . . . 10

2.2.3 The cointegration approach . . . 10

2.2.4 Other approaches . . . 11

2.3 Trading execution 11
2.3.1 Threshold-based trading model 11
2.3.2 Other trading models in the literature 13
2.4 The application of Machine Learning in Pairs Trading 13
2.5 Conclusion 14

3 Proposed Pairs Selection Framework 15
3.1 Problem statement 15
3.2 Proposed framework 16
3.3 Dimensionality reduction 17
3.4 Unsupervised Learning 18
3.4.1 Problem requisites 19
3.4.2 Clustering methodologies 19
3.4.3 DBSCAN 20
3.4.4 OPTICS 22
3.5 Pairs selection criteria 25
3.6 Framework diagram 27
3.7 Conclusion 27

4 Proposed Trading Model 29
4.1 Problem statement 29
4.2 Proposed model 30
4.3 Model diagram 32
4.4 Time series forecasting 33
4.5 Autoregressive Moving Average 34
4.6 Arti cial Neural Network models 35
4.6.1 Long Short-Term Memory 35
4.6.2 LSTM Encoder-Decoder 36
4.7 Arti cial Neural Networks design 37
4.7.1 Hyperparameter optimization 37
4.7.2 Weight initialization 38

4.7.3 Regularization techniques . . . 38
4.8 Conclusion . . . 40

5 Implementation 41
5.1 Research design . . . 41
5.2 Dataset . . . 42
5.2.1 Exchange-Traded Funds . . . 42
5.2.2 Data description . . . 43
5.2.3 Data preparation . . . 44
5.2.4 Data partition . . . 45
5.3 Research Stage 1 . . . 47
5.3.1 Development of the pairs selection techniques . . . 48
5.3.2 Trading setup . . . 48
5.3.3 Test portfolios . . . 48
5.4 Research Stage 2 . . . 49
5.4.1 Building the forecasting algorithms . . . 49
5.4.2 Test conditions . . . 50
5.5 Trading simulation . . . 50
5.5.1 Portfolio construction . . . 50
5.5.2 Transaction costs . . . 52
5.5.3 Entry and exit points . . . 53
5.6 Evaluation metrics . . . 53
5.6.1 Return on Investment . . . 53
5.6.2 Sharpe Ratio . . . 54
5.6.3 Maximum Drawdown . . . 56
5.7 Implementation environment . . . 56
5.8 Conclusion . . . 56

6 Results 57
6.1 Data cleaning . . . 57
6.2 Pairs selection performance . . . 58

6.2.1 Eligible pairs 58
6.2.2 Pairs selection rules 59
6.2.3 OPTICS application results 60
6.2.4 Trading performance 64
6.3 Forecasting-based trading model performance 67
6.3.1 Eligible pairs 67
6.3.2 Forecasting algorithms training 69
6.3.3 Forecasting performance 70
6.3.4 Trading performance 72
6.3.5 Implementation remarks 73
6.3.6 Alternative methods 74

7 Conclusions and Future Work 77
7.1 Conclusions 77
7.2 Future work 78

Bibliography 79

Appendix A List of ETFs A.83

Appendix B Sharpe Ratio Scale Factors B.85

Appendix C t-SNE Visualization C.87

Appendix D Forecasting Performance Measures D.89
D.1 Mean Squared Error D.89
D.2 Mean Absolute Error D.89
D.3 Root Mean Squared Error D.90

Appendix E Forecasting Results E.91

Appendix F Code F.93
F.1 Code organization F.93
F.2 Running the code F.94

F.3 Deep Learning models graphs F.95

List of Tables

2.1 Literature exploring the application of Machine Learning in Pairs Trading. 14

3.1 Key points from the proposed pairs selection framework. 28

4.1 Key points from the proposed trading model. 40

5.1 Commodity categories considered in the dataset. 44

5.2 Dataset partitions. 47

5.3 Threshold-based model parameters. 48

5.4 Transaction costs considered. 52

5.5 Risk-free rates considered per test period. 55

6.1 Data preprocessing results. 58

6.2 Pairs selected for different clustering approaches. 58

6.3 Pairs selection progress on an unrestricted pairs search environment. 59

6.4 Clustering statistics when varying number of principal components. 61

6.5 Clustering statistics using large number of principal components. 61

6.6 Validation results for each pairs grouping technique. 65

6.7 Test results for each pairs grouping technique. 66

6.8 Pairs selected during research stage 2. 67

6.9 Candidate pairs identi ed from Jan 2009 to Dec 2017. 68

6.10 GCC Sector Breakdown. 68

6.11 Forecasting results comparison. 71

6.12 Trading results comparison using a 9-year-long formation period. 72

6.13 Trading results for standard trading model using a 3-year-long formation period. 73

6.14 Forecasting algorithms training duration. 74

A.1 List of ETFs (1). A.83

A.2 List of ETFs (2). A.84

B.1 Scale factors for time-aggregated Sharpe ratios when returns follow an AR(1) process. . . B.85

E.1 ARMA forecasting results. E.91

E.2 LSTM forecasting results. E.92

E.3 LSTM Encoder Decoder forecasting results (1). E.92

E.4 LSTM Encoder Decoder forecasting results (2). E.92

List of Figures

1.1 Price series which could potentially form pro table pairs. 1

1.2 Price series of two constituents of a pair during 2009-2018. 2

1.3 Exemplifying a Pairs Trading strategy execution. 3

1.4 Research questions to be answered. 4

2.1 Pairs search common techniques. 9

2.2 Cointegration approach for selecting pairs. 10

2.3 Applying the threshold-based model. 12

3.1 Example of different cluster con gurations. 20

3.2 DBSCAN illustration of basic concepts. 21

3.3 Clusters with varying density. 22

3.4 Illustration of core-distance and reachability-distance in OPTICS. 24

3.5 Extracting clusters using OPTICS. 24

3.6 Pairs selection rules. 26

3.7 Pairs selection diagram. 27

4.1 Illustration of an untimely entry position. 29

4.2 Local momentum forecasting-based strategy. 30

4.3 Spread percentage change distributions. 32

4.4 Proposed forecasting-based strategy. 32

4.5 Proposed model application diagram. 33

4.6 Recurrent Neural Network structure. 35

4.7 Long Short-Term Memory cell. 36

4.8 LSTM Encoder-Decoder. 37

4.9 Early Stopping illustration. 39

4.10 Application of dropout. 40

5.1 Research design overview. 41

5.2 Exempli cation of Univariate, Quasi-Multivariate and Multivariate strategies. 42

5.3 Data preprocessing steps 44

5.4 Period decomposition. 45

5.5 Data partition periods. 46

5.6 Test portfolios. 49

5.7 Trading models interaction diagram. 50

5.8 Market position de nition. 51

5.9 Calculation of transaction costs 52

6.1 Application of t-SNE to the clusters generated by OPTICS during Jan 2014 to Dec 2016. . 62

6.2 Price series composition of some clusters formed in the period from Jan 2014 to Dec 2016. 63

6.3 Applying DBSCAN with varying " to the formation period from Jan 2014 to Dec 2016. . . 64

6.4 Spread from the ve pairs identi ed between Jan 2009 and Dec 2018. 69

6.5 Encoder-Decoder training loss history for DGP-UGL spread. 70

6.6 Return series forecast. 74

C.1 OPTICS clustering in the Jan 2012- Dec 2014 formation period. C.87

C.2 OPTICS clustering in the Jan 2013- Dec 2015 formation period. C.88

F.1 Graphs of the neural network models with the best forecasting-score, produced with Keras. F.95

List of Acronyms

ADF	Augmented Dickey-Fuller
ANN	Arti cial Neural Network
ARMA	Autoregressive–moving-average
DBSCAN	Density-based spatial clustering of applications with noise
ETF	Exchange-Traded Fund
FNN	Feedforward Neural Network
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MDD	Maximum Drawdowm
MLP	Multilayer Perceptron
MSE	Mean Squared Error
OPTICS	Ordering points to identify the clustering structure
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
ROI	Return on Investment
SR	Sharpe Ratio
SSD	Sum of Euclidean Squared Distance
t-SNE	T-distributed Stochastic Neighbor Embedding

Chapter 1

Introduction

1.1 Topic Overview

Pairs Trading is a well-known investment strategy developed in the 1980s. It has been employed as one important long/short[1] equity investment tool by hedge funds and institutional investors [1], and is a fundamental topic in this work. This strategy comprises two steps. First, it requires the identi cation of two securities, for example two stocks, for which the corresponding prices series display a similar behaviour, or simply seem to be linked to each other. Ultimately, this indicates that both securities are exposed to related risk factors and tend to react in an identical way. Figure 1.1 illustrates how this behaviour can be found in some popular stocks. In Figure 1.1(a), we may observe how the price series from two car manufacturers seem to be tied to each other. The same behaviour is also illustrated in Figure 1.1(b), this time illustrating the price series of two of the biggest retail stores in the United States. Two securities that verify an equilibrium relation between their price series can compose a pair[2].

(a) Price series from two automobile manufacturers, Groupe Renault (RNL) and Peugeot S.A (UG).

(b) Price series from two retail corporations, Walmart Inc. (WML) and Target Corporation (TGT).

Figure 1.1: Price series which could potentially form pro table pairs.

Once the pairs have been identi ed, the investor may proceed with the strategy's second step. The

[1] In the nancial markets nomenclature, a long position entails buying a security whereas a short position entails selling a security.

[2] Each constituent of a pair is sometimes referred to as a pair's leg.

underlying premise is that if two securities' price series have been moving close in the past, then this should persist in the future. Therefore, if an irregularity occurs, it should provide an interesting trade opportunity to pro t from its correction. To nd such opportunities, the spread [3] between the two constituents of the pairs must be continuously monitored. When a statistical anomaly is detected, a market position is entered. The position is exited upon an eventual spread correction. It is interesting to observe that this strategy relies on the relative value of two securities, regardless of their absolute value.

We proceed to introduce how the strategy may be applied using an example from this work. A more formal description concerning the trading setup is presented in section 2.3. For now, we assume that two securities (identi ed by the tickers PXI and PXE) have been previously identi ed as forming a potential pair. PXE and PXI are two different securities that track indices related to energy exploration in the United States, and thus it is not surprising that their prices tend to move together. To con rm that this is the case, the two price series, from 2009 to 2018, can be observed in Figure 1.2. The investor may calculate the mean value of the spread formed by the two constituents of the pair, as well as its standard deviation. These values describe the statistical behaviour known for that pair and which the investor expects to remain approximately constant in the future.

Figure 1.2: Price series of two constituents of a pair during 2009-2018.

In the subsequent period, the spread, de ned as $S_t = PXI_t \quad PXE_t$, is normalized[4] and cautiously monitored, as illustrated in Figure 1.3. Although it evolves around its mean, it displays some noticeable deviations. Depending on their magnitude, they may trigger a trade. For that purpose, the investor de nes the long and short thresholds, which de ne the minimum required deviation to open a long or short position, respectively. A long position presumes the spread will widen since its current value is below expected. Therefore, it entails buying PXI and selling PXE. Contrarily, a short position presumes the spread will narrow, and thus the opposite transaction takes place. The positions are liquidated when the spread reverts to its expected value.

Revisiting Figure 1.3, we can identify the market position by the green line, which takes the values -1, 0 or 1 depending on whether the current position is short, outside the market, or long, respectively. In

[3] For now, the spread is de ned as the difference between two securities' price series.
[4] Normalization, in this case, corresponds simply to subtract the mean and divide by the standard deviation.

Figure 1.3: Exemplifying a Pairs Trading strategy execution.

the second half of February 2018, we witness the opening of a short position, when the spread deviates across the short threshold, and its successive closure when the spread reverts back to zero.

The strategy just described presents diverse advantages. One incentive is overcoming the arduous process of characterizing the securities from a valuation point of view, which is a fundamental step in deciding to sell the overvalued securities and to buy the undervalued ones. By focusing on the idea of relative pricing, this issue is mitigated. Furthermore, this strategy is extremely robust to different market conditions. Regardless of the market direction (going up, down or sideways), if the asset the investor bought is performing relatively better than the one he sold, a pro t can be made.

1.2 Objectives

Research in the eld focuses mainly on traditional methods and statistical tools to improve the critical aspects of this strategy. Although in recent years Machine Learning techniques have gained momentum, reported Machine Learning based research for Pairs Trading in speci c is sparse and lacks empirical analysis [2]. This leaves an opportunity to explore the integration of Machine Learning at different levels of a Pairs Trading framework.

The success of a Pairs Trading strategy highly depends on nding the right pairs. But with the increasing availability of data, more traders manage to spot interesting pairs and quickly pro t from the correction of price discrepancies, leaving no margin for the latecomers. If pro table pairs that are not being monitored by so many traders could be found, more lucrative trades would come along. The problem is that it can be extremely hard to nd such opportunities. On the one hand, if the investor limits its search to securities within the same sector, as commonly executed, he is less likely to nd pairs not yet being traded in large volumes. If on the other hand, the investor does not impose any limitation on the search space, he might have to explore an excessive number of combinations and is more likely to nd spurious relations.

To solve this issue, this work proposes the application of Unsupervised Learning to de ne the search space. It intends to group relevant securities (not necessarily from the same sector) in clusters, and detect rewarding pairs within them, that would otherwise be harder to identify, even for the experienced investor. We thus aim to answer the following: “Can Unsupervised Learning nd more promising pairs?”.

One of the disadvantages typically pointed out to a Pairs Trading strategy is its lack of robustness. Since the spread might still diverge after opening a market position, which counts on its reversion, the investor may see its portfolio value decline while the spread does not converge. In this work, a trading model which makes use of time series forecasting is proposed. It intends to de ne more precise entry points in order to reduce the number of days the portfolio value declines due to diverging positions. For this purpose, the potential of Deep Learning is evaluated. Therefore, the second objective of this work is to respond to the following question: “Can a forecasting-based trading model achieve a more robust performance?”

Figure 1.4: Research questions to be answered.

In the process of answering the research questions presented in Figure 1.4, this work provides two additional contributions to the literature. First, it analyzes the suitability of Exchange-Traded Fund (ETF) linked with commodities in a Pairs Trading setting. A total of 208 ETFs are considered. Secondly, it examines the application of a Pairs Trading strategy using 5-minutes frequency price series. This is particularly relevant given that most studies in the Pairs Trading literature use daily prices, with few exceptions [3–7]. Simulations are run during varying periods, between January 2009 and December 2018.

1.3 Thesis Outline

This thesis is composed by a total of seven chapters. Chapter 2 introduces the background and state-of-the-art concerning Pairs Trading along with some critical mathematical concepts to grasp the workings of this strategy. Chapter 3 and 4 describe the two implementations proposed in this work. The former describes in detail the proposed pairs selection framework (Research Stage 1), and the latter proposes a trading model that makes use of time series forecasting (Research Stage 2). Chapter 5 comprises some practical information concerning the way this investigation is designed. The results obtained are illustrated in chapter 6. Finally, chapter 7 focuses on answering the two research questions that motivated this research work, emphasizing the contributions made.

Chapter 2

Pairs Trading - Background and Related Work

This chapter presents an overview of the fundamental concepts of Pairs Trading, along with a revision of the related work in the field.

2.1 Mean-Reversion and Stationarity

Before delving into more specific details of Pairs Trading, there exist a set of tools that should be introduced for the full comprehension of the related formalities. We start by introducing the concept of stationarity. A stochastic process is said to be stationary if its mean and variance are constant over time [8]. That said, a stationary time series is mean reverting in nature and fluctuations around the mean should have similar amplitudes. A stationary process can also be referred to with respect to its order of integration $I(d)$, in which case a stationary time series is said to be an $I(0)$ process, whereas a non-stationary process is $I(1)$. The order of integration d is a summary statistic which reports the minimum number of differences required to obtain a stationary series.

Stationary series are particularly interesting for financial traders who take advantage of the stationary property by placing orders when the price of a security deviates considerably from its historical mean, considering the price will revert back. However, they are rarely found in the set of financial time series. In fact, price series are often non-stationary, and thus, in theory, completely unpredictable. This pattern should remind the reader of the spread illustrated in Figure 1.3. This is not a coincidence. As we will see, Pairs Trading is particularly interesting because it provides a way of obtaining an artificial stationary time series from the combination of two non-stationary time series.

It should be clarified that mean-reversion and stationarity are not the same mathematical concept. According to [9], the mathematical description of a mean-reverting time series is that the change of the

time series in the next instant is proportional to the difference between the mean and the current value. Whereas, the mathematical description of a stationary time series is that the variance of the logarithmic value of the time series increases slower than that of a geometric random walk[1]. That is, its variance is a sub-linear function of time, rather than a linear function, as in the case of a geometric random walk. Some necessary tools to detect and model the aforementioned properties are now presented in detail.

