LUISS

Corso di laurea in Data Science and Management

Cattedra di Machine Learning

Machine Learning in Finance: Developing a Pairs Trading Investment Strategy with Python

Prof. Giuseppe F. Italiano

RELATORE

Prof. Antonio Simeone

CORRELATORE

Lorenzo Meloncelli

CANDIDATO

Anno Accademico 2022/2023

Summary

1. Introduction

The explosive growth of big data has transformed the finance industry by creating new opportunities to extract value from financial information. For global banking, McKinsey & Co. analysts estimate that AI technologies could potentially deliver up to $1 trillion of additional value each year. The use of machine learning techniques in finance has in fact shown promise in improving investment decisions, maximizing returns while minimizing risk, and developing more accurate trading strategies. Despite this, the use of such techniques in pairs trading is still scarce.

This thesis aims to contribute to the literature around the interaction between machine learning and trading by replicating the paper “A Machine Learning Based Pairs Trading Investment Strategy” (Sarmento & Horta, 2020), and by exploring modifications that validate and extend its results. The paper presents a novel approach that involves using machine learning techniques to select pairs of assets to trade and a trading algorithm that uses a neural network to predict the price movements of the pairs.

The thesis will follow the same structure as the paper, which is divided into two research stages: the pairs selection stage and the trading stage. Research Stage 1, the pairs selection stage, consists of two steps: (i) finding the appropriate candidate pairs and (ii) selecting the most promising ones. Here machine learning will be used in the form of unsupervised learning to identify pairs of assets suitable for pairs trading. In Research Stage 2, the trading stage, we will develop and test a trading algorithm that leverages supervised learning algorithms to predict the price movements of the selected pairs and generate buy/sell signals. To have a benchmark, the results obtained from the algorithms will be compared with those obtained with more traditional statistical techniques.
Each step necessary to complete these two research phases will be explained first from a theoretical point of view and then from a practical point of view. All the codes developed for this thesis are fully reproducible and made publicly available on the author's GitHub page.

To the best of my knowledge, this thesis is one of the first to replicate and extend the original paper's approach. Firstly, we will increase the frequency of the data from 5-min to 1-min to further reduce the required formation period and consequently find more pairs. In doing so, we can also examine one of the authors propositions, which posits that shortening the training duration could enhance the predictive accuracy of the forecast-driven model. Secondly, we will combine commodity-linked ETFs and currency-linked ETFs to increase the chances of finding profitable pairs. Finally, we will experiment training the Artificial Neural Networks in a classification setting. While regression seems like the obvious choice when predicting real-valued outputs, it presents challenges in terms of optimization and robustness. The MSE loss, commonly used in regression, is more difficult to optimize and prone to disruptions from outliers, resulting in large gradients. On the other hand, classification, utilizing a stable loss function like Softmax, provides a

more efficient optimization process and improved robustness. Additionally, classification offers the benefit of not only yielding a single regression output but also providing a distribution that signifies the confidence associated with the predictions.

Overall, this thesis aims to provide a comprehensive overview of the approach, including its theoretical and practical aspects, with a particular focus on the machine learning techniques used. The thesis and accompanying codes will be valuable resources for academics and practitioners interested in the application of machine learning in finance, and it is hoped that the results of this thesis will contribute to the growing body of literature around the interaction between machine learning and trading.

1.1 Overview

Since this thesis has as its object the meeting of two worlds which, although superimposable, are intrinsically different, it might be useful to briefly describe them both. In this way, those who are not familiar with one of the two can get an idea directly here without having to first consult other sources.