2.1.1 Augmented Dickey-Fuller test

The Augmented Dickey-Fuller (ADF) test is a hypothesis test designed to assess the null hypothesis that unit root is present in a time series, proposed by Dickey and Fuller [11]. The test assumes stationarity as the alternative hypothesis. A consecutive change in time series can be modelled as

$$y(t) = \quad y\,(t \quad 1) + \quad + \quad t + \quad {}_1 \quad y(t \quad 1) + \quad + \quad {}_k \quad y(t \quad k) + \quad {}_t; \tag{2.1}$$

where $y(t) \quad y(t) \quad y(t \quad 1)$, is a constant, is the coef cient on a time trend and k is the lag order of the autoregressive process. The ADF test evaluates whether $= 0$. If the hypothesis is rejected, and $\neq 0$, the change of a time series at time t depends on the value of the series at time $t \quad 1$, meaning that the series cannot be a simple random walk. The test statistic associated is the regression coef cient (calculated with $y(t \quad 1)$ as the independent variable and $y(t)$ as the dependent variable) divided by the standard error (SE) of the regression t, $\frac{}{SE}$. Note that when mean reversion is expected, $\frac{}{SE}$ has a negative value. This value should be compared with the critical values corresponding to the distribution of the test statistic, which can be found tabulated and used to decide whether the hypothesis can be accepted or rejected at a given probability level.

2.1.2 Hurst exponent

The Hurst exponent can be used to evaluate the stationarity of a time series. Thus, according to the de nition presented in section 2.1, this metric assesses whether the time series' speed of diffusion from its initial value is slower than that of a geometric random walk [12]. Formally, the speed of diffusion can be characterized by the variance as

$$\mathrm{Var}(\quad) = \quad jz(t + \quad) \quad z(t)j^2 \quad ; \tag{2.2}$$

where $z(t)$ is the logarithmic time series value at time t, is an arbitrary time lag and $h\cdot i$ is the average across time. Having de ned an indicator for the speed of diffusion, it is still necessary to de ne the geometric random walk standard speed, by which to compare. We may assume that its speed of diffusion can be approximated as

$$jz(t + \quad) \quad z(t)j^2 \qquad {}^{2H} : \tag{2.3}$$

[1]The geometric random walk is the default stochastic process model used to describe stock market data. Describing this model falls out of the scope of this work, but the interested readers may refer to [10].

The symbol H represents the Hurst exponent. For a price series exhibiting a geometric random walk, H = 0:5, and (2.3) simply becomes

$$jz(t + \quad) \quad z(t)j^2 \qquad : \tag{2.4}$$

Yet, as H decreases towards zero, the speed of diffusion reduces, meaning the price series is more mean-reverting. Contrarily, as H increases toward 1, the price series is increasingly trending. The Hurst exponent can thus be interpreted as an indicator of the degree of mean-reversion or trendiness.

2.1.3 Half-life of mean-reversion

The half-life of mean-reversion is a measure of how long it takes for a time series to mean-revert. To calculate this metric, the discrete-time series from (2.1) can be transformed into its differential form, so that the changes in prices now become in nitesimal quantities. Additionally, the term may be ignored since we are dealing with price series and the constant drift in price, if any, tends to be of a much smaller magnitude than its daily uctuations. If, for further simpli cation, the lagged differences are also ignored in (2.1), the expression can be written as

$$dy(t) = (\; y \; (t \quad 1) + \quad)dt + d'': \tag{2.5}$$

This expression describes an Ornstein-Uhlenbeck process, where d'' represents some Gaussian Noise [9]. Assuming has been calculated as described in section 2.1.1, the analytical solution for the expected value of y(t) is represented by

$$E(y(t)) = \; y_0 \exp(\; t \;) \quad = \; (1 \quad \exp(\; t \;)): \tag{2.6}$$

From the analysis of (2.6), we may con rm that for > 0 the time series will not be mean reverting, as stated in section 2.1.1. For a negative value of , the expected value of the time series decays exponentially to the value − with the half-life of decay equals to $\frac{\log(2)}{}$. This result implies that the expected duration of mean reversion is inversely proportional to the absolute value of , meaning that a larger absolute value of is associated with a faster mean-reversion. Therefore, $\frac{\log(2)}{}$ can be used as a natural time scale to de ne many parameters on a trading strategy and avoiding brute-force optimization of a free parameter.

2.1.4 Cointegration

Cointegration was rst proposed in an article, by two econometricians, Engle and Granger [13]. A set of variables is said to be cointegrated if there exists a linear combination of those variables, of order d, which results in a lower order of integration, I(d 1). In the context of this work, cointegration is veri ed if a set of I(1) variables (non-stationary) can be used to model an I(0) variable (stationary). Formally,

considering two time series, y_t and x_t which are both I(1), cointegration implies there exist coef cients, and such that

$$y_t \quad x_t = u_t + \quad ; \tag{2.7}$$

where u_t is a stationary series. This is particularly interesting as it provides a way to arti cially create a stationary time series that can be used for trading. Finding a time series with the same properties amid raw nancial time series is an extremely hard task.

Tests for cointegration identify stable, long-run relationships between sets of variables. The most common cointegration tests are the Engle-Granger two-step method and the Johansen test. The two-step Engle-Granger cointegration test proceeds as follows:

1. Test if a unit root is present in the series y_t and x_t using ADF test. If af rmative, proceed to step 2.
2. Run regression de ned in (2.7), using Ordinary Least Squares, and save the residuals, $\hat{u}_t$.
3. Test the residuals $\hat{u}_t$ for a unit root, using ADF test (or similar).
4. If the null hypothesis of a unit root in the residuals (null of no-cointegration) is rejected, meaning the residual series is stationary, the two variables are cointegrated.

A major issue with the Engle-Granger method is that the choice of the dependent variable may lead to different conclusions, as pointed by Armstrong [14]. The Johansen test is particularly useful for detecting cointegrating vectors when working with more than two securities, but that falls out of the scope of this work.

Having described the fundamental mathematical concepts required to understand the workings of a Pairs Trading strategy, we proceed to disclose how they can be applied in practice. As described in section 1.1, a Pairs Trading strategy is usually organized in two stages, the identi cation of the pairs and the trading itself. Following this line of reasoning, the two next sections will go over each stage separately.

2.2 Pairs selection

The pairs selection stage encompasses two steps: (i) nding the appropriate candidate pairs and (ii) selecting the most promising ones.

Starting with (i), the investor should select the securities of interest (e.g. stocks, futures, ETFs, etc.) and from there start searching for possible pair combinations. In the literature, two methodologies are typically suggested for this stage: performing an exhaustive search for all possible combinations of the selected securities, or arranging them in groups, usually by sector, and constrain the combinations to pairs formed by securities within the same group. While the former may nd more interesting unusual pairs, the latter reduces the likelihood of nding spurious relations. For example, [15, 16] impose no restriction on the universe from which to select the pairs. Contrarily, [7, 17, 18] arrange the securities on category groups and select pairs within the same group. Figure 2.1 illustrates these two techniques, in

a two-dimensional reduced setting, with the characters indicating hypothetical categories.

(a) Searching for pairs without restrictions. (b) Searching for pairs within the same category.

Figure 2.1: Pairs search common techniques.

With respect to (ii), the investor must de ne what criteria should be used to select a pair combination. The most common approaches to select pairs are the distance, the cointegration and the correlation approaches, which we proceed to describe in more detail.

2.2.1 The minimum distance approach

The work from Gatev et al. [17] can be considered as the baseline for distance-based selection criteria. It proposes selecting pairs which minimize a distance criterion. The authors construct a cumulative total return index for each stock and normalize it to the rst day of a 12-month formation period. This period is used for calculating the Sum of Euclidean Squared Distances (SSD) between the constructed time series. Then, the authors suggest ranking the pairs according to the minimum historic SSD.

However, according to Krauss [2], the choice of Euclidean squared distance as a selection metric is analytically sub optimal. To prove this statement, let $p_{i;t}$ and $p_{j;t}$ be realizations of the normalized price processes $P_i = (P_{i;t})_{t \in T}$ and $P_j = (P_{j;t})_{t \in T}$ of securities i and j composing a pair. Additionally, $s^2_{P_i - P_j}$ denotes the empirical spread variance and it is expressed as

$$s^2_{P_i - P_j} = \frac{1}{T}\sum_{t=1}^{T}(p_{i;t} - p_{j;t})^2 - \left(\frac{1}{T}\sum_{t=1}^{T}(p_{i;t} - p_{j;t})\right)^2 : \tag{2.8}$$

The average sum of squared distances, $\overline{ssd}_{P_i;P_j}$ in the formation period is then given by

$$\overline{ssd}_{P_i;P_j} = \frac{1}{T}\sum_{t=1}^{T}(p_{i;t} - p_{j;t})^2 = s^2_{P_t - P_j} + \left(\frac{1}{T}\sum_{t=1}^{T}(p_{i;t} - p_{j;t})\right)^2 : \tag{2.9}$$

An optimal pair according to the SSD criterion would be one that minimizes (2.9). But this implies that a zero spread pair across the formation period would be considered optimal. Logically, this is not consistent with the idea of a potentially pro table pair, since a good candidate should have high spread variance and strong mean-reversion properties to provide trading opportunities. As this metric does not account for this requirement, it is likely to form pairs with low spread variance and limited pro t potential.

2.2.2 The correlation approach

The application of correlation on return levels to de ne a pairs' selection metric is examined in [19]. The authors intend to construct a robust pair selection metric that nds promising pairs without penalizing individual price divergences, as it happens in [17]. Therefore, they use the same dataset as in [17] for a direct comparison. The sample variance of the spread returns of a pair composed by asset i and j is de ned as return on buy minus return on sell, as

$$s^2_{R_i \quad R_j} = s^2_{R_i} + s^2_{R_j} \quad 2\hat{}_{R_i;R_j} \sqrt{s^2_{R_i}} \sqrt{s^2_{R_j}}; \tag{2.10}$$

where ^ is the correlation found. It can be inferred from (2.10) that, although imposing high return correlation between two stocks' returns leads to lower variance of spread returns, the return time series of the individual stocks may still exhibit high variances.

The results using Pearson correlation show better performance, with a reported monthly average of 1.70% raw returns, almost twice as high as the obtained using the distance metric suggested by Gatev et al. [17]. However, it should be noted that, although correlation performed better than SSD as a selection metric, it is still not optimal, as two return level correlated securities might not share an equilibrium relationship, and divergence reversions cannot be explained theoretically.

2.2.3 The cointegration approach

The cointegration approach entails selecting pairs for which the two constituents are cointegrated. If two securities, Y_t and X_t are found to be cointegrated, then by de nition the series resulting from the linear combination

$$S_t = Y_t \quad X_t; \tag{2.11}$$

where is the cointegration factor, must be stationary. Naturally, de ning the spread series this way is particularly convenient, since under these conditions it is expected to be mean-reverting. According to [2], cointegration approach for pairs selection is a more rigorous framework when compared to the distance approach. This is mainly because cointegration nds econometrically more sound equilibrium relationships. The selection criteria follows the procedure illustrated in Figure 2.2. The most cited

Figure 2.2: Cointegration approach for selecting pairs.

author in this field is Vidyamurthy [20]. In his book, a set of heuristics for cointegration strategies are proposed. Huck and Afawubo [21] performs a comparison study between the cointegration approach and the distance approach using S&P 500 constituents, under varying parameterizations, considering risk loadings and transactions costs. The authors found that the cointegration approach significantly outperforms the distance method, which corroborates the hypothesis that the cointegration approach identifies more robust relationships.

2.2.4 Other approaches

Although the three approaches just described are the most commonly mentioned in the literature, there are naturally other methodologies used for tackling this issue. We find it relevant to mention the strategy introduced by Ramos-Requena et al. [22]. The authors suggest ranking the pairs based on the spread's Hurst exponent. They obtain superior results when compared with the classical approaches based on cointegration and correlation, especially as the number of pairs in the portfolio grows. This indicates this may be an interesting research direction.

The approaches described up to this point are statistically motivated. Nevertheless, this does not have to be the case. Some research work also focuses on specific pairs which tend to display an equilibrium that is theoretically founded. For instance, Dunis et al. [23] focus on the gasoline crack spread, which is defined as the differential between the price of crude oil and petroleum products extracted from it. In this scenario, there is an evident justification for the underlying equilibrium. The same happens with the corn/ethanol crush spread, which is examined in [24]. Here, the link can be explained by the fact that corn is the principal raw ingredient used to produce ethanol.

2.3 Trading execution

Having explored the most relevant techniques for selecting pairs, we proceed to examine the existent trading strategies. Unlike pairs selection, which can be found in diverse configurations, the trading strategy usually follows the threshold-based trading model described next.

2.3.1 Threshold-based trading model

In the classical framework, proposed by Gatev et al. [17], the criteria for opening a trade is based on the spread divergence. If the spread between two price series composing a pair, diverges by more than two historical standard deviations, a trade is opened. The trade is closed upon convergence to the mean, at the end of the trading period or when delisting[2] occurs. This model should be familiar to the reader as it corresponds to the implementation introduced in section 1.1.

[2] Delisting is the removal of a listed security from a stock exchange.

Figure 2.3, is presented by Gatev et al. [17] to exemplify how this strategy proceeds. The illustration uses two stocks, Kennecott and Uniroyal. The two lines on top represent the normalized price series, and the bottom line indicates whether the position is open or closed, on a daily basis. As we may observe, positions are opened when the spread deviates from its expected value and closed on convergence. This example illustrates a favourable scenario, in which the identi ed pair provides ve good trading opportunities. However, it could be the case that the spread does not converge, in which case a loss is incurred in the investor's portfolio.

Figure 2.3: Applying the threshold-based model. Source: [17].

This model can be described more formally as follows:

i Calculate the spread's ($S_t = Y_t \quad X_t$) mean, $_s$ and standard deviation, $_s$ during the pair's formation period.

ii De ne the model thresholds: the threshold that triggers a long position, $_L$, the threshold that triggers a short position, $_S$, and the exit threshold $_{exit}$ that de nes the level at which a position should be exited.

iii Monitor the evolution of the spread, S_t and control if any threshold is crossed.

iv In case $_L$ is crossed, go long the spread by buying Y and selling X. If $_S$ is triggered, short the spread by selling Y and buying X. Exit position when $_{exit}$ is triggered and a position was being held.

It is important to note that the way the spread, S_t, is de ned is tied to the technique used to identify the pairs. For example, when applying the minimum distance approach, the spread composed by the securities Y and X is simply de ned as $S_t = Y_t \quad X_t$. However, when using cointegration, the spread is de ned in a slightly different manner, as $S_t = Y_t \quad X_t$. This subtlety affects the way positions are set. If the spread is de ned as $S_t = Y_t \quad X_t$, then entering a long position entails buying Y and selling X in the same number of shares, or investing the same capital on both, in case the investor wants to assume a dollar-neutral position. However, if the spread is de ned as $S_t = Y_t \quad X_t$, one must decide how to handle the cointegration factor. Different alternatives have been proposed in the literature, and there seems to be no consensus on the most appropriate one. According to Chan [9], the investor should invest in one share of Y and shares of X. Rad et al. [25] suggests the investor should invest $1 in the

long leg and determine the value in the short leg according to the cointegration factor. Other authors, such as Dunis et al. [7], simply neglect the cointegration factor to enforce a money-neutral position and invest the same amount in each leg.

2.3.2 Other trading models in the literature

The strategy just described is not concerned with optimizing the market entry points. There is no guarantee that the point immediately after crossing the threshold is an optimal entry point. In fact, the spread might continue to diverge before converging again.

Some efforts have emerged trying to propose more robust models. Techniques from different elds, such as stochastic control theory, statistical process modelling and Machine Learning have been studied. Some examples in the literature include Elliott et al. [26], who model the spread with a mean-reverting Gaussian Markov chain, observed in Gaussian noise. The trading positions are set according to the relative difference between the series prediction and the observation. Dunis et al. [23] proposes a similar framework but this time based on Arti cial Neural Networks. The results obtained by Machine Learning approaches have proved very promising, and are explored in the next section in more detail.

2.4 The application of Machine Learning in Pairs Trading

Over the last years, more research has become available on the application of Machine Learning in the eld of nance [1]. With the advent of more computational power, it became easier to train complex Deep Neural Networks capable of generating very promising results. Nevertheless, the application of Machine Learning in Pairs Trading in speci c is still scarce.

Dunis et al. [23, 24, 27] are the main authors applying Machine Learning to Pairs Trading, even though their work is limited to speci c spreads. Dunis et al. [23] apply Arti cial Neural Networks to model the gasoline crack spread. The authors train the neural networks to predict the percentage change in the spread, from one closing price to the next. The trading rules are then set by comparing the change forecast with xed thresholds, according to,

$$\begin{cases} \text{if} \quad \hat{S}_t > X; & \text{open or stay long the spread} \\ \text{if} \quad \hat{S}_t < \quad X; & \text{open or stay short the spread} \\ \text{if} \quad X < \quad \hat{S}_t < X; & \text{remain out of the spread} \end{cases} \tag{2.12}$$

where $\hat{S}_t$ is the model's predicted spread return and X is the value of the threshold lter (which the authors optimize in-sample). A Recurrent Neural Network (RNN) achieved pro ts of 15% on average before transaction costs. However, the impact of transaction costs is signi cant. Dunis et al. [24] apply Arti cial Neural Networks in an adjusted framework to the corn/ethanol crush spread. In this case, the

authors found more satisfying results with a pro t after transaction costs of approximately 20%.

Huck [28] also explores the integration of Machine Learning in Pairs Trading, by introducing a combined approach, built on Arti cial Neural Networks. The methodology uses RNNs to generate a one-week ahead forecast, from which the predicted returns are calculated. A ranking algorithm is then applied to extract the undervalued and overvalued stocks. In the trading step, the top stocks of the ranking are bought and the bottom stocks sold. The results obtained are, once again, very satisfactory.

Table 2.1 summarizes what we consider to be the most relevant work applying Machine Learning in Pairs Trading. In addition to the studies just described, we include some additional work in the eld [29, 30].

Table 2.1: Literature exploring the application of Machine Learning in Pairs Trading.

Apart from the studies presented, Machine Learning techniques applied in Pairs Trading remain fairly unexplored. Furthermore, the existing limitations suggest there might be open research opportunities in this direction. Lastly, all the applications focus on the supervising power of Machine Learning and seem to neglect how Unsupervised Learning could also be advantageous in this eld.

2.5 Conclusion

In this section, we introduced the main research topics related to Pairs Trading. In section 2.2, we started by reviewing the standard procedures to search for pairs, which include searching for pairs within the same sector or searching for all possible combinations. Next, we analyzed the selection methods, with an emphasis on the correlation, distance and cointegration approaches. In section 2.3, we examined the threshold-based trading model and brie y motivated some more innovative trading techniques. Finally, we focused on the literature available, exploring the application of Machine Learning in Pairs Trading.

Chapter 3

Proposed Pairs Selection Framework

In the previous chapter, we introduced the Pairs Trading investment scheme and its fundamental notions. In this chapter, we propose a new pairs selection framework as part of the rst research stage, aiming to make the pairs search methodology more effective.

3.1 Problem statement

As the popularity of Pairs Trading grows, it is increasingly harder to nd rewarding pairs. The scarcity of such promising pairs forces a search expansion to broader groups of securities, on the expectation that by considering a larger group, the likelihood of nding a good pair will increase. The simplest procedure commonly applied is to generate all possible candidate pairs by considering the combination from every security to every other security in the dataset, as it was represented in Figure 2.1(a). This results in a total of $\frac{n \quad (n \quad 1)}{2}$ possible combinations, where n represents the number of available securities.

Two different problems arise immediately. First, the computational cost of testing mean-reversion for all the possible combinations increases drastically as more securities are considered. In a setting where securities are constantly being monitored to form new pairs, this is especially relevant. The second emerging problem is frequent when performing multiple hypothesis tests at once, and is referred to as the multiple comparisons problem. By de nition, if 100 hypothesis tests are performed (with a con dence level of 5%) the results should contain a false positive rate of 5%. To illustrate how this might be relevant, one might consider the hypothetical situation where 20 shares are being handled. There would be $\frac{20 \quad 19}{2}$ = 190 possible combinations, which means approximately 9 results would be wrong (5% of 190). Since nding real cointegrated pairs is uncommon among a randomly selected group of securities, this number is considerably high. Formally, the probability ($\bar{\ }$) of committing at least one type I error[1] when

[1] In statistical hypothesis testing a type I error is the rejection of a true null hypothesis (false positive), while a type II error is the non-rejection of a false null hypothesis (false negative).

performing m independent comparisons is given by

$$\bar{\ } = 1 \quad (1 \qquad)^{m} ; \tag{3.1}$$

where represents the probability level of a single test. This is known as the family-wise error rate (FMER).

The multiple comparison problem cannot be eliminated entirely, but its impact can be mitigated. Two options consist of applying multiple correction tests like Bonferroni or running fewer statistical tests. Bonferroni is one of the most commonly used approaches for multiple comparisons. However, the Bonferroni correction is substantially conservative. In fact, Harlacher [31] found that Bonferroni correction turns out to be too conservative for pairs selection and impedes the discovery of even truly cointegrated combinations. The author recommends the effective pre-partitioning of the considered asset universe with the purpose of reducing the number of feasible combinations and, therefore, the number of statistical tests.

This aspect might lead the investor to pursue the usual, more restrictive approach of comparing securities only within the same sector, as previously illustrated in Figure 2.1(b). This dramatically reduces the number of necessary statistical tests, consequently reducing the likelihood of nding spurious relations. Moreover, there are fundamental reasons to believe that securities within the same sector are more likely to move together as they are exposed to similar underlying factors. This method also has the advantage of being particularly easy to execute. However, the simplicity of this process might also turn out to be a disadvantage. The more traders are aware of the pairs, the harder it is to nd pairs not yet being traded in large volumes, leaving a smaller margin for pro t.

This disequilibrium motivates the search for a methodology that lays in between these two scenes: an effective pre-partitioning of the universe of assets that does not limit the combination of pairs to relatively obvious solutions, while not enforcing excessive search combinations.

3.2 Proposed framework

In this work, we propose the application of an Unsupervised Learning algorithm, on the expectation that it infers meaningful clusters of assets from which to select the pairs. The motivation is to let the data explicitly manifest itself, rather than manually de ning the groups each security should belong to. The proposed methodology encompasses the following steps:

1. Dimensionality reduction - nd a compact representation for each security;
2. Unsupervised Learning - apply an appropriate clustering algorithm;
3. Select pairs - de ne a set of rules to select pairs for trading.

The three following sections will explore in detail how each of the previous steps is conducted.

3.3 Dimensionality reduction

In the quest for pro table pairs, we wish to nd securities with the same systematic risk-exposure. According to the Arbitrage Pricing Theory[2], these securities generate the same long-run expected return. Any deviations from the theoretical expected return can thus be seen as mispricing and serve as a guideline to place trades. To extract the common underlying risk factors for each security, the application of Principal Component Analysis (PCA) on the return series is proposed, as described in [33].

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables, the principal components. The transformation is de ned such that the rst principal component accounts for as much of the variability in the data as possible. Each succeeding component, in turn, has the highest variance possible under the constraint that it must be orthogonal to the preceding components. It is especially interesting to note that each component can be seen as representing a risk factor [34].

The PCA application goes as follows. First, the return series for a security i at time t, $R_{i;t}$, is obtained from the security price series P_i,

$$R_{i;t} = \frac{P_{i;t} \quad P_{i;t \quad 1}}{P_{i;t \quad 1}}: \tag{3.2}$$

Next, the return series must be normalized since PCA is sensitive to the relative scaling of the original variables. This is performed by subtracting the mean, $\overline{R}_i$, and dividing by the standard deviation $_i$, as

$$Y_i = \frac{R_i \quad \overline{R}_i}{_i}: \tag{3.3}$$

From the normalized return series of all assets, the correlation matrix is calculated, where each entry is determined by

$$_{ij} = \frac{1}{T \quad 1} \sum_{t=T}^{M} Y_{i;t} Y_{j;t} : \tag{3.4}$$

The role of matrix will soon be made more clear. Note that the motivation for applying PCA on return series lies in the fact that a return correlation matrix is more informative for evaluating price co-movements. Using the price series instead could result in the detection of spurious correlations as a result of underlying time trends.

The next step consists of extracting the eigenvectors and eigenvalues, to construct the principal components. The eigenvectors determine the directions of maximum variance, whereas the eigenvalues quantify the variance along the corresponding direction. Eigenvalues and eigenvectors can be determined using singular value decomposition (SVD). For that purpose, we stack the normalized return series for all n securities in a matrix A. By direct application of the SVD theorem on matrix A, it is decomposed as

$$A = USV^T: \tag{3.5}$$

[2] Arbitrage pricing theory (APT) is a general theory of asset pricing that holds that the expected returns of a nancial asset can be modelled as a linear function of various factors or theoretical market indices. For more information, we redirect the reader to [32].

Matrix U is orthogonal, and its columns are the left singular vectors. Matrix S is diagonal and contains the singular values arranged in descending order along the diagonal, from the eigenvalue corresponding to the highest variance to the least. Matrix V is the transposed orthogonal matrix, and its rows are the right singular vectors.

Note that by multiplying $A^T A$ we obtain $V S^2 V^T$. But $A^T A$ is exactly the correlation matrix previously calculated, hence, V can be determined. In this way, it is possible to nd the eigenvectors of the data matrix A. At this point, we select the k eigenvectors corresponding to the k directions of maximum variance, where k represents the number of features to describe the transformed data. The more eigenvectors are considered, the better the data is described. The matrix containing the selected eigenvalues in order of signi cance is called the feature vector.

Finally, the new dataset is obtained by multiplying the original matrix A by the feature vector, resulting in a matrix with size n k. Please note that the position (i; j) of the nal matrix contains the result from the product between the normalized return series, i by the eigenvector j. The matrix obtained is now reduced to the selected k features.

To conclude this section, the number of features, k, needs to be de ned. A usual procedure consists in analyzing the proportion of the total variance explained by each principal component, and then use the number of components that explain a xed percentage, as in [34]. In this work, however, a different approach is adopted. Because an Unsupervised Learning algorithm is applied using these data features, there should be a consideration for the data dimensionality. High data dimensionality presents a dual problem. The rst being that in the presence of more attributes, the likelihood of nding irrelevant features increases. The second is the problem of the curse of dimensionality. This term is introduced by Bellman [35] to describe the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space. This has a tremendous impact when measuring the distance between apparently similar data points that suddenly become all very distant from each other. Consequently, the clustering procedure becomes very ineffective. According to Berkhin [36], the effect starts to be severe for dimensions greater than 15. Taking this into consideration, the number of PCA dimensions is upper bounded at this value and is chosen empirically. The implications of varying this parameter can be seen in section 6.2.3.

3.4 Unsupervised Learning

Having transformed each time series in a smaller set of features with the application of PCA, an Unsupervised Learning technique can be effectively applied. We proceed to describe the decision process followed in selecting the most appropriate algorithm.

3.4.1 Problem requisites

There exist several Unsupervised Learning clustering techniques capable of solving the task at hand. Thus, some problem-speci c requisites are rst de ned to constrain the set of possibilities:

No need to specify the number of clusters in advance;

No need to group all securities;

Strict assignment that accounts for outliers;

No assumptions regarding the clusters' shape.

In more detail, rst, no assumptions should be taken concerning the appropriate number of clusters in the data given that there is no prior information in this regard. By making the number of clusters data-driven, we introduce as little bias as possible. Secondly, it should not be imposed that every security is assigned to a group because we should expect to nd securities with very distinct price series and ultimately divergent PCA decompositions. From a clustering point of view, these securities will be identi ed as outliers. Therefore, we should ensure they do not interfere with the clustering procedure by selecting a clustering algorithm robust in dealing with outliers. In third place, the assignment should be strict otherwise the number of possible pair combinations increases, which is con icting with the initial goal. Finally, because there is no prior information indicating the clusters should be regularly shaped, the selected algorithm should not adopt this assumption.

3.4.2 Clustering methodologies

Having described the problem requisites, we may proceed to select an appropriate clustering type. A common scheme to refer to the different clustering algorithms consists of categorizing them in three broad groups: partitioning clustering, hierarchical clustering, and density-based clustering. The goal of this work is not to perform an exhaustive exploration of all the existent options but rather to nd a sound approach consistent with the properties stated so far. Hence, we are not concerned with explaining all the different methodologies in detail but rather to provide some intuition on the decisions taken.

Partitioning algorithms, such as k-means [37], are not considered suitable mainly for three reasons. The rst is that these algorithms do not handle well noisy data and outliers. Secondly, clusters are forced to have convex shapes. The assumption that the data follows a normal distribution around the centre of the cluster might be too strong of an approximation at this stage. Especially because the data-dimensionality precludes the visualization of its distribution in space. Lastly, the obligation of specifying the number of clusters in advance is also undesirable in a scenario under which the model should have as few parameters as possible. One could argue that by performing a grid search on the number of clusters, an optimal partition should correspond to the con guration that minimizes a given criterion. This way, specifying a range of clusters would be suf cient. However, it is not evident what criteria leads to the selection of the most promising pairs. Clustering integrity metrics, such as the silhouette coef cient [38], sound appropriate. However, we empirically veri ed that greater cluster integrity is not

necessarily strictly correlated with more promising pairs.

Hierarchical clustering algorithms have the advantage of providing a dynamic termination criteria. This lets the investor select a group displacement with the desired level of granularity. Yet, this might add unnecessary bias from the investor. As consequence, the selection can approximate the results obtained using standard search methods, which we aim to prevent. For this reason, hierarchical clustering would only be appropriate under the existence of an automatic termination criterion.

Finally, density-based clustering algorithms display some bene ts in the context of this work. First, clusters can have arbitrary shapes, and thus no gaussianity assumptions need to be adopted regarding the shape of the data. Secondly, it is naturally robust to outliers because it does not group every point in the dataset. Lastly, it requires no speci cation of the number of clusters. Therefore we proceed to investigate how density-based clustering could be applied in this work.

3.4.3 DBSCAN

A density-based clustering algorithm interprets the clustering space as an open set in the Euclidean space that can be divided into a set of its connected components. The algorithm DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [39], is the most in uential in this category. In [39], the author illustrates, with Figure 3.1, that the reason why we can look at each of the databases and easily detect clusters of points is that there is a typical density of points within each cluster which is considerably higher than outside of the cluster.

Figure 3.1: Example of different cluster con gurations. Source: [39].

To formalize this notion of clusters' dependence on density, the key idea relies on the concepts of density and connectivity. There are essentially two model parameters. They are ε and minP ts . To understand the algorithm, we start by de ning the underlying fundamental concepts, w.r.t ε and minP ts .

De nition 3.4.1. ε-neighborhood: The ε-neighborhood of a point q is de ned as

$$N_{\varepsilon}(q) = \{ p \in X \mid d(q;p) \leq \varepsilon \}; \tag{3.6}$$

where $d(q;p)$ represents the distance between q and p, and X is the set of all points.

De nition 3.4.2. Core point: A point q is core point if it veri es

$$jN_{"}(q)j \quad minPts;$$

where $jN_{"}(q)j$ represents the number of points within the "-neighborhood of q. Note also that minPts accounts for the point itself.

De nition 3.4.3. Directly density-reachable: Point p is directly density-reachable from point q in a set of points X provided that $p \; 2 \; N_{"}(q)$ and q is a core point.

De nition 3.4.4. Density-reachable: A point p is density-reachable from a point q if there is a chain of objects $p_1; :::; p_n$, where $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

De nition 3.4.5. Density-connected: A point p is density-connected to a point q if both points are density-reachable from a common core point.

Figure 3.2 illustrates the concepts just de ned, when the parameter minPts takes the value of 5. Figure 3.2(a) describes a core point. Point q is a core point, given that is contains minPts within the circle of radius ", its "-neighborhood. Point p_1 belongs to the "-neighborhood of q. Like p_1, a point that is not a core point but is included in an "-neighborhood is called a border point. A point that does not belong to any neighborhood, such as p_2, is considered an outlier.

Figure 3.2(b) illustrates the concept of directly density-reachable and density reachable points. Point p_1 is directly density-reachable from q. Furthermore, p_2 is density-reachable from q, as both q and p_1 are core points. However, q is not density reachable from p_2, because p_2 is not a core point. As just demonstrated, density-reachability is not symmetric in general. Only core objects can be mutually density-reachable.

Finally, Figure 3.2(c) illustrates the concept of density connection. Points p_2 and p_4 are density-connected, since both points are density reachable from q. Note that density-connectivity is a symmetric relation.

(a) Core points and outliers. (b) Density-reachable points. (c) Density-connected points.

Figure 3.2: DBSCAN illustration of basic concepts, with minPts = 5.

We may now characterize a cluster. If p is a core point, then it forms a cluster together with all points (core or non-core) that can be reached from itself. This implies that each cluster contains at least one core point. Non-core points can also be part of a cluster, but they form its edge since they cannot be used to reach more points. Thus, a cluster satis es two properties: (i) All points within a cluster are

mutually density-connected and (ii) if a point is density-reachable from any point of the cluster, it must belong to the cluster. If a point does not belong to any cluster, it is considered noise. The DBSCAN execution can be described in a simpli ed manner as follows:

1. Find the points in the "-neighborhood of every point and identify the core points with more than minP ts neighbours.
2. Find the connected components of core points on the neighbour graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an "-neighbor, otherwise assign it to noise.

In spite of the DBSCAN advantages stated so far, there exist some caveats that should be noted. First, the algorithm is very sensitive to the parameters, namely the de nition of ". Moreover, even after nding a suitable ", the algorithm assumes clusters are evenly dense. This results in an inability to nd clusters of varying density. Although the parameter " may be well adapted for one given cluster density, it might be unrealistic for another cluster with a different density, within the same dataset. This situation is depicted in Figure 3.3. It is evident that cluster A and B could eventually be found using the same ". But in doing so, we would only identify cluster C and consequently neglect clusters C_1, C_2 and C_3. On the other hand, if we adapt " to nd C_1, C_2 and C_3, clusters A and B will be indistinguishable from noise.

Figure 3.3: Clusters with varying density. Source: [40].

The last drawback pointed out is particularly relevant in this work. Because there is a natural unbalance in the categories of the securities considered (described in Table 5.2), and assuming the category plays a relevant role in describing a security, then clusters are expected to have different densities. The expectation of unbalanced density clusters is a catalyst for moving on with OPTICS.