1.1.1 What is Pairs trading?

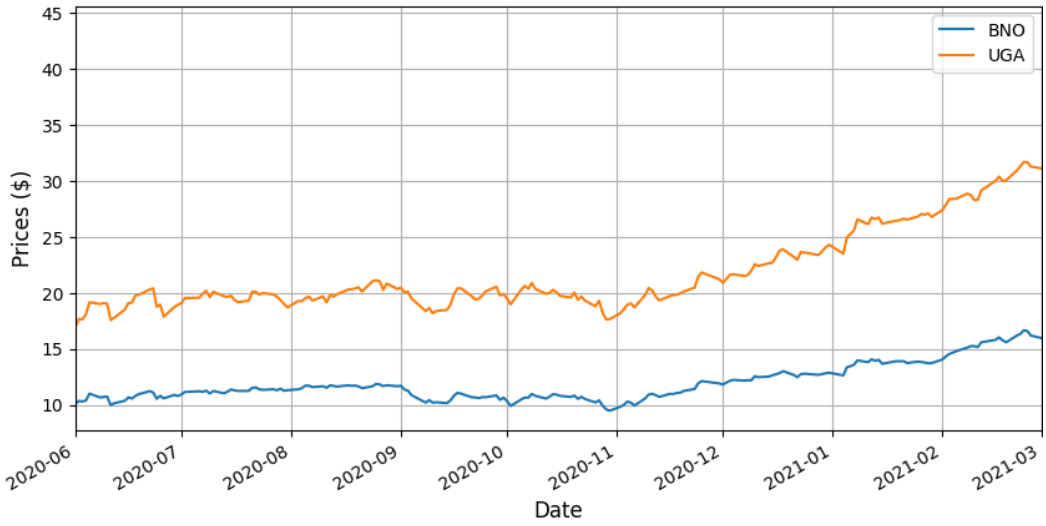
Pairs trading is a market-neutral trading strategy that involves the simultaneous buying and selling of two highly related securities to profit from the price spread between them (Gatev et al., 2006). The strategy is based on the concept of mean reversion, which suggests that, over time, the prices of two related securities will tend to move towards their long-term average relationship (Chan, 2013). The goal is therefore to identify two stocks that have a stable relationship and initiate positions to profit from deviations from this relationship.

The relationship between two securities can be established by various methods, such as cointegration. As we will see later, cointegration is a statistical technique that involves regressing the prices of two securities on each other and testing for the presence of a long-term relationship. Other methods for establishing the relationship between two securities include fundamental analysis and technical analysis. Fundamental analysis involves examining the underlying financial and economic factors that affect the securities, such as earnings, revenue, and market share. Technical analysis involves studying price charts and identifying patterns and trends that can be used to predict future price movements.

Regardless of the chosen strategy, once a stable relationship is established, traders look for deviations from this relationship, which can be caused by various factors such as news events, changes in market sentiment, or fundamental shifts. Traders then initiate a long position in the underpriced security and a short position in the overpriced security, expecting the spread between the two securities to eventually revert to its mean value (Chan, 2013).

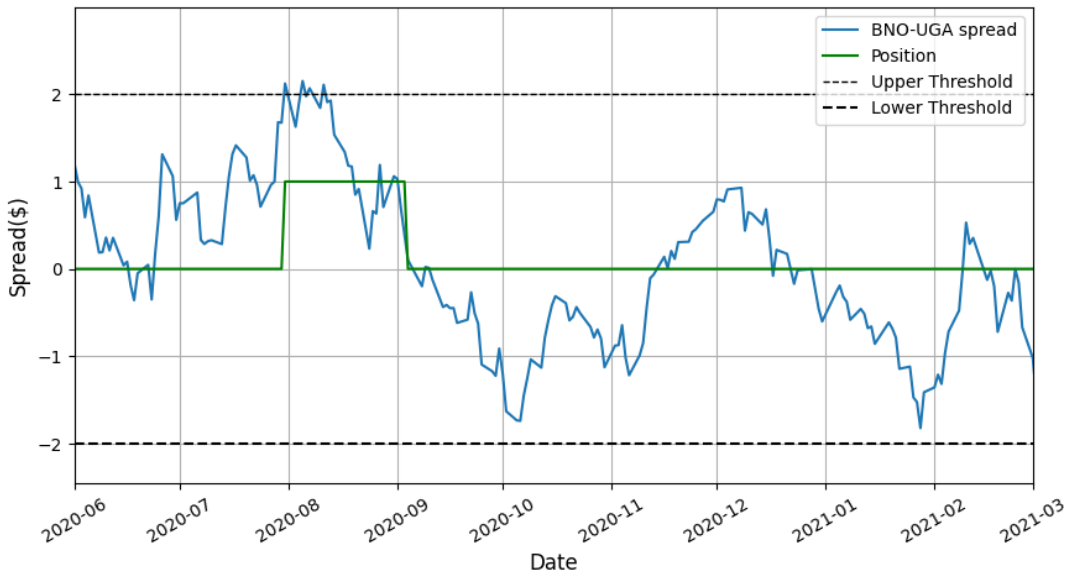
Figures 1.1 and 1.2 show a practical example. A trader has identified two stocks, BNO and UGA, that have a stable relationship.