3.4.4 OPTICS

Ordering points to identify the clustering structure (OPTICS), proposed by Ankerst et al. [40], addresses the problem of detecting meaningful clusters in data of varying density. The algorithm is inspired on

DBSCAN, with the addition of two new concepts that we proceed to de ne. On the following de nitions, we maintain the nomenclature used so far.

De nition 3.4.6. Core-distance: Let minP ts -distance(p) be the distance from a point p to its minP ts ' neighbor. Then, the core-distance of p is de ned as

$$\text{core-dist}_{\varepsilon;\text{minPts}}(p) = \begin{cases} \text{Undefined}; & \text{if } |N_\varepsilon(q)| < \text{minPts} \\ \text{minPts-distance}(p); & \text{otherwise} \end{cases} \quad (3.7)$$

The core-distance of a point p is, simply put, the smallest distance ε' between p and a point in its ε-neighborhood such that p is a core point with respect to ε'. If p is not a core point in the rst place, the core-distance is not de ned.

De nition 3.4.7. Reachability-distance: The reachability-distance of p with respect to o, described as $\text{reachability-dist}_{\varepsilon;\text{minPts}}(p;o)$ is de ned as

$$\text{reachability-dist}_{\varepsilon;\text{minPts}}(p;o) = \begin{cases} \text{Undefined}; & \text{if } |N_\varepsilon(o)| < \text{minPts} \\ \max(\text{core-distance}(o); \text{distance}(o;p)); & \text{otherwise} \end{cases} \quad (3.8)$$

The reachability-distance of a point p with respect to a point o can be interpreted as the smallest distance such that p is directly density-reachable from o. This requires that o is a core point, meaning that the reachability-distance cannot be smaller than the core-distance. If that was the case, o would not be a core point.

Figure 3.4 represents the concepts of core-distance and reachability-distance just described. The scenario represented de nes minP ts = 5 and ε = 6 mm. On the left image, the core-distance, represented as ε', portrays the minimum distance that makes p a core point. For a radius of $\varepsilon' < \varepsilon$, there already exist ve points (minP ts) within the circle. The concept of reachability-distance is represented on the right image. The reachability-distance between q_1 and p must assure p is a core point, although the distance between the two points is actually smaller. The reachability-distance between q_2 and p, corresponds to the actual distance between the two points because any distance larger than ε' already assures p is a core point.

The OPTICS implementation is fundamentally equivalent to DBSCAN, with the exception that instead of maintaining a set of known, but so far unprocessed cluster members, a priority queue is used. We do not include the pseudo-code here, but the interested readers may refer to [40].

The algorithm returns the points and the corresponding reachability-distances. The points returned are ordered in such a way that spatially closest points become neighbours in the ordering. From this information, a reachability plot can be constructed, by arranging the ordered points in the x-axis and the reachability-distance on the y-axis, as shown in the bottom of Figure 3.5. The upper left plot represents the original dataset, whereas the upper right plot depicts the spanning-tree produced by OPTICS. The

Figure 3.4: Illustration of core-distance and reachability-distance in OPTICS. Source: [40].

colours simply aim to facilitate the visualization of different clusters. Since points belonging to a cluster are closer from each other, they have a low reachability-distance to their nearest neighbour. As a consequence, the clusters are pictured as valleys in the reachability plot, which will be deeper for denser clusters. Additionally, the yellow points are considered noise, and no valley is found in their reachability plot.

Figure 3.5: Extracting clusters using OPTICS. Adapted from: [41].

Having described the reachability plot framework for cluster analysis, we may now describe the clustering extraction procedure. It can be performed mainly in two ways. The rst is to inspect the reachability plot and x a suitable " on the y-axis. This way, the chosen value is imposed for every cluster in the dataset, which simply yields the same results as applying DBSCAN with the same value for " and minP ts . The second alternative consists in letting an automatic procedure select the most suitable " for each cluster. This is performed by de ning the minimum steepness on the reachability plot that should constitute a cluster boundary. Ankerst et al. [40] describes in detail how this process can be accomplished, with the

introduction of some additional new concepts. From this work's point of view, the key idea to retain is that this procedure addresses the problem of nding the most suitable " for each cluster. As a byproduct, it makes the clustering process closer to being parameterless, as the investor only needs to specify minP ts . Note that, in practice, the automatic cluster extraction procedure requires information regarding the minimum steepness on the reachability plot necessary to detect a cluster boundary. However, the default value proposed by the library implementation is adopted as it is independent of the data. Hence, the investor does not need to worry about this value.

3.5 Pairs selection criteria

Having generated the clusters of assets, it is still necessary to de ne a set of conditions for selecting the pairs to trade. It is critical that the pairs' equilibrium persists. To enforce this, we propose the uni cation of methods applied in separate research work. According to the proposed criteria, a pair is selected if it complies with the four conditions described next:

1. The pair's constituents are cointegrated.
2. The pair's spread Hurst exponent reveals a mean-reverting character.
3. The pair's spread diverges and converges within convenient periods.
4. The pair's spread reverts to the mean with enough frequency.

We proceed to go over each step with more detail to describe the reasoning behind. First, as described in section 2.2.3, cointegration methods proved to achieve good performance in pairs selection, as cointegration based approaches identify econometrically more sound equilibrium relationships when compared to distance/correlation based methods. Thus, a pair is only deemed eligible for trading if the two securities that form the pair are cointegrated. To test this condition, we propose the application of the Engle-Granger test, due to its simplicity. One critic Armstrong [14] points out to the Engle-Granger test has to do with the fact that the choice of the dependent variable may lead to different conclusions. To mitigate this issue, we propose that the Engle-Granger test is run for the two possible selections of the dependent variable and that the combination that generated the lowest t-statistic is selected.

Secondly, an additional validation step is suggested to provide more con dence in the mean-reversion character of the pairs' spread. It aims to constrain false positives, possibly arising as an effect of the multiple comparisons problem. The condition imposed is that the Hurst exponent associated with the spread of a given pair is enforced to be smaller than 0.5, assuring the process leans towards mean-reversion. This additional step ensures that undesirable data samples that made it through the cointegration test but are not mean-reverting can now be detected and discarded. This criterion is adapted from the work of Ramos-Requena et al. [22], in which the authors claim that cointegration per se is too restrictive of a condition to impose and that is rarely ful lled (something that is also mentioned in [9]). For that reason, the authors propose ranking the pairs based on the Hurst exponent and choose them accordingly. The perspective assumed in this work is different, as we still impose that two legs forming a pair are cointe-

grated, but we use the Hurst exponent value as a second check to look for mean-reverting properties.

It should be noted that the Hurst exponent measures the degree of mean-reversion of a time series, and not its stationarity. Strictly speaking, this means that it cannot be interpreted as a direct replacement of the cointegration test. Although in practice, most stationary time series are also mean-reverting, there may be exceptions. As an example, we urge the reader to consider the process X which veri es $X_t = X_{t-1}$ for $t > 0$. Let X_0 take the value 1 with probability 0.5 and 0 otherwise. X is stationary but not mean reverting, proving stationarity does not imply mean-reversion. Nevertheless, these situations are rare, and for that reason, we neglect them in our approximation.

In third place, as discussed in section 2.1, the spread's stationarity is a desirable property to look for in the selected pairs. However, a mean-reverting spread by itself does not necessarily generate pro ts. There must be coherence between the mean-reversion duration and the trading period. If a spread takes on average 400 days to mean-revert, but the trading period lasts one year, it is unlikely that a pro table situation is encountered. Likewise, a very short mean-reversion period is not desirable either. Since the half-life can be interpreted as an estimate of the expected time for the spread's mean-reversion, we propose ltering out pairs for which the half-life is not coherent with the trading period.

Lastly, we enforce that every spread crosses its mean at least once per month, to provide enough liquidity. It should be noted that even though there is logically a (negative) correlation between the number of mean crosses and the half-life period, as more mean crosses are naturally associated with a shorter half-life, both properties are not necessarily interchangeable. Adding this constraint might not merely enforce the previous condition but also discard pairs that in spite of meeting the mean-reversion timing conditions, do not cross the mean, providing no opportunities to exit a position.

Having described the four stages proposed to select an eligible pair, we proceed to describe the parameters used at each stage. Figure 3.6 represents the four rules a pair must verify to be eligible for trading. First, it is imposed that pairs are cointegrated, using a p-value of 1%. Then, the spread's Hurst exponent, represented by H should be smaller than 0.5. Additionally, the half-life period, represented by hl, should lay between one day and one year. Finally, it is imposed that the spread crosses a mean at least 12 times per year, which ideally corresponds to a minimum of one cross per month, on average.

Figure 3.6: Pairs selection rules.

3.6 Framework diagram

Up to this point, we described in detail the three building blocks of the proposed pairs selection framework. In this section, we intend to illustrate how they connect.

Figure 3.7 illustrates the progress along the three pairs stages. As we can observe, the initial state should comprise the price series for all the possible pairs' constituents. We assume this information is available to the investor. Then, by reducing the data dimensionality, each security may be described not just by its price series but also by the compact representation emerging from the application of PCA in the return series (State 1). Using this simpli ed representation, the OPTICS algorithm is capable of organizing the securities into clusters (State 2).

Finally, we may search for pair combinations within the clusters and select those that verify the rules described in Figure 3.6 (State 3).

Figure 3.7: Pairs selection diagram.

3.7 Conclusion

In this chapter, we started by motivating the exploration a novel way of grouping nancial securities to search for pairs. Next, we proposed a compact data-driven scheme to accomplish that , and explored each of its stages. We described how the application of dimensionality reduction followed by a density-based clustering algorithm could help to tackle the previously stated issues. In particular, we presented the OPTICS method as being convenient for the conditions under study. We also de ned a meticulous

set of rules to de ne which pairs are eligible for trading. Some important take-away points from the proposed framework are summarized in Table 3.1.

Table 3.1: Key points from the proposed pairs selection framework.

Chapter 4

Proposed Trading Model

In the previous chapters, we described the background of Pairs Trading and proposed a new framework for selecting pairs. This chapter explores how the trading strategy may be improved, as part of research stage 2.

4.1 Problem statement

One downside of the threshold-based trading model, de ned in section 2.3.1, is that the entry points are not de ned with much precision. The only criterion to enter a position is crossing a prede ned threshold, independently of the spread's current direction. This can result in unpleasant portfolio decline periods, in case a pair continues to diverge, as depicted in Figure 4.1.

Figure 4.1: Illustration of an untimely entry position.

The solid red line in Figure 4.1 corresponds to the entry point de ned by the model, which is triggered when the spread crosses the short threshold. Because the spread continues to diverge, a loss is incurred

in the portfolio. Gains will only manifest once the spread starts converging, in the point marked by the dashed red line, at the end of February. In addition, entering the position at the point when convergence begins provides a larger pro t margin since the spread distance to the mean is bigger. Hence, not optimizing the entry points does not only cause longer portfolio decline periods, but it can also reduce the full potential one could get from the spread convergence.

4.2 Proposed model

To create a more robust trading model, it would be interesting to make use of predicted future data when de ning the market entry positions. Initially, we conceive three different solutions based on time series forecasting. From these three approaches, which we proceed to describe, we select the most suitable based on empirical testing.

One possible approach could be the one followed by Dunis et al. in [23, 24, 27]. This approach attempts to model the spread predicted returns directly, which the author de nes as

$$R_t = \frac{Y_t \quad Y_{t \quad 1}}{Y_{t \quad 1}} \quad \frac{X_t \quad X_{t \quad 1}}{X_{t \quad 1}}; \tag{4.1}$$

where X_t and Y_t represent the prices of the two legs composing the pair, at time t. This way, when the predicted return is higher than a prede ned threshold (optimized in-sample), a position is entered.

Another possibility is to focus on modelling the spread series instead and use this information for trading. The investor may track the spread's recent values and look for a pattern of local trendiness. In case an uptrend is detected, the investor can go long on the spread. Similarly, if a downtrend is identi ed, the investor can go short on the spread. Eventually, a position is exited once the predicted value for the next time instant contradicts the trend's direction. This strategy is exempli ed in Figure 4.2. The coloured triangles express the forecasting input, used to indicate whether the spread is expected to go up (positive forecast) or down (negative forecast). For the sake of illustration, the forecasting has an accuracy of 100%, enabling the positions to be exited at the exact right moment. Naturally, this is extremely hard to achieve in practice.

Figure 4.2: Local momentum forecasting-based strategy.

Both methodologies described, even if theoretically appealing, did not perform well in practice. In section 6.3.6, some details concerning why these two methodologies did not succeed are presented. From this point on, we focus on the approach that proves more reliable. This approach also focuses on forecasting the spread series, de ned according to equation (2.11). In this case, it continuously monitors the percentage change between the spread at the current time, and the predicted spread in the next time-step. When the absolute value of the predicted change is larger than a prede ned threshold, a position is entered, on the expectation that the spread suffers an abrupt movement from which the investor can bene t from. We proceed to describe the workings of this methodology more formally. We de ne S_t and S_t as the true and predicted value for the spread at time t, respectively. The predicted change is then given by equation (4.2a). Having calculated the spread predicted change, the position settlement conditions work as described in equation (4.2b), where $_L$ and $_S$ correspond to the long and short position trigger thresholds respectively. Once a position is entered, it is held while the predicted spread direction persists and closed when it shifts.

$$\text{Predicted Change}: \quad {}_{t+1} = \frac{S_{t+1} \quad S_t}{S_t} \quad 100 \tag{4.2a}$$

$$\text{Market entry conditions}: \quad \begin{cases} \text{if} \quad {}_{t+1} \quad {}_L; & \text{open long position} \\ \text{if} \quad {}_{t+1} \quad {}_S; & \text{open short position} \\ \text{otherwise}; & \text{remain outside market} \end{cases} \tag{4.2b}$$

It is yet to be described how the thresholds ($_L$, $_S$) should be de ned. A possible approach could consist of framing an optimization problem and try to nd the pro t-maximizing values. However, this approach is rejected due to its risk of data-snooping and unnecessary added complexity. We propose a simpler, non-iterative, data-driven approach to de ne the thresholds. For each spread series, we start by obtaining f (x), the spread percentage change distribution during the formation period. The spread percentage change at time t is de ned as

$$x_t = \frac{S_t \quad S_{t \quad 1}}{S_{t \quad 1}} \quad 100.$$

From the distribution f (x), the set of negative (negative percentage changes) and positive (positive percentage changes) values are considered separately, originating the distributions f (x) (illustrated in 4.3(a)) and $f^+(x)$ (illustrated in 4.3(b)), respectively. Since the proposed model targets abrupt changes but also requires that they occur frequently enough, looking for the extreme quantiles seems adequate. As quantiles adapt to the spread volatility, the thresholds will ultimately be linked to the spread's properties. As Figure 4.3 illustrates, we recommend selecting the top decile and quintile[1] from $f^+(x)$ as candidates for de ning $_L$ and the bottom ones, from f (x), for de ning $_S$. Then, the quintile-based and decile-based thresholds are both tested in the validation set, and the most optimistic combination is adopted. The reason why we also consider quintiles and not just deciles is that we fear that the deciles

[1] A quintile is any of the four values that divide the sorted data into ve equal parts, so that each part represents $\frac{1}{5}$ of the sample or population. It takes any of the values $Q_X(0:20); Q_X(0:40); : : : ; Q_X(0:80)$.

(a) Spread percentage change distribution between $x(t)$ and $x(t \quad 1)$, for negative values, $f \quad (x)$.

(b) Spread percentage change distribution between $x(t)$ and $x(t \quad 1)$, for positive values, $f^{+}(x)$.

Figure 4.3: Spread percentage change distributions.

might be too restrictive. The process just described can be presented more formally as

$$f \quad {}_{S}; \quad {}_{L}\, g = \underset{q}{\operatorname{argmax}}\ R^{val}(q); \quad q \ 2 \quad \left[Q_{f^{-}(x)}(0{:}20); Q_{f^{+}(x)}(0{:}80) \ ; \quad Q_{f^{-}(x)}(0{:}10); Q_{f^{+}(x)}(0{:}90) \right] : \quad (4.3)$$

Figure 4.4 illustrates the application of this model. The colored symbols are used to express the information provided by the forecasting algorithm. The triangles indicate one of the thresholds has been triggered and the corresponding direction, whereas the squares only provide information concerning the predicted direction. Once again, for the sake of illustration, the forecasting has perfect accuracy, meaning the positions can be set in optimal conditions.

Figure 4.4: Proposed forecasting-based strategy.

As a nal note, it is worth emphasizing that the methodology just proposed is fundamentally different from the benchmark trading strategy presented in section 2.3.1. Because the underlying motivation for entering a position is distinct in both models, it is not expected any relation between the entry and exit points of the two strategies.

4.3 Model diagram

The proposed model can be applied in line with Figure 4.5. As depicted, the investor should start by training the forecasting algorithms to predict the spreads. Furthermore, the decile-based and quintile-based thresholds should be collected to integrate the trading model. Having tted the forecasting algorithms

and obtained the two combinations for the thresholds (State 1), the model can be applied on the validation set. From the validation performance, the best threshold combination is selected (State 2). At this point, the model is nally ready to be applied on unseen data, from which the performance may be inferred (State 3).

Figure 4.5: Proposed model application diagram.

4.4 Time series forecasting

Motivated by the potential of applying a time series forecasting based trading strategy, we proceed to describe which alternatives may be more suitable for the task at hand. Different models have been applied in the literature for time series forecasting. These models can be divided into two major classes, parametric and non-parametric models.

Parametric models appear as the early time series predictions and assume that the underlying process has a particular structure, which can be described using a small number of parameters. In these approaches, the goal is to estimate the parameters of the model that best describe the stochastic process. However, these models' representation capacity have severe limitations [42], creating the need to search for a modelling approach with no structural assumptions.

Nonparametric modelling is especially attractive because it makes no structural assumptions about the underlying structure of the process. Arti cial Neural Networks (ANN) are an example of non-parametric models. Several factors motivate their use. First, ANNs have been an object of attention in many different elds, which makes its application in this context an interesting case study. Furthermore, ANN-based models have shown very promising results in predicting nancial time series data in general [1]. Finally, Machine Learning techniques are fairly unexplored in this eld, as analyzed in section 2.4. This encourages its application in this work.

In conclusion, we propose the application of a benchmark parametric approach, along with other non-parametric models found relevant. Each of the proposed algorithms is detailed next.

4.5 Autoregressive Moving Average

As stated by Si and Yin [43], nancial time series are inherently complex, highly noisy and nonlinear. These aspects naturally motivate the use of ANNs for forecasting, given their capability to address such conditions. However, the nancial time series under analysis do not necessarily t the characterization above. By construction, the spread series is stationary during the formation period. This is a requirement, given that a pair is only selected if both constituents are cointegrated, which consequently implies that its spread is stationary. Therefore, even though stationarity in the trading period cannot be guaranteed, because it corresponds to unseen data, the series is surely more likely to be so. It is thus fair to ask if a simpler model could perform this task just as effectively. Moreover, it would be interesting to determine whether the extra model complexity from neural networks is worthwhile. The application of an autoregressive moving-average (ARMA) model for forecasting is proposed. The use of an ARMA model also allows inferring to what extent the pro tability of the strategy depends on the time series forecasting algorithm itself, or if it is robust under different forecasting conditions.

The ARMA model was rst described by Whitle [44]. This model describes a stationary stochastic process as the composition of two polynomials. The rst polynomial, the autoregression (AR), intends to regress the variable at time t on its own lagged values. The second polynomial, the moving average (MA), models the prediction error as a linear combination of lagged error terms and the time series expected value. We proceed to describe in a formal way how each of the polynomials models a time series X_t. The AR(p) model is described as

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t; \tag{4.4}$$

where p is the polynomial order, $\varphi_1, \ldots, \varphi_p$ are the model parameters, c is a constant, and ε_t is a random variable representing white noise. The MA(q) model is described as

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}; \tag{4.5}$$

where q is the polynomial order, $\theta_1, \ldots, \theta_q$ represent the model parameters and μ represents the mean value of X_t. The variables $\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}$ correspond to white noise error terms in the corresponding time instants. Finally, the ARMA(p,q) model results from the combination of (4.4) and (4.5), and can be represented as

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}. \tag{4.6}$$

4.6 Arti cial Neural Network models

Although the simplicity of the ARMA model just described is advantageous from a practical point of view, it might also be interpreted as a limitation for learning more complicated data patterns. Arti cial Neural Networks have proved particularly ef cient in this regard. From the vast amount of existing ANNs con gurations, we proceed to select the two architectures deemed more appropriate for this work.

4.6.1 Long Short-Term Memory

One con guration considered suitable is the Long-Short Term Memory (LSTM). The LSTM is a type of Recurrent Neural Network (RNN). Therefore, RNNs should rst be described. Unlike standard feedforward neural networks (FNN), RNNs can process not only single data points but also entire sequences of data [45]. This means that while in a traditional FNN, inputs are assumed to be independent of each other, in the case of an RNN, a sequential dependency is assumed to exist among inputs, and previous states might affect the decision of the neural network at a different point in time, as represented in Figure 4.6. This is achieved with inner loops, allowing RNNs to have a memory of their previous computations.

Figure 4.6: Recurrent Neural Network structure.

In Figure 4.6, W_i and b_i represent the neural networks weights, and the bias term respectively. The subscript i can take the values of x for the input, h for hidden and o for output. The hidden layer's RNN cell, represented by the grey rectangle, is a block which repeats (unfolds) along the time dimension. This block receives an input and together with the previous state computes the next state and the output at the current time step. The two cell outputs can be described as

$$\begin{cases} h_t = \quad (W_h h_{t\ \ 1} + W_x x_t + b_h) \\ o_t = \quad (W_o h_t + + b_o) \end{cases} ; \tag{4.7}$$

where represents the layer's activation function. A problem arising with the RNN architecture just presented is the dif culty in propagating the prediction error. To understand why we rst describe the standard backpropagation in regular FNNs. In that case, the backpropagation moves backwards from the nal error through the outputs, weights, and inputs of each hidden layer, considering those weights accountable for a portion of the error, by calculating their partial derivatives $\frac{@E}{@w_i}$, where E represents the

error function. Those derivatives are then used to adjust the weights in whichever direction decreases error. To train the RNN models, the same process applies, by representing the network on its unfolded version, as in Figure 4.6. However, when propagating the error along the entire network's time dimension, the gradient might vanish or explode, depending on whether the derivative of the state update function decreases or increases the backpropagation error too drastically. In the rst case, the learning process takes too long to complete, whereas the second case leads to numerical issues.

The aspect described is partly mitigated with LSTMs [46] by selectively updating the internal state as a function of the current state and the input, through a gating mechanism. The LSTM cell structure is represented in Figure 4.7.

Figure 4.7: Long Short-Term Memory cell. Source: [47].

In this topology, the internal state and current input de ne the input, output and forget gate's ow of information. The input gate de nes the in uence of the new input on the internal state. The forget gate de nes which parts of the internal state are retained over time. Finally, the output gate de nes which part of the internal state affects the output of the network. Formally, the equations describing the LSTM cell can be summarized as

$$\begin{cases} f_t = \quad W_x^f x_t + W_h^f h_{t\ \ 1} + b_f \\ i_t = \quad W_x^i x_t + W_h^i h_{t\ \ 1} + b_i \\ o_t = \quad (W_x^o x_t + W_h^o h_{t\ \ 1} + b_o) \\ c_t = f_t \quad c_{t\ \ 1} + i_t \quad (W_x^c x_t + W_h^c h_{t\ \ 1} + b_c) \\ h_t = o_t \quad (c_t) \end{cases} ; \tag{4.8}$$

where f_t, i_t and o_t represent the forget, input and output gate respectively. By selectively accepting new information and erasing past one, this topology is able to maintain the error ow across the network.

4.6.2 LSTM Encoder-Decoder

From a trading perspective, it might be bene cial to use information regarding the prediction not just of the next time instant, but also of later time steps. To accomplish that, one might use the data collected

up to time t to directly predict the next N time steps $(t+1;\dots;t+N)$ at once. An improvement may consist of using not just the data collected up to time t, but also bene t from the new predicted data. For instance, to predict the time series value at time $t+2$, one would use the data collected up to t with the addition of the prediction for $t+1$.

An LSTM Encoder-Decoder architecture is naturally tted to such scenarios. This architecture is composed by two groups of LSTMs, one for reading the input sequence and encoding it into a xed-length vector, the encoder, and a second for decoding the xed-length vector and outputting the predicted sequence, the decoder. The two networks are trained jointly to minimize the forecasting loss with respect to a target sequence given the source sequence. The architecture of a LSTM based Encoder-Decoder can be observed in Figure 4.8. A sequence $x_1;\dots;x_t$ is inputted sequentially to the Encoder LSTM, which aggregates the relevant information in the encoder vector. Then, this information is used to forecast the series for the next N time-steps. Naturally, it is expected that the error increases with N.

Figure 4.8: LSTM Encoder-Decoder.

In this multi-step forecasting scenario, the trading rules de ned in section 4.2 need to be updated. The prediction change should be calculated N times in advance. Likewise, the thresholds $_L$ and $_S$ should be calculated with respect to the distribution of the percentage change between $x(t)$ and $x(t \quad N)$.

4.7 Arti cial Neural Networks design

The Arti cial Neural Network models demand a laborious design process. In this section, we highlight some techniques adopted, which we consider especially relevant to describe.

4.7.1 Hyperparameter optimization

One of the disadvantages pointed out to Deep Learning models is the cumbersome process associated with optimizing the network's con guration. Tuning the hyperparameters can have a tremendous in uence on its result, but it is also very time-consuming. Thus, a compromise must be achieved.

Due to the limited computation resources, the tuning of the LSTM models is constrained to a set of most relevant variables: (i) sequence length, (ii) number of hidden layers, and (iii) nodes in each hidden layer. We experiment increasingly complex con gurations until signs of over tting are detected.

4.7.2 Weight initialization

Neural networks weights are very often initialized randomly. This initialization may take a fair amount of iterations to converge to the optimal weights. Moreover, this kind of initialization is prone to vanishing or exploding gradient problems. One way to reduce this problem is to select a random weight initialization carefully. In this work, we adopt the glorot weight initialization, proposed by Glorot and Bengio [48]. The authors introduce a weight initialization algorithm which factors into the equation the size of the network (number of input and output neurons). The proposed methodology does not only reduce the chances of running into gradient problems but also brings substantially faster convergence.

This initialization, also called Xavier initialization, suggests sampling the layer weights in a way that preserves the input variance, such that the variance remains constant as the information passes through each layer of the neural network. We proceed to describe how this can be accomplished. First, the variance is given by

$$Y = \sum_{N_{in}} W_i X_i \ , \quad \mathrm{Var}(Y) = \mathrm{Var}\left(\sum_{N_{in}} W_i X_i \right) ; \tag{4.9}$$

where N_{in} represents the number of input nodes. If the input variables are assumed to be uncorrelated, we may consider

$$\mathrm{Var}\left(\sum_{N_{in}} W_i X_i \right) = \sum_{N_{in}} \mathrm{Var}(W_i X_i) : \tag{4.10}$$

Finally, assuming the inputs and weights are uncorrelated and that the variance of both inputs and weights is identically distributed, (4.10) further simpli es according to

$$\sum_{N_{in}} \mathrm{Var}(W_i X_i) = \sum_{N_{in}} \mathrm{Var}(W_i)\,\mathrm{Var}(X_i) = N_{in}\ (\mathrm{Var}(X_i)\,\mathrm{Var}(W_i)) : \tag{4.11}$$

Thus, to preserve the input variance, $\mathrm{Var}(X_i)$, the weights variance should be given by $\frac{1}{N_{in}}$. To accomplish this, a Gaussian sampling distribution with zero mean and a variance given by $\frac{2}{N_{in} + N_{out}}$ is proposed. This corresponds to the geometric mean between the forward propagation pass variance, $\frac{1}{N_{in}}$, and back propagation pass variance, $\frac{1}{N_{out}}$.

4.7.3 Regularization techniques

The functions that de ne neural network topologies are generally non-convex [45]. Therefore, the training algorithm does not give us a formal guarantee of convergence to the optimal solution. The imposition of regularization terms in the optimization process aids tackling this issue. Regularization refers to the introduction of additional constraints, potentially derived from problem-speci c information, to solve an

ill-posed problem or avoid over tting. More speci cally, it helps to reduce the generalization error by making the gap between the errors observed in the training set and validation set less evident. At the same time, by introducing additional information, the convergence rate can increase [45]. We proceed to describe two regularization techniques adopted in this work, Early-stopping and Dropout.

Early-stopping is likely the most commonly used form of regularization in Deep Learning due both to its effectiveness and its simplicity [45]. The foundation of Early-stopping goes as follows. When training large models, with suf cient representational capacity to over t, it is often visible that training error decreases steadily over time. However, the validation set error begins to rise again at some point. This behaviour is illustrated in Figure 4.9. In this gure, the described pattern is particularly evident.

Figure 4.9: Early Stopping illustration.

By accounting for this pattern, a model with a better validation set error (and hopefully a better test set error) can be obtained. To achieve that, we should return to the parameter setting with the lowest validation error, identi ed by the black triangle in Figure 4.9. This can be accomplished by maintaining a copy in memory of the model parameters that so far generated the best validation error. When the training terminates, the algorithm returns the stored parameters, rather than the most recent ones. This process can be made more ef cient if the system terminates the training procedure after detecting no improvement in the validation loss within a number of p iterations after the best parameterization, rather than having to wait for the training to terminate. The parameter p is usually called patience.

The second regularization technique adopted is Dropout. Its fundamental idea is based on the premise that the risk of over tting may be reduced by tting different neural networks to the same dataset and use the average prediction of all networks as the nal prediction. The view is that noise learned by a speci c model architecture is attenuated when averaged over all models. However, the process of training multiple models can be very challenging, especially when dealing with such complex models. Dropout emerges as a more practical solution to emulate this. It approximates the training of a large number of neural networks with different architectures in parallel, while only training one neural network. This is accomplished by randomly ignoring some layer nodes during the training period. This has the effect of treating the layers like they have a different number of nodes and connectivity to the prior layer. Hence, each update during training is performed with a different perspective of the con gured layer.

Figure 4.10 represents on the left side the standard neural network con guration and on the right side

one con guration that may be encountered during the training process.

(a) Standard Neural Network

(b) Neural Network with dropout.

Figure 4.10: Application of dropout. Source:[49].

4.8 Conclusion

In this chapter, we started by emphasizing the motivation for exploring a new trading strategy. We advanced with the description of the proposed trading model. Next, we explored the potential forecasting algorithms to integrate it. Namely, we stated that the application of an ARMA model, a regular LSTM neural network and an LSTM Encoder-Decoder seem adequate in this context. In addition, we discussed some technical aspects concerning the design of the ANN-based models. Table 4.1 compiles some key points concerning the proposed trading model.

Table 4.1: Key points from the proposed trading model.

Chapter 5

Implementation

So far we discussed the state-of-the-art in Pairs Trading (Chapter 2), and proposed a new framework for selecting pairs (Chapter 3) and a new trading model (Chapter 4). This chapter discusses some speci c research implementation aspects relevant to test the two recommended methods. It informs the reader about the employed dataset, the test setup for each research stage and the constructed trading environment.

5.1 Research design

In this work, we propose a methodology for each phase composing a Pairs Trading strategy, the pairs selection phase (Research Stage 1) and trading phase (Research Stage 2). Thus, we design a dedicated research setup for each, as illustrated in Figure 5.1.

Figure 5.1: Research design overview.

One convenient research scheme consists of responding to this work's rst research question, and apply the optimal pairs clustering technique just found to answer the second research question. This is represented by the greyed arrow in between. The coloured blocks intend to facilitate the visualization of the Pairs Trading phase being tested.

5.2 Dataset

In this section, we present the dataset suggested for simulating the proposed implementation. We start by describing which type of securities we nd appropriate to explore. Next, we describe the resulting dataset and how it is prepared for application. Finally, we detail how the data is partitioned in several periods, to be used in different experiments.

5.2.1 Exchange-Traded Funds

Exchange-Traded Funds (ETFs) are considered an interesting type of securities to explore in this work. An ETF is a security that tracks an index, a commodity or a basket of assets just like an index fund, but trades like a stock. According to Chan [9], the one advantage of trading ETF pairs instead of stock pairs is that once found to be cointegrated, ETF pairs are less likely to fall apart in out-of-sample data. This is because the fundamental economics of a basket of stocks changes much slower than that of a single stock. This robustness is very appealing in the context of Pairs Trading, in which the fundamental premise is that securities whose prices are cointegrated today should remain cointegrated in the future.

To further support the hypothesis that ETFs are convenient in a Pairs Trading setting, the work of Chen et al. [19] and Perlin [50] are an important reference. To better comprehend the authors' ndings, some nomenclature should be introduced rst. A strategy in which both constituents of the pair are single securities is called univariate. Whereas if one side of the pair is formed by a weighted average of comoving securities, the strategy is named quasi-multivariate. Reasonably, when both sides of the spread are formed by a weighted average of a group of securities, the strategy is said to be multivariate, as represented in Figure 5.2.

(a) Univariate strategy (b) Quasi-multivariate strategy (c) Multivariate strategy

Figure 5.2: Exempli cation of Univariate, Quasi-Multivariate and Multivariate strategies.

Proceeding with the analysis of the aforementioned work, [19] and [50] attempt to produce more robust mean-reverting time series by using a linear combination of stocks to form each leg of the spread, on the prospect that one outlier stock will not affect the spread so drastically. In speci c, Chen et al. [19] applies a Pairs Trading strategy based on Pearson correlation both in a univariate and a quasi-multivariate setup. In the former, the two most correlated stocks form a pair. In the latter, the pair is formed by a stock and an equal-weighted portfolio of the 50 most correlated stocks. Results show that the quasi-multivariate trading strategy obtains a better performance, whereas reducing the number of stocks in the weighted portfolio causes the returns to drop up to two thirds. Perlin [50] also evaluates the ef ciency of a univariate strategy against a quasi-multivariate strategy, for the 57 most liquid stocks in the Brazilian market. In the univariate scenario, the two most correlated stocks are identi ed as a pair, whereas in the quasi-multivariate scenario, one stock is paired with ve partner stocks. The results obtained show that the quasi-multivariate strategy outperforms the univariate strategy, once again. Nevertheless, both studies mentioned in this paragraph see their results affected by the high associated transaction costs of purchasing multiple securities, which disincentive the application of a multivariate strategy. In this work, we suggest using ETFs as a proxy to obtain the bene ts of multivariate strategies more practically. The reasoning is that an ETF is fundamentally a weighted average of a set of securities, and thus, trading ETFs may be seen as a more practical way of running a multivariate pairs trading strategy.