Figure 1.1: Price series that have the potential to constitute profitable pairs



The trader has also found that historically, the price of stock BNO tends to be 1.5 times the price of stock UGA. However, due to a recent news event that negatively impacts stock UGA, its price has dropped significantly relative to stock BNO, causing the price spread between the two to widen. The trader believes that this deviation is temporary and that the price spread will eventually revert to its long-term average relationship. To profit from this mispricing, the trader initiates a long position in stock UGA and a short position in stock BNO.

Figure 1.2: Providing an example of the execution of a pairs trading strategy



The trader profits from the price spread between the two stocks as it narrows towards its long-term average relationship, regardless of whether the broader market is going up or down.

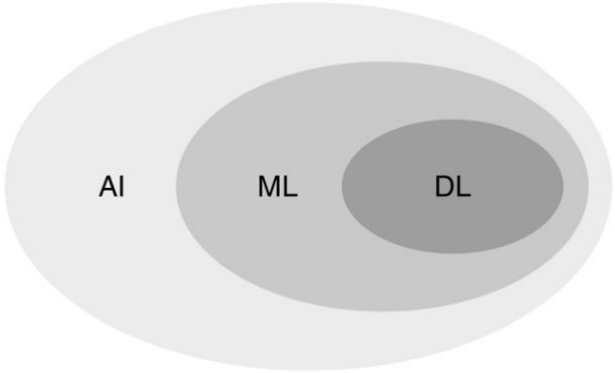
As the price spread narrows, the trader can close the positions and lock in the profit. However, if the spread widens significantly, the trader may need to adjust the positions or close them to limit potential losses.

In summary, pairs trading is a market-neutral strategy that aims to profit from the price spread between two closely related securities. This strategy has been successfully applied in various financial markets and has been found to be profitable, as shown by studies such as Gatev et al. (2006).

1.1.2 What is Machine Learning?

To facilitate understanding, the well-known scheme shown in Figure 1.3 will be followed.

Figure 1.3: The difference between AI, ML and DL



Artificial intelligence (AI) refers to the development of computer systems capable of performing tasks that typically require human intelligence, such as reasoning, learning, perception, and problem-solving (Russell & Norvig, 2016). AI can be classified into two categories: narrow AI, which is designed for specific tasks, and general AI, which aims to replicate human-like cognitive abilities across various domains (Bostrom, 2014).

A major subfield of AI is Machine learning (ML). ML is centered on the development of algorithms that allow computers to learn from data and make predictions or decisions without explicit programming (Samuel, 1959). ML techniques can be grouped into three categories:

1. Supervised learning: In this approach, an algorithm is trained using labeled data, where the input-output pairs are provided. The algorithm learns a mapping from input to output, enabling it to make predictions on new, unseen data. Common supervised learning algorithms include linear regression, logistic regression, support vector machines, and decision trees (Bishop, 2006).
2. Unsupervised learning: In contrast to supervised learning, unsupervised learning algorithms work with unlabeled data, identifying patterns or structures within the data. Clustering, a common unsupervised learning technique, groups similar data points together, while dimensionality reduction

techniques, such as principal component analysis, reduce the number of variables in a dataset while preserving its structure (Bishop, 2006).