On top of the motivations described, the universe of ETFs has one extremely interesting particularity. The rising popularity of ETFs motivates the creation of new ETFs every year that just marginally differ from existing ones. These price series naturally form good potential pairs.

Finally, few research studies explore the application of Pairs Trading in the ETF universe, with some exceptions [34, 51], which makes it even more appealing to try the suggested approaches using ETFs.

5.2.2 Data description

This work focuses on a subset of ETFs that track single commodities, commodity-linked indexes or companies focused on exploring a commodity. Constraining the ETFs to this subset reduces the number of possible pairs, making the strategy computationally faster and leaving space for a careful analysis of the selected pairs. More speci cally, the set of ETFs considered is composed only by commodity-linked ETFs that are available for trading in January 2019. To nd this information, the data source [52] proved useful.

A total of 208 different ETFs verify the conditions just described. This universe of commodity-linked ETFs can be categorized according to the type of commodities being tracked. We propose ve different categories, as shown in Table 5.1. Appendix A includes a table describing all the eligible ETFs.

It should be noted that by considering just the ETFs active throughout an entire period, survivorship bias is induced. By neglecting delisted ETFs, possible positions that would have had to be left abruptly are not being accounted for in the results. This is because no survivorship free dataset containing the

Table 5.1: Commodity categories considered in the dataset.

pretended ETFs was found. To attenuate this impact, recent periods are considered whenever possible.

Having selected the universe of ETFs to consider for trading, we proceeded to retrieve the price series for each ETF. We considered price series data with 5-min frequency. The motivation for using intraday data is three-fold. First, with ner granularity, the entry and exit points can be de ned with more precision, leaving space for higher pro ts margins. Secondly, we may detect intraday mean-reversion patterns that could not be found otherwise. Lastly, it provides more data samples, allowing to train complex forecasting models with less risk of over tting.

5.2.3 Data preparation

Before delving into the search of promising pairs, some data cleaning steps are implemented, as described in Figure 5.3.

Figure 5.3: Data preprocessing steps.

First, ETFs with any missing values are directly discarded. Then, we remove ETFs that do not verify the minimum liquidity requisites, to ensure the considered transaction costs associated with bid-ask spread are realistic[1]. The minimum liquidity requisites follow the criterion adopted in [17, 53], which discards all ETFs not traded during at least one day.

At last, some price series contain occasional outliers. In the literature, several techniques have been proposed to tackle the issue of anomaly detection. Resorting to methodologies such as cluster analysis based techniques, Hidden Markov Models or other Machine Learning techniques could be considered for this purpose. Nonetheless, because these events are infrequent in the considered dataset, a simpler technique is proposed. For each price series, the return series is calculated. Then, every point for which

[1]Trading illiquid ETFs would result in higher bid-ask spreads which could dramatically impact the pro t margins.

the percentage change is higher than 10%, which is extremely unlikely to occur in just 5 minutes, is considered an outlier. These situations are analyzed one-by-one. In case there is a single outlier in the time series of an ETF, this point is corrected manually by analyzing other data sources. When more outliers are present, the ETF is discarded.

Additionally, when training the forecasting-based models, the data is standardized according to

$$\overline{S}_t = \frac{S_t \quad {}_s}{{}_s}; \tag{5.1}$$

where $\overline{S}_t$ represents the result of standardizing the spread S_t. The resulting series has zero mean and unit variance. Standardizing the inputs can make the training of the neural networks faster and reduce the chances of getting stuck in local optima.

5.2.4 Data partition

For each trading simulation, the data must be partitioned in two periods. The formation period and the trading period. The formation period simulates the data available to the investor before engaging in any trade, and its role is three-fold. First, this period is used to nd the most appealing candidate pairs. Secondly, a smaller portion of the data, the validation set, is used to simulate how the strategy would have performed in recent times. Thirdly, and only applicable to the forecasting-based models, a portion of the formation period data, the training set, is used to train the forecasting algorithm[2]. The trading period is used to resemble how the implemented trading model would perform with future unseen data. Figure 5.4 illustrates how this periods are arranged.

Figure 5.4: Period decomposition.

It is enticing to consider several partitions of the dataset, since analyzing the trading results for different conditions provides more con dence in the statistical signi cance of the results obtained. The typical procedure in classical Machine Learning problems is to perform k-fold cross-validation. However, time series (or other intrinsically ordered data) can be problematic for cross-validation, as one wants to avoid the look-ahead bias that may arise from training the model with data not available at a given historical date. To tackle this issue, a moving training and test window are used sequentially. Forward chaining

[2]To be more precise, the training set is used to train the ANN-based models, whereas the ARMA model is trained on the entire formation period, as described in section 6.3.2.

could also be applied, but in that case, the training and test periods' size is not fixed. Hence, the results can be affected by the varying data size.

The periods considered for simulating each research stage are illustrated in Figure 5.5. There are essentially two possible configurations: (i) the 3-year-long formation periods, and (ii) the 9-year-long formation period. In both cases, the second-to-last year is used for validating the performance, before running the strategy on the test set. We define a 1-year-long trading period, based on the findings of [18], that claim the profitability can be increased if the initial 6-month trading period proposed in [17] is extended to 1 year.

Figure 5.5: Data partition periods.

The 3-year-long formation period is adopted when using the threshold-based trading model. Although it is slightly longer than what is commonly found in the literature[3], we decide to proceed on the basis that a longer period may identify more robust pairs. The 9-year-long formation period is used for simulating the forecasting-based trading model. In this case, more formation data is required to fit the forecasting algorithms. The first 8 years are used for training them, as indicated in Figure 5.5. In this case, the application of several test periods is not conceivable due to the computational burden of training the associated forecasting algorithms. The number of neural networks requiring training when using T periods can be calculated as

$$\sum_{i=1}^{T} M_i \quad N_i; \tag{5.2}$$

where M_i is the number of pairs found in the i-th formation period, and N_i is the number of model configurations to be tested in the i-th period (e.g. RNN, Encoder-Decoder, etc.). Therefore, studying several periods consumes a considerable amount of extra time. Because we want to take special attention to the configuration of each model, we find more prudent to use a single formation/test period [4], and utilize the available time to optimize the Deep Learning architectures and corresponding parameters meticulously.

[3] [17, 18, 25] use a 1-year-long formation period. [7] makes use of a 3-month formation period.

[4] Although it is not mentioned explicitly, we suspect the computational effort is also one motivation that led [15, 23] to consider a single displacement for training, validation, and test sets.

One question that naturally arises is whether the threshold-based model used to compare with the forecasting-based model, in research stage 2 (see Figure 5.1), should be tested under the same period (2009-2018) or in the conditions previously applied (3-year-long formation period). We suggest conducting both scenarios to distinguish which part of the results is explained by the trading model itself and what part is affected by the period duration. For this reason, the period ranging from 2015 to 2019 is selected to test the benchmark model.

Table 5.2 describes some relevant data concerning the data partitions described in this section. The majority of ETFs found are relatively recent. Consequently, when longer periods are considered, the number of active ETFs drops signi cantly.

Table 5.2: Dataset partitions.

5.3 Research Stage 1

This section describes how we propose to evaluate the pairs selection framework introduced as part of research stage 1. We suggest comparing the three pairs' search methods mentioned in this work: (i) searching for all combinations of pairs, (ii) searching for pairs within the same category and (iii) searching for pairs within clusters generated using OPTICS. Therefore, we proceed to detail the necessary steps to build this comparison setup.

5.3.1 Development of the pairs selection techniques

To compare each of the three different pairs' search techniques, each approach is built from scratch. We also develop all the pairs selection rules proposed in section 3.5. For information concerning the code, the reader may refer to Appendix F.

We nd relevant to emphasize that grouping by category requires establishing a mapping between every ETF and the corresponding group. To accomplish this, we start by retrieving the available information regarding the segment of each ETF, found in [52]. The segment represents the precise market to which the fund provides exposure to. From this information, a second mapping is constructed containing the correspondence between every segment and one of the ve categories described in Table 5.1.

5.3.2 Trading setup

Since the main goal of this research stage is to compare the results from the pairs selection techniques relative to each other, we do not worry about optimizing the trading model. Therefore, the standard threshold-based model described in section 2.3.1 is applied, with the parameters speci ed in Table 5.3.

Table 5.3: Threshold-based model parameters.

The model parameters used are the ones suggested by Gatev et al. [17], which serve as a reference to several studies in the eld. The spread's standard deviation, $_s$, and mean, $_s$, are calculated with respect to the entire formation period. The exit threshold is de ned as the mean value, meaning that a position is closed only when the spread reverts to its mean[5]. We are aware that there exists some research work which explicitly explores the optimization of the trading thresholds, such as [54, 55]. Also, some authors, as Dunis et al. [7], explore the application of sliding windows to constantly update $_s$ and $_s$. Nevertheless, we stick to this framework for simplicity.

5.3.3 Test portfolios

We implement three different test portfolios resembling probable trading scenarios. Portfolio 1 considers all the pairs identi ed in the formation period. Portfolio 2 takes advantage of the feedback collected from running the strategy in the validation set by selecting only the pairs that had a positive outcome. Lastly, Portfolio 3 corresponds to the situation in which the investor is constrained to invest in a xed number of k pairs. In such a case, we suggest selecting the top-k pairs according to the returns obtained in the

[5]Note that the parameters are simpli ed when working with the standardized spread, because in that case $_s$=0 and $_s$=1.

validation set. We consider $k = 10$, as it stands in between the choices of [17], which uses $k = 5$ and $k = 20$. By testing different portfolio constructions, we may not only nd the optimal clustering procedure but also evaluate its best condition of application. The three portfolios are represented in Figure 5.6.

Figure 5.6: Test portfolios.

It is worth emphasizing that even though selecting the pairs based on the performance on the validation set seems appropriate, there is no guarantee that the best performing pairs in the validation set will follow the same behaviour in the test set. The results will dictate the competence of this heuristic.

5.4 Research Stage 2

In this section, we describe the proposed implementation to gauge the potential of the forecasting-based trading model introduced, as part of research stage 2.

5.4.1 Building the forecasting algorithms

To compare the robustness provided by the standard threshold-based model with the proposed forecasting-based model, simulated using the ARMA model, an LSTM and an LSTM Encoder-Decoder, the three models are built from scratch. For more details concerning the implementation of these algorithms, the reader may refer to Appendix F. Due to the limited available resources and the complexity of training these models, there are severe constraints on the number of combinations we can experiment. For this reason, the architecture of each model is tuned in a manual iterative process based on the feedback from the results obtained. The training is stopped when the results evidence signs of over tting. Concerning the LSTM Encoder-Decoder architecture, we propose that it outputs two time-steps at a time. Forecasting the time series two steps in advance seems appropriate, as we fear that the precision deteriorates for more distant values.

A naive baseline de ning a lower bound objective is set to support the training process. The persistent forecasting algorithm is selected for this purpose. It uses the value at the previous time step to predict the expected outcome at the next time step, according to

$$\text{Naive} : Y_{t+1} = Y_t : \tag{5.3}$$

5.4.2 Test conditions

Comparing the forecasting-based models using the three different clustering techniques would be very costly. For this reason, as illustrated in Figure 5.1, we consider the application of the clustering methodology that proves to be more appealing. As for the test portfolios, we consider the option of applying the strategy exclusively to the pairs that obtained positive results during the validation period (Portfolio 2 in Figure 5.6). Relying on selecting the top 10 pairs also seemed an adequate choice. However, as we will see, it might be the case that the selected pairs are fewer than ten.

Figure 5.7 summarizes how the proposed methodology will be set aside with the threshold-based model in the simulation covering the period of January 2009 to December 2018.

Figure 5.7: Trading models interaction diagram.

5.5 Trading simulation

In this section, we describe the details regarding the trading simulation. We go over the aspects considered in the construction of the portfolio, the transactions costs, and the position settlement conditions.

5.5.1 Portfolio construction

We start by noting that the portfolios studied in this work may have varying sizes. Nevertheless, we establish that all pairs should be equally weighted in the portfolio. This way, the portfolio returns can be obtained by simply averaging the performance of all pairs, with no need to be concerned about relative proportions of the initial investment.

A natural question that follows is how the capital is allocated for each pair. In theory, if the same value is being held in the long and short position of a pair, the capital earned from the short position can entirely

cover the cost of entering the long position, and hence no initial nancing is needed[6]. This is called a self- nancing portfolio. Even though this assumption is reasonable in theory, in practice, it does not apply because a collateral[7] is always required for borrowing the security being shorted, making a zero initial investment unfeasible. Therefore, the necessary investment corresponds to the collateral, which we assume to be the value of the security being shorted. This amount allows the investor to enter a long position with that same value. By nancing long positions with proceeds from short positions, the investor is leveraging its investment. This leveraging approach is typical in hedge funds to increase the absolute return in the portfolio, and for that reason is considered in this work.

To simplify the calculations, it is particularly useful to consider a one dollar amount investment in each pair. This approach is undertaken by most authors in the eld ([7, 16, 17, 25, 34]), thus making it more suitable for comparison purposes as well. It is worth noticing that this approximation works on the assumption that an investor can buy fractions of trading ETFs, as the security itself is most probably not valued at exactly one dollar. Nonetheless, in this work, we adopted this assumption backed by the fact that the investor may always nd the least common multiple between the two security prices that allow him to take an equal-sized position in both ETFs.

Contrarily to some research work that intends to be dollar-neutral, as Gatev et al. [17] and Dunis et al. [7], which invest $1 in the long position and $1 in the short position, the methodology followed in this work respects the cointegration ratio between the two securities. Therefore, we propose that the value invested in X is times the value invested in Y. In these conditions, to xate a $1 initial investment, it needs to be assured that neither the long nor the short position costs more than $1. Formally, the condition is being imposed is that

$$\max(\text{leg1}; \text{leg2}) = \$1;$$

where leg1 and leg2 represent the capital invested in each leg of the pair. On this basis, we construct a framework illustrated in Figure 5.8.

Figure 5.8: Market position de nition.

As the trading progresses, we consider that all the capital earned by a pair in the trading period is reinvested in the next trade. For instance, if the rst time a pair trades it has a return of 5%, the second

[6]Assuming the capital from the security being shorted is earned immediately after shorting.

[7]Collateral is an asset that a lender accepts as security for a loan. If the borrower defaults, the lender can seize the collateral and resell it to recoup the losses.

time a trade is opened for that pair, the initial capital will be \$1.05, and not \$1 as before. This mechanism facilitates the calculation of the nal return.

5.5.2 Transaction costs

We assure that all the results presented in this work account for transaction costs. The transaction costs considered are based on estimates from Do and Faff [53]. The authors perform an in-depth study on the impact of transaction costs in Pairs Trading. Conservative estimates of all the inherent costs are provided. These estimates are also adopted by other authors in the eld, such as Huck and Afawubo [21]. The costs considered can be divided into three components, as described in Table 5.4[8]. The charge is described relative to the position size.

Table 5.4: Transaction costs considered.

Commission and market impact costs must be adapted to account for both assets in the pair. The costs associated with one transaction are calculated as represented in Figure 5.9.

Figure 5.9: Calculation of transaction costs.

[8]Basis point (bps) refers to a standard unit of measure in nance, 1 bps = 0.01%.

It should be noted that these costs are estimated by Do and Faff [53] between 2009 and 2012, and their work indicates the trend is clearly favourable, as transaction costs tend to reduce in more recent years. Thus, if the results are robust to the considered transaction costs, at the time of writing they would only improve.

5.5.3 Entry and exit points

One important subtlety regarding the trading execution is related to the prices considered when entering a position. In both the threshold-based and forecasting-based models, the system continuously monitors the prices to verify if a given threshold is triggered, in which case, a position is set. But since a trading system can not act instantaneously, there might be a small deviation in the entry price inherent to the delay of entering a position. It should be discussed how this situation is handled in the proposed simulation environment.

In line with [7, 17, 18], we assume a conservative delay of one period to enter a position. Hence, if a position is triggered at $t = i$, we only enter the position at time $t = i + 5\,\text{min}$ (recall the dataset considered has a frequency of 5 minutes). This way, we ensure the strategy is viable in practice. If it proves pro table under these conditions, then the results could only improve considering a shorter delay than ve minutes, which is easily conceivable in practice.

Concerning the de nition of exit points, this work does not comprehend an implementation of a stop-loss system, under any circumstances. This means a position is only exited if one of two conditions is veri ed. Either the pair converges or the trading period ends. In the last scenario, a position that was recently entered might deteriorate the total return, even though it could turn out to be pro table in the future. Nevertheless, this is the best approximation of how a hedge fund would announce its pro ts at the end of a given period.