3. Reinforcement learning: This paradigm involves an agent learning optimal actions by interacting with its environment, receiving feedback in the form of rewards or penalties. The agent's goal is to maximize the cumulative reward over time. Q-learning and deep Q-networks are examples of reinforcement learning algorithms (Sutton & Barto, 2018).

Deep learning (DL) is a subfield of machine learning that employs artificial neural networks (ANNs) to model complex relationships within data (LeCun, Bengio, & Hinton, 2015). ANNs consist of interconnected nodes, or neurons, organized into layers, with each layer transforming the input data into higher-level representations. The depth of these networks, comprised of multiple layers, allows for the extraction of intricate features and patterns, giving deep learning its name.
Convolutional neural networks (CNNs) are a specific type of ANN particularly well-suited for image recognition tasks, as they can identify spatial patterns and hierarchies in data (Krizhevsky, Sutskever, & Hinton, 2012). Recurrent neural networks (RNNs), another ANN variation, have been instrumental in natural language processing and time series analysis due to their ability to model sequences and retain information from previous time steps (Hochreiter & Schmidhuber, 1997). More recently, transformer models, such as BERT and GPT, have advanced the state of the art in natural language understanding and generation, thanks to their attention mechanisms and self-supervised learning approaches (Vaswani et al., 2017).

In the finance domain, machine learning, and more specifically, deep learning techniques have shown promising results in various applications. Recurrent neural networks, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures, have been employed for time-series forecasting, enabling more accurate predictions of stock prices, exchange rates, and other financial variables (Fischer & Krauss, 2018). CNNs have been utilized for sentiment analysis on financial news, extracting relevant information from text and images to inform trading strategies and risk management decisions (Ding et al., 2015). Other applications of machine learning in finance include credit scoring, fraud detection, algorithmic trading, and portfolio optimization (J.P. Morgan, 2016). By leveraging the predictive capabilities of these advanced models, financial institutions can increase efficiency, enhance risk management, and offer more personalized services to clients.

1.2 Research Question

As the interest in Pairs Trading escalates, finding profitable pairs becomes more challenging. This scarcity prompts investors to explore wider security groups in hopes of increasing the probability of identifying a

successful pair. A standard technique involves generating all potential pair combinations by comparing each security with every other security in the dataset. This results in n (n − 1) possible combinations, with n representing the number of available securities. Two issues emerge from this approach. Firstly, the computational expense of evaluating mean reversion for all combinations skyrockets as more securities enter the equation. This is particularly significant in situations where securities are continually monitored to create new pairs. Secondly, when conducting numerous hypothesis tests simultaneously, the multiple comparisons problem occurs. For instance, with 100 hypothesis tests executed at a 5% confidence level, the results will have a 5% false positive rate[1]. Considering a hypothetical scenario with 20 stocks, there would be 190 potential combinations, and around 9 results would be incorrect (5% of 190). This number is notably high since discovering genuine cointegrated pairs among randomly chosen securities is rare.

Although it's impossible to entirely eliminate the multiple comparison problem, its impact can be mitigated. Applying multiple correction tests, such as Bonferroni[2], or conducting fewer statistical tests are two options. Bonferroni is frequently employed for multiple comparisons, but it is known to be overly conservative. In this regard, Harlacher (2016) discovered that the Bonferroni correction is excessively cautious for pairs selection, hindering the identification of genuinely cointegrated combinations. Instead, the author suggests efficiently partitioning the asset universe to decrease the number of viable combinations and, consequently, the number of statistical tests. This approach might lead investors to focus on comparing securities exclusively within the same sector. This significantly lowers the required number of statistical tests, thus reducing the chances of identifying false relationships. Additionally, securities within the same sector are more likely to exhibit correlated movements due to exposure to similar underlying factors. This method is also relatively simple to implement. However, this simplicity may prove to be a drawback, as the more traders know about the pairs, the more difficult it becomes to find pairs not already being heavily traded, leaving less room for profit.

The pursuit of a solution to this disparity involves seeking a technique that achieves equilibrium between two situations: effectively dividing the range of assets beforehand to avoid confining potential pairings to obvious solutions, all while minimizing the need for an excessive quantity of search combinations. Sarmento & Horta (2020) suggests employing an unsupervised learning technique on the expectation that it will deduce significant groupings of assets for choosing pairs. The primary objective is to enable the data to reveal its inherent patterns, as opposed to assigning predefined categories to each security individually.

The methods for identifying the candidate pair space described above will be compared in terms of profitability. Once selected, the pairs will in fact be fed to a threshold-based trading model in which,

[1] In 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). Formally, the probability (α) of committing at least one type I error when performing m independent comparisons is given by $\alpha = 1 - (1 - \alpha)^{m}$.

[2] The Bonferroni correction controls the Famili-Wise Error Rate, the probability of making a Type I error among a specified group (or "family") of tests in multiple comparisons, by adjusting the significance level (α). It divides the overall significance level (α) by the number of tests (m) to obtain an adjusted significance level (α_adjusted). Each test is then performed using α_adjusted as the threshold for statistical significance.

following the classical framework proposed by Gatev et al. (2016), trades are initiated based on the spread deviation. If the spread between two constituent price series of a pair exceeds two historical standard deviations, a trade is initiated. This trade is concluded either upon a return to the mean, at the conclusion of the trading period, or in the event of a delisting.

A drawback of the threshold-based trading model is the imprecise definition of entry points. The sole criterion for entering a position is surpassing a predetermined threshold, regardless of the spread's present trajectory, which could lead to unfavorable portfolio downtrends if divergence persists. To validate the different clustering methods the threshold-based trading model is sufficient. However, it would be interesting to define a more robust strategy. To do this, Sarmento & Horta (2020) incorporates predicted future data in determining market entry points and propose the application of a forecast-driven trading approach. The model continuously track the percentage difference between the current spread and the anticipated spread in the following time-step. When the absolute value of the projected change exceeds a preset threshold, a position is initiated with the expectation of profiting from a sudden spread fluctuation. There are several models we will explore to predict the spread, both parametric (e.g., ARMA) and non-parametric (e.g., ANN). Also in this case the implemented models will be compared in terms of profitability.

2. Data preparation

Data preparation, also known as data preprocessing, is the process of collecting data and transforming it into a format suitable for analysis. It is a critical step in the machine learning pipeline, as the quality of the input data directly impacts the accuracy and reliability of the resulting models.

2.1 Data collection

In the following section we will define the three dimensions that were followed to build the dataset on which the analysis was carried out: asset, market and time frame. Once this is done, we will move on to the cleaning phase and, finally, we will get to the heart of the analysis: the research stages.