5.6 Evaluation metrics

We proceed to describe the nancial evaluation metrics deemed more appropriate to analyze the proposed strategies: the Return on Investment (ROI), the Sharpe Ratio (SR) and the Maximum Drawdown (MDD).

5.6.1 Return on Investment

By de nition, the return on investment in its simplest form is given by the net pro t divided by the initial investment, as follows:

$$\text{ROI} = \frac{\text{Net Pro t}}{\text{Initial Investment}} \quad 100. \tag{5.4}$$

Recall that the portfolio construction designed in this work establishes the initial required investment is always \$1. This scenario has the advantage that the return at any point in time can also be interpreted as the net pro t.

When calculating the returns of the entire portfolio, one question that occurs is whether the portfolio returns should be averaged over all the pairs selected for trading, or simply over the pairs that actually opened positions during the trading period. Gatev et al. [17] coined the former approach the return on committed capital (RCC), and the latter fully invested return (FII). This work considers the rst approach. In spite of being more conservative, it accounts for the opportunity cost of hedge funds for having to commit capital to a strategy even if the strategy does not trade.

An additional subtlety should be noted. As Dunis et al. [7] claim, in a realistic scenario, the nancial institution lending shares would pay an interest on the collateral. Therefore this interest should be summed to the strategy's net pro t. However, we neglected this for the ease of calculation since it would represent only a very small portion of the net pro t.

The returns, as presented, are calculated on a leveraged position, as explained in section 5.5.1. In the unleveraged case, the initial capital corresponds to the initial gross exposure (value of the long position plus the value of the short position), and hence the returns would be slightly reduced.

5.6.2 Sharpe Ratio

Sharpe ratio is a risk-adjusted evaluation of return on investment rst developed by Nobel laureate William F. Sharpe [56]. It aims at measuring the excess return per unit of deviation in an investment, typically referred to as risk. The annual Sharpe ratio is calculated as

$$SR_{year} = \frac{R^{port} \quad R_f}{\quad {}_{port}} \quad \text{annualization factor:} \tag{5.5}$$

We proceed to describe each of the elements composing (5.5). First, R^{port} represents the expected daily portfolio returns. This may be calculated as the mean value of the portfolio returns, given by

$$R_t^{port} = \sum_{i=1}^{N} !_i R_t^i; \tag{5.6}$$

where R_t^i and $!_i$ represent the daily returns and the weight of the i-th pair in the portfolio of size N, respectively[9]. The risk-free rate, R_f, represents the expected rate of return of a hypothetical investment with no risk of nancial loss. This is subtracted from returns earned by a given strategy to account for the interest that same cash amount could be generating instead, with no risk. In this work, we follow the common practice of setting the risk-free rate equal to the interest paid on the 3-month US government treasury bill[10]. Table 5.5 illustrates the risk-free annualized rates considered for each period tested. The

[9] For an equal weighted portfolio, $!_i = \frac{1}{N}$.

[10] The risk-free rate is a theoretical number since technically all investments carry some form of risk. Although the government can default on its securities, the probability of this happening is very low.

values are obtained by averaging the 3-Month Treasury bill rate during the corresponding period. The data is collected from [57].

Table 5.5: Risk-free rates considered per test period.

The values presented in Table 5.5 must be converted to the expected daily returns, for coherence with (5.5). The term $_{port}$ represents the volatility of the portfolio. This depends not only on the standard deviation of each security but also on the correlation among them, and is calculated as

$$_{port} = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} !_i \, cov(i; j) \, !_j} = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} !_i \; _{i;j} \, !_j} : \tag{5.7}$$

Finally, the annualization factor enables expressing the estimated daily Sharpe ratio in annual terms. Expressing the annualized Sharpe ratio is convenient as this is a common practice and is consistent with the duration of the test periods. A common approximation to calculate the annual Sharpe ratio from the Sharpe ratio daily estimate consists of multiplying it by $\sqrt{252}$, where 252 is the number of trading days in a year. However, as pointed out by Lo [58], this is possible only under very special circumstances. Depending on the serial correlation of the portfolio returns, this approximation can yield Sharpe ratio values that are considerably smaller (in the case of positive serial correlation) or larger (in the case of negative serial correlation). The only case in which the approximation is truly valid is if the portfolio returns are independently and identically distributed (IID). This is a restrictive and often violated condition by nancial data. Thus, for a more rigorous approximation, we adopt the correction factor proposed in [58]. In this case, the portfolio returns serial correlation is measured, and a scale factor is applied in accordance. The scale factors are described in Appendix B.

One disadvantage of SR is that the volatility measurement penalizes all volatility the same, regardless of if its upside (big positive returns) or downside volatility. Yet, large losses and large gains are not equally undesirable. A common alternative is the Sortino ratio, which is calculated just like the Sharpe ratio, with the exception that the volatility is calculated by taking only the standard deviation of the negative returns. Thus, the reader might be questioning why Sharpe ratio is adopted in this work instead of the Sortino ratio. The motivation is two-fold. First, Sharpe ratio is widely used in the Pairs Trading literature. It is the metric of choice in similar research work [16, 17, 34] which facilitates the comparison of the results. Secondly, the Sharpe ratio is independent of leverage. To justify this claim, we may quickly infer that leveraging up the committed capital n times would result in multiplying both the numerator and the denominator in equation (5.5) by n, meaning result remains unchanged[11]. This property is particularly convenient, given that the constructed portfolio is leveraged.

[11] This approximation neglects the cost of funding leverage, which is considered a minor factor.

5.6.3 Maximum Drawdown

While the Sharpe ratio adjusts the returns for the risk taken, the Maximum Drawdown (MDD) indicates the maximum observed fall from a peak to a trough of a time series, before a new peak is attained. In this work, the Maximum Drawdown is calculated with respect to the account balance during the trading period. More formally, if we de ne $X(t)$, with $t \quad 0$ as representing the total account balance, the MDD is calculated as

$$\mathrm{MDD}(T) = \max_{2(0;T)} \max_{t2(0;\ )} \frac{X(t) \quad X(\ )}{X(t)} \quad : \tag{5.8}$$

5.7 Implementation environment

The code developed in this work is implemented from scratch using Python. The motivation for doing so is the vast amount of available resources that facilitate the implementation of the chosen algorithms, namely data mining procedures and data analysis. Furthermore, Python is, at the time of writing, the language of choice for Machine Learning related projects, which makes it more suitable from a collaboration point of view as well. The code is publicly available in [59].

Some libraries are particularly useful in this work. First, sci-kit learn proves helpful in the implementation of PCA and the OPTICS algorithm. Second, statsmodels provides an already implemented version of the ADF test, useful for testing cointegration. Lastly, we make use of Keras, a Deep Learning library that provides the building blocks of a Neural Network. This way, we do not need to concern with the lower-level details, since most of the features described in chapter 4 are already implemented.

Most of the simulation is run on a regular CPU (Intel Core i7 @ 3GHz). The exception is the training process of the Deep Learning models. These models involve a tremendous amount of matrix multiplications which result in long processing times when using the CPU. Fortunately, these operations can be massively sped up by taking advantage of the parallelization capabilities of a GPU. In this work, a GPU provided by Google Collaboratory (NVIDIA Tesla T4) is used to train the proposed architectures.

5.8 Conclusion

This chapter began with an overview of how the research experiment as a whole is conducted. We proceeded with a detailed description of the proposed dataset. We then explored the research environment suggested for each research stage in separate, describing the respective pairs selection conditions, trading setup, and test portfolios. Subsequently, we presented some important considerations from a simulation point of view. Namely, the way a portfolio is constructed, how transaction costs are accounted for, and the detail with which entry and exit points are de ned. In addition, the most relevant metrics monitored in this work were also introduced. At last, we described some relevant aspects concerning the implementation environment.

Chapter 6

Results

At this point, we had the opportunity to explore the workings of Pairs Trading and to present two new methodologies to strengthen this strategy. Furthermore, in the previous chapter, we delineated how the simulations should be conducted. This chapter compiles the results obtained. It starts by illustrating the dataset preprocessing outcome. It proceeds to describe the results achieved by the application of the proposed pairs selection framework (Research Stage 1) in section 6.2. Lastly, in section 6.3, the results concerning the new proposed forecasting-based trading model are analyzed (Research Stage 2).

6.1 Data cleaning

As a reminder, the initial set of securities considered in this work includes 208 different ETFs. Naturally, not all ETFs trade throughout every period considered. Furthermore, some violate the required conditions de ned in section 5.2.3.

To illustrate the outcome of processing the original dataset, Table 6.1 describes the effect of each stage in the process. The rst column represents the number of ETFs discarded due to missing values. This may be either because the ETF is not trading during the entire period or because there are sporadic missing values in the price series. The second column represents the number of ETFs, from the ones remaining, that do not verify the liquidity constraint imposed. The third column lists the number of ETFs containing outliers. On the left, the number of ETFs containing a single outlier in the price series is represented, in which case the value is corrected. On the right side, the number of ETFs for which the number of outliers is larger than one is detailed, in which case the ETF is discarded. Finally, the rightmost column indicates the percentage of initial ETFs analyzed to form pairs.

Table 6.1: Data preprocessing results.

6.2 Pairs selection performance

This section focuses on analyzing the results addressing the rst research question: “Can Unsupervised Learning nd more promising pairs?”. The periods under analysis correspond to those assigned to research stage 1 in Table 5.2.

6.2.1 Eligible pairs

We start by presenting the number of pairs selected by the system under the three different clustering methodologies. These values are represented in Table 6.2.

Table 6.2: Pairs selected for different clustering approaches.

As expected, the single cluster approach has a broader set of ETFs from which to select the pairs. Naturally, more pairs are selected using this setting. What is still left to asses is whether the pairs selected are all equally pro table.

We may observe that grouping the ETFs in ve categories (according to the categories described in section 5.2.2) results in a reduction in the number of possible pair combinations. This is not more evident due to the underlying unbalance across the categories considered. Since energy-linked ETFs represent nearly half of all ETFs, the combinations within this sector are still vast.

At last, the number of possible pair combinations when using OPTICS is remarkably lower. The results are obtained using ve principal components to describe the reduced data, as it will be explained in sec-

tion 6.2.3. Also, the only OPTICS parameter to be de ned, minP ts , is set to three, since we empirically verify this value provides a reasonable distribution of clusters. Although the number of clusters is higher than when grouping by category, their smaller size results in fewer combinations. This does not just reduce the computational burden of performing multiple statistical tests but also reduces the probability of nding bogus pairs. The OPTICS reliance on the data is evidenced by the variation of the number of clusters across different periods.

6.2.2 Pairs selection rules

In this section, we intend to dig deeper into the details of the pairs selection procedure. The proposed selection process comprehends four criteria, as described in section 3.5. There are essentially two aspects we would like to address in more detail. First, we want to examine what portion of the pairs passed the cointegration test but missed the spread Hurst exponent criterion. This result can be used as a proxy to estimate the real impact caused by the multiple comparisons problem and the in uence of the Hurst exponent in handling this situation. Secondly, we are also interested in verifying the portion of identi ed mean-reverting pairs that do not comply with the two remaining imposed conditions, to gauge the importance of establishing those constraints. This is accomplished by illustrating the number of pairs ltered out by each condition. Focusing on the scenario in which no clustering is performed seems more appropriate as it results in more possible pair combinations and is, therefore, more representative.

Table 6.3 illustrates the pairs removed at each step of the selection procedure. It is important to note that the steps are performed in a sequential way. This means that each row represents the pairs eliminated from the subset resulting from the previous selection condition. Hence, it is expected that the majority of pairs are eliminated initially. Indeed, this is what we observe. Moreover, we can con rm the role of the Hurst exponent in eliminating pairs which passed the cointegration test, but their spread did not report a Hurst exponent lower than 0.5, thus not describing a mean-reverting process.

Table 6.3: Pairs selection progress on an unrestricted pairs search environment.

It can also be inferred from the results in Table 6.3 that some pairs are not eligible because their convergence duration is not compatible with the trading period, thus not verifying the half-life condition. Nevertheless, it is evident that the mean-crossing criterion is met for the entire subset of pairs that veri ed the previous conditions, which indicates that enforcing this last rule is redundant in this scenario.

In conclusion, we may say there is a visible in uence of the Hurst exponent and the half-life constraints in ltering out pairs that would not have been discarded if simply cointegration had been enforced.

6.2.3 OPTICS application results

In this section, we intend to highlight some results from the application of the proposed technique to search for securities, based on the OPTICS algorithm. Namely, we present the outcome from varying the number of principal components used, and the composition of the clusters found. We also illustrate the application of DBSCAN in place of OPTICS, for comparison purposes.

Number of principal components

The rst stage in the application of the proposed approach for clustering ETFs consists of de ning an appropriate number of principal components to reduce the original data dimension. This procedure must achieve a balance between two desirable but incompatible features. On the one hand, more accurate representations may be obtained when the number of principal components increases, since the transformed data explains more of the original variance. On the other hand, increasing the number of components brings along more components expressing random uctuation of prices, which should be neglected. In addition, it also contributes to the curse of dimensionality, as described in section 3.3.

Table 6.4 illustrates how the clusters and the pairs found vary as the number of principal components changes. The numbers described are obtained in the period ranging from January 2014 to December 2018. This period is chosen for illustration arbitrarily since the behaviour is found to be consistent among the three periods used in this research stage. The last row emphasizes the pairs which contain constituents from distinct categories, thus emphasizing part of the diversity provided by this method. It is clear from the results disclosed in Table 6.4 that the impact of varying the number of principal components within the experimented interval is minimal. The demonstrated consistency regarding the number of clusters and their average size for a different number of principal components is a direct consequence of the robustness provided by the OPTICS algorithm. If the number of principal components is selected within an acceptable range, the OPTICS algorithm adapts its parameters automatically to t best the data described by different dimensions. Obtaining the same result using DBSCAN is considerably harder, as the investor would have to nd the best value for the parameter " in each scenario.

In the previous paragraph, we described how the clustering varies within a range of acceptable dimensions. As it was motivated at the end of section 3.3, we considered as acceptable a number of features up to 15, to protect from the curse of dimensionality. We nd relevant to conduct an exploratory analysis concerning how the algorithm performance is affected for considerably higher dimensions. For this purpose, we simulate the same scheme but now using more principal components. The results are presented in Table 6.5. Interestingly, the algorithm is still capable of nding pairs when using 50 and even 75 dimensions, but naturally with fewer clusters formed. However, for 100 dimensions, the algorithm is

Table 6.4: Clustering statistics when varying number of principal components.

incapable of detecting any pair, as one would expect to witness eventually. It should be emphasized that we do not proceed to con rm the reliability of the identi ed pairs from a trading perspective in these conditions.

Table 6.5: Clustering statistics using large number of principal components.

To conclude, we settled the number of principal components to 5. As demonstrated above, this parameter should not affect distinctively the results obtained as long as it is within an acceptable dimension range. We adopt 5 dimensions since we nd adequate to settle the ETFs' representation in a lower dimension provided that there is no evidence favouring higher dimensions.

Clusters composition

In this section, we intend to validate the clusters formed by the OPTICS algorithm and get an insight into their composition. Illustrating the process for all the periods considered would be very extensive. Because the process is common across all the periods, we decide to proceed with the analysis of the period of January 2014 to December 2017 in this section and include the information for the remaining periods in Appendix C.

To illustrate the clustering results it would be particularly appealing to show a representation of the generated clusters in the two-dimensional space. However, the data is described by ve dimensions [1]. To tackle this issue, we propose the application of the T-distributed Stochastic Neighbor Embedding algorithm (t-SNE) [60]. Brie y, t-SNE can be described as a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two (or three) dimensions. The algorithm models each high-dimensional object using two (or three) dimensions

[1] Recall we selected ve principal components when applying PCA.

in such a way that similar objects are modelled by nearby points and dissimilar objects by distant points with high probability[2].

Using the technique just described, a two-dimensional map can be attained. Figure 6.1 illustrates the clusters formed during the formation period, in the transformed space. Each cluster is represented by a different colour. The ETFs not grouped are represented by the smaller grey circles, which were not labelled to facilitate the visualization. It should be noted that Figure 6.1 is not displaying the entire set of ETFs, as two of them are placed in distant positions, and their representation makes it harder to visualize the clusters with the desired level of granularity. The existence of such outliers, from a data distribution point of view, con rms the importance of using a clustering technique robust to such cases.

Figure 6.1: Application of t-SNE to the clusters generated by OPTICS during Jan 2014 to Dec 2016.

Although the clusters formed look plausible, based solely on the current evidence, it may be premature to commentate on their integrity. To get a better insight of the securities composing each cluster, we suggest analyzing the corresponding price series. For this purpose, we select four clusters for visualization. Figure 6.2 illustrates the logarithm of the price series of each ETF. The price series illustrated result from subtracting the mean of the original price series, to facilitate the visualization.

Starting with Figure 6.2(a), it is interesting to observe how close the series follow each other, making it hard to differentiate them. One question that emerges is whether there is a fundamental reason that

[2]For more information concerning the t-SNE algorithm the interested reader may refer to the original paper [60].

(a) Normalized prices of ETFs grouped in Cluster 1.

(b) Normalized prices of ETFs grouped in Cluster 2.

(c) Normalized prices of ETFs grouped in Cluster 3.

(d) Normalized prices of ETFs grouped in Cluster 4.

Figure 6.2: Price series composition of some clusters formed in the period from Jan 2014 to Dec 2016.

justi es this behaviour. There is. The four securities identi ed do not just belong to the same category, Precious Metals, but are also linked to Gold. This is the rst evidence that the OPTICS-based approach can go one level deeper in grouping ETFs, as it is capable of detecting not just categories, but speci c segments. A similar situation is encountered in Figure 6.2(b). In this case, the securities identi ed also belong to the same segment, the Equity US: MLPs, which is a component of the entire Energy sector.

Figure 6.2(c) and 6.2(d) demonstrate the OPTICS clustering capabilities extend beyond selecting ETFs within the same segment. In cluster 3, we may nd ETFs belonging to the Agriculture sector (CGW, FIW, PHO, and PIO), ETFs associated with Industrial Metals (LIT and REMX) and ETFs in the Energy sector (YMLI). There is a visible relation among the identi ed price series, even though they do not all belong to the same category. The same applies to cluster 4, which is composed of three different categories, Energy (KOL), Industrial Metals (PICK) and Broad Market (UCI).

To conclude, we con rm the clusters formed accomplished its intent of combining the desired traits from the two other techniques studied. Namely, a tendency to group subsets of ETFs from the same category while not impeding clusters containing ETFs from different categories.

Comparison with DBSCAN

As attentively motivated at the end of section 3.4.3, the OPTICS algorithm is particularly advantageous in comparison with DBSCAN for two reasons. It handles well clusters of varying density within the same dataset, and it avoids the cumbersome process the investor would have to go through to nd

the appropriate value for ". To illustrate how this is evident in practice, we proceed to represent the clusters that would have been formed using DBSCAN, during the same formation period, for a varying parameterization of " in Figure 6.3.

(a) DBSCAN with " = 0 :1. (b) DBSCAN with " = 0 :15. (c) DBSCAN with " = 0 :2.

Figure 6.3: Applying DBSCAN with varying " to the formation period from Jan 2014 to Dec 2016.

From the observation of the three scenarios in Figure 6.3, the rst aspect that stands out is the high sensibility of the algorithm to variations in the parameter ". This aspect reinforces that the investor must be very cautious when choosing the parameterization. Furthermore, from the three con gurations presented, how should the investor select " ? It might not be straightforward to decide which combination yields the most promising results and leaving this decision to the investor might be too demanding. The OPTICS algorithm, however, allows the automation of this process.

6.2.4 Trading performance

In this section, we analyze how the selected pairs under the three distinct clustering conditions perform from a trading point of view. This enables validating which clustering technique generates the most promising portfolio.

We start by revealing the results obtained with the threshold-based trading model, as proposed in section 5.3.2, during the validation periods included in Table 6.6. The results presented in this table should be analyzed with caution because there is an implicit bias arising from the fact that the strategy is being examined in the same period used to identify the pairs being tested, which does not resemble a real trading scenario. Hence, the results are not truly representative of the real performance. If on the one hand, one might expect the results to be more satisfactory, due to the extra con dence on the robustness of the pairs being used, on the other hand, they may decline, since we cannot eliminate pairs that have not been pro table in the past, as there are no available records yet. Nevertheless, the validation records are relevant to support the construction of the test portfolios, which use these results as a heuristic for selecting pairs. For this reason, we decide to present them. One key idea to retain from Table 6.6 is that the strategy is pro table in every scenario, with a satisfactory margin. Therefore we can con rm the success of the pairs selection process in selecting pro table pairs up to this point.

Table 6.7 unveils the results concerning the unseen data, the test periods. In this case, for each test

Table 6.6: Validation results for each pairs grouping technique.

period, we describe the results when adopting any of the three implemented test portfolios, described in section 5.3.3. As a quick reminder, the portfolio identi ed by number 1 comprehends the entire set of pairs identi ed in the formation period, portfolio number 2 contains only the pairs that had a satisfactory performance in the validation period, and portfolio 3 comprises the ten pairs which achieved the best performance in the validation period. To aggregate the information in a more concise way, the average over all years and portfolios is described on the rightmost column. Note also that the three evaluation metrics described in section 5.6 are accentuated, to differentiate from the remaining, less critical, descriptive statistics.

We can con rm the persisting pro tability in any of the environments tested. This result further corroborates the idea that the pairs selection procedure is robust. Besides, we may assert that if an investor is focused on obtaining the highest possible ROI, regardless of the incurred risk, performing no clustering is particularly appealing. Except for 2015, when the OPTICS based strategy performed slightly better in this sense, the no clustering approach obtains higher ROI overall. However, if the investor is also concerned with the risk its portfolio is exposed to, the remainder indicators suggest that clustering using OPTICS may achieve a better trade-off. We proceed to analyze why.

The OPTICS-based approach is capable of generating the highest average portfolio Sharpe ratio of 3.79, in comparison with 3.58 obtained when performing no clustering or 2.59 when grouping by category. It also shows more consistency with respect to the portion of pro table pairs in the portfolio, with an average of 86% pro table pairs, against 80% when grouping by category and 79% when performing no clustering at all. It is interesting to observe that the only circumstances under which 100% of the pairs in the portfolio turned out to be pro table correspond to the situation that makes use of the OPTICS algorithm. This happened during two different periods, 2015 and 2017, when using portfolio 3. This

Table 6.7: Test results for each pairs grouping technique.

stresses the consistency provided by this clustering approach.

Concerning the maximum drawdowns, we notice that this metric evidences some instability when grouping by category or not grouping at all. The results indicate that when these strategies use the ten best-performing pairs from the validation period (portfolio 3), the portfolio maximum drawdown amplitude tends to increase markedly. This is especially evident in 2017, in which case the MDD has nearly the same amplitude as the ROI. Contrarily, the OPTICS based strategy proves capable of maintaining a favourable MDD in every period, with no irregularities. This is con rmed by the lower average MDD, which emphasizes the robustness provided by this technique.

Lastly, we can verify that since the OPTICS algorithm only tends to group pairs that seem more sound, the system is less likely to identify pairs that achieve a very good validation performance and see it deteriorate in the test period. This aspect may be validated from the analysis of the Sharpe ratio progression from the validation to the test results. While on the validation set (Table 6.6) the OPTICS based approach is only able to outperform the other strategies in 2015, in the test set it surpasses the remaining

techniques most of the times.

Curiously, regarding the two standard grouping techniques, we observe that grouping the clusters by category yields poorer results than those obtained when not grouping at all. This suggests that, in the absence of OPTICS, the investor might be better off not performing any grouping, instead of limiting pairs to be formed by securities from the same sector.

On a different note, the results indicate that there is no indisputable best way of constructing the test portfolio. Evaluating by the Sharpe ratio, the portfolio with the best performance depends on the period and the clustering technique being used. As every portfolio outperforms the other two at least once, the results are inconclusive. However, evaluating in terms of ROI, we may infer that using information from the validation performance for constructing the test portfolio is a useful heuristic after all, as portfolio 2 and 3 outperform portfolio 1 in every case.

6.3 Forecasting-based trading model performance

This section focus on analyzing the results that let us answer the second research question: “Can a forecasting-based trading model achieve a more robust performance?”. To accomplish this, we evaluate how the forecasting-based model, introduced in chapter 4, performs when compared with the threshold-based model, simulated so far. In doing so, we gradually describe how the simulation progresses, from the pairs identi cation until the trading itself.

6.3.1 Eligible pairs

We proceed to describe the selected pairs during this research stage. In this case, only the OPTICS clustering is applied to cluster the candidate pairs. This approach is chosen due to its already demonstrated ability to form robust portfolios. The clustering is then followed by the regular pairs selection procedure, just as before. Table 6.8 presents some relevant statistics regarding the pairs found for both periods used in this second research stage.

Table 6.8: Pairs selected during research stage 2.

Only 5 pairs are deemed eligible for trading based on the formation period from Jan 2009 to Dec 2017, considerably fewer than those found in the other periods analyzed. This is not surprising since there are just 58 active ETFs throughout this period, as detailed in Table 5.2. In addition, the portion of cointegrated ETFs during such a long period must be smaller. Nonetheless, this does not represent a

threat since selecting fewer pairs can be desirable in this setting. Because training the Deep Learning forecasting models is computationally very expensive, the computation time can be greatly reduced by considering just a small number of pairs. This also provides a good opportunity to take a closer look at the identi ed ETFs, one by one. Table 6.9 lists the selected pairs, along with the corresponding segment, and their description. Both the segment and the description information are retrieved from [52].

Table 6.9: Candidate pairs identi ed from Jan 2009 to Dec 2017.

It is interesting to notice that two out of ve pairs belong to different categories. To get a better insight into the underlying motive, we analyze the DBA-GCC pair in more detail. As GCC is a Broad Market type of ETF, meaning that it follows the commodity market as a whole, we gather more information from [61] regarding its sector decomposition, described in Table 6.10.

Table 6.10: GCC Sector Breakdown.

By analyzing Table 6.10, one can observe that the major component driving GCC is Agriculture. Thus, an equilibrium between GCC and DBA can be interpreted as a consequence of major exposure to the

same sector. The deviations from the equilibrium can be potentially justi ed by market changes in the remaining sectors.

Revisiting Table 6.8, now focusing on the period of January 2015 to December 2018, we nd 19 pairs eligible for trading. This number is consistent with the values found in the previous periods of the same duration. These pairs are used to gauge the performance of the benchmark trading model, applied in its standard interval range (4 years, as applied in section 6.2).

6.3.2 Forecasting algorithms training

The forecasting algorithms are trained to t the spreads from the ve pairs selected during the formation period of January 2009 to December 2017, represented in Figure 6.4. It is interesting to observe that the spreads look stationary, as expected. Moreover, there is a noticeable difference in the volatility among the time series, which further supports the idea that data-driven model parameters, as the proposed thresholds for the forecasting-based trading model, seem appropriate in these conditions.

Figure 6.4: Spread from the ve pairs identi ed between Jan 2009 and Dec 2018.

Having described the training data, we advance with the forecasting architectures experimented. The architectures are made more complex until there is no more evidence of improvement. A total of 31 forecasting model architectures are implemented, meaning a total of 155 models are trained (31 architectures 5 spreads). From the 31 implemented architectures, 9 correspond to the con gurations experimented with the ARMA model, 8 with the LSTM based model and 14 with the LSTM Encoder-Decoder. The implemented con gurations are described in Appendix E. The cost function that each forecasting algorithm aims to optimize is the Mean Squared Error (MSE). To see the details regarding MSE as well as other forecast measures used in this work, please refer to Appendix D.

It is worth adding that although both the parametric and the non-parametric approaches use the for-

mation period for training, the methodology slightly differs for each case. While the ARMA model uses the entire formation period to learn the combination of coef cients that minimizes the forecasting error, the LSTM, and LSTM Encoder-Decoder use only the training period for that purpose. In this last case, the validation set is considered separately to control over tting. Through early-stopping, as described in section 4.7.3, the system continuously monitors the validation score and stops the training process if signs of over tting are detected. Figure 6.5 illustrates one typical training scenario with an example from the LSTM Encoder-Decoder tting the spread resulting from the pair DGP-UGL.

(a) Mean Squared Error during training.

(b) Mean Absolute Error during training.

Figure 6.5: Encoder-Decoder training loss history for DGP-UGL spread.

On Figure 6.5(a), we may observe how the training and validation MSE evolve. As expected, they both decrease as the training progresses. But the algorithm seems to reach its optimal con guration at epoch 41. From that point on, there is no improvement in the validation MSE, despite the still decreasing training MSE. This suggests the algorithm is over tting. The gure represents a total of 91 epochs (although not visible) because the early-stopping is con gured with a patience of 50 epochs. Thus, the training continues for 50 epochs and stops at the 91st, since no better score is obtained in the meanwhile. It is also worth observing the image on the right, 6.5(b), which depicts the evolution of the Mean Absolute Error (MAE). Although the model is con gured to minimize the MSE, it is visible that the MAE progression displays a similar pattern.

6.3.3 Forecasting performance

Having described the selected pairs and how the training is conducted for each forecasting approach, we may proceed with the analysis of the forecasting results obtained. An extensive list of results concerning the forecasting score of all the implemented architectures is presented in Appendix E. In this section, we focus on comparing the results from the best performing architectures.

It is important to clarify that we consider the best architectures as those which minimize the validation mean squared error. The reason for using the validation score as a reference is that it resembles the information available to the investor prior to start trading, the point at which this decision must be taken. The mean squared error is the targeted metric used for comparison purposes as this is the loss

function being minimized by the models. According to these criteria, the best performing architectures are presented in Table 6.11. Also, the results described for each model correspond to the average forecasting performance across the ve spreads' forecasts.

Table 6.11: Forecasting results comparison.

Inspecting the results from Table 6.11, we may validate that all the implemented models are capable of outperforming the naive implementation (de ned in section 5.4.1) during the validation period. Curiously, we note that the LSTM-based models do not manage to surpass the ARMA model, at least with respect to the chosen metrics. Also, the results obtained in the test set show signs of over tting, besides the efforts taken in that regard, as the LSTM-based models are no longer superior to the naive model.

These results naturally make us question the suitability of LSTMs under this work's conditions, which leads us to dig deeper into the literature. Although the tness of different forecasting algorithms strongly depends on the dataset, the literature evidence that there is no consensus concerning the suitability of LSTMs for time series forecasting.

On the one hand, some research work supports the dominance of RNNs over simpler techniques such as multi-layer perceptrons (MLP) or auto-regressive models. For instance, [62, 63] allege these networks are generally better than MLP networks, although they do suffer from long computational times. Also, [64] investigates whether LSTMs are superior to the traditional algorithms, such as the ARIMA model, and the empirical studies conducted con rm the hypothesis. These type of considerations motivated the application of LSTM-based models in this work in the rst place.

On the other hand, some authors as Gers et al. [65] claim that common time series problems found in the literature are often conceptually simpler than many tasks already solved by LSTMs, and that more often than not, they do not require RNNs at all, because a few recent events convey all relevant information about the next event. More speci cally, the authors nd evidence that the time window based MLP outperforms the LSTM approach on certain time series prediction benchmarks solvable by looking at a few recent inputs only, thus concluding that LSTM's special capabilities are not necessary for such circumstances. Our results are aligned with this point of view. Although an MLP model is not implemented, the results clearly indicate that under this problem conditions, the traditional ARMA

model proves more robust with respect to the metrics identi ed in Table 6.11. This suggests the extra complexity provided by the LSTM-based models might be unnecessary. In the next section, we intend to analyze how this impacts trading pro tability.

6.3.4 Trading performance

We start by recalling that the primary motivation for pursuing the forecasting-based pairs trading approach is to make the portfolio more robust, by reducing the number of days the portfolio sees its value decrease. To evaluate the effectiveness of the proposed methodology, we proceed to describe the results obtained by the forecasting-based trading model using the different forecasting algorithms, during the period of January 2009 to December 2018. The results, in Table 6.12, are set side by side with the results obtained using the standard threshold-based model. It is important to remind once more that the results in the validation set are presented for completeness. Its real duty is to extract the pairs that prove capable of generating pro ts in the validation period, rather than testing the strategy itself.

Table 6.12: Trading results comparison using a 9-year-long formation period.

Focusing on the test results, which resemble a practical trading environment, we can validate that the period of portfolio decline is considerably shorter when using the forecasting-based models, in comparison with the standard trading model. The longer period corresponds to the LSTM Encoder-Decoder, of 21 days, that is still less than 25% of the standard trading model's total decline period, of 87 days. However, that is not all there is to it, as this comes at the expense of a worse performance in terms of total return on investment and portfolio Sharpe ratio.

We suspect the long required formation period is also responsible for the visible pro tability decline. Therefore we proceed to analyze the standard trading model in a regular 3-year-long period, this time

focused on the pro tability indicators, to have a baseline for comparison. The results are displayed in Table 6.13.

Table 6.13: Trading results for standard trading model using a 3-year-long formation period.

By comparison, the performance in the 10-year-long period seems greatly affected by the long required duration, suggesting the less satisfactory returns emerge not merely as a consequence of the trading model itself, but also due to the underlying time settings. Following this line of reasoning, if the forecasting-based model performance increases in the same proportion as the standard trading model when reducing the formation period, the results obtained could be much more satisfactory. The formation period may be reduced, for example, by using higher frequency data, thus maintaining the same level of data points.

On a different note, as we suspected from the forecasting results presented in section 6.3.3, the LSTM-based models do not manage to surpass the ARMA-based trading model. Yet, the fact that the LSTM Encoder-Decoder clearly exceeds the simpler LSTM model indicates there is some potential to using multi-step forecasting in this setting. Thus, it is left for future research to evaluate how an adapted multi-step prediction using ARMA would have performed.

6.3.5 Implementation remarks

From a practical point of view, it is relevant to describe the training duration associated with tting the forecasting algorithms. Naturally, the duration increases with the complexity of the architectures experimented. In our analysis, we focus on the models which obtained the best forecasting score. Table 6.14 introduces the training duration for two different test environments. In the rst one, a regular CPU is used. In the second case, the K80 GPU provided by Google Collaboratory is exploited. In this case, the