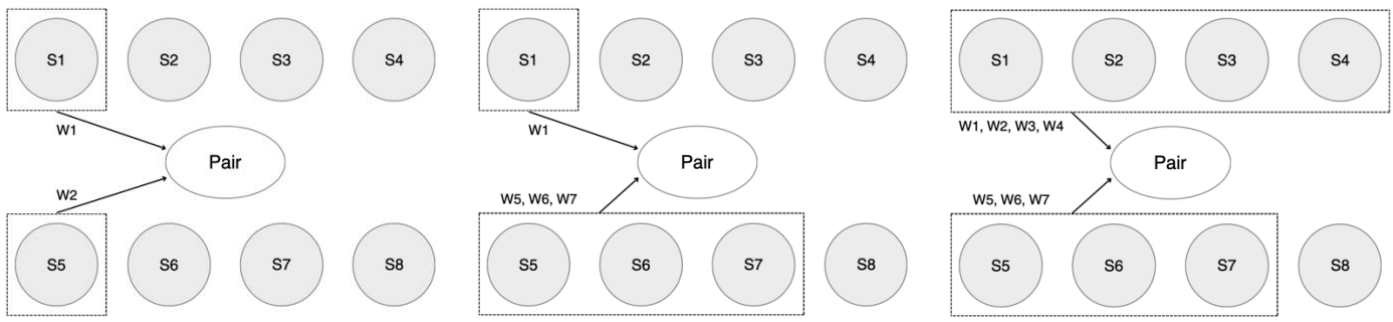
2.1.1 Asset of choice

As regards the choice of the asset to be traded, I followed the steps of the paper that I intend to replicate and choose Exchange Traded Founds (ETFs).

ETFs track an index, commodity, or collection of assets and trade like a stock. Chan (2013) highlights that trading ETF pairs, rather than stock pairs, offers increased stability when found to be cointegrated, as the fundamental economics of a basket of stocks evolves more slowly than individual stocks. This stability is particularly attractive for Pairs Trading, where the core assumption is that cointegrated security prices will continue to be cointegrated in the future.

To further corroborate the hypothesis that ETFs are useful in Pairs Trading, Chen et al. (2017) and Perlin (2017) provide significant references. However, before delving into the authors' findings, it is necessary to introduce some terminology. A univariate strategy is one in which both constituents of the pair are individual securities. On the other hand, if one side of the pair is a weighted average of comoving securities, the strategy is known as quasi-multivariate. Finally, when both sides of the spread are a weighted average of a group of securities, the strategy is multivariate, as illustrated in Figure 2.1.

Figure 2.1: Illustration of Univariate, Quasi-Multivariate, and Multivariate strategies



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

In the context of the works mentioned above, specifically Chen et al. (2017) and Perlin (2017), the aim is to create more stable mean-reverting time series by constructing each leg of the spread through a linear combination of stocks. This approach is taken to reduce the potential impact of outlier stocks on the spread. In Chen et al. (2017), a Pairs Trading strategy is employed based on Pearson correlation, and it is applied in both univariate and quasi-multivariate setups. In the univariate setup, pairs are formed by selecting the two most correlated stocks, while in the quasi-multivariate setup, a pair consists of one stock and an equally weighted portfolio comprised of the 50 most correlated stocks. The results of these studies indicate that the quasi-multivariate trading strategy demonstrates superior performance, whereas reducing the number of stocks in the weighted portfolio leads to a significant drop in returns, up to two thirds in some cases.
Perlin (2017) also assesses the efficacy of a univariate strategy versus a quasi-multivariate strategy using the 57 most liquid stocks in the Brazilian market. In the univariate scenario, pairs are created by pairing the two most correlated stocks, whereas in the quasi-multivariate scenario, one stock is paired with five partner stocks. Once again, the findings support the superiority of the quasi-multivariate strategy over the univariate approach.
However, it's important to note that both studies are influenced by the presence of high transaction costs associated with purchasing multiple securities. To address this issue, Sarmento & Horta (2020) propose an alternative approach by using ETFs (Exchange-Traded Funds) as a substitute. ETFs are essentially a weighted average of a group of securities, making them a more practical option for implementing multivariate pairs trading strategies. This is because trading ETFs can offer the benefits of multivariate strategies in a more cost-effective and convenient manner.
Additionally, the expanding universe of ETFs offers an interesting characteristic: the growing popularity of ETFs leads to the development of new, slightly different ETFs each year, which naturally create promising pairs for Pairs Trading.
Lastly, there is limited research on Pairs Trading applications within the ETF universe, making it an intriguing area for further exploration. This gap in the literature presents an opportunity to investigate the proposed approaches using ETFs in Pairs Trading.

2.1.2 Market of choice

As regards the choice of the reference market, I decided to expand the one chosen by Sarmento & Horta (2020) by adding the sector of currencies to that of commodities, with the aim of increasing the chances of finding profitable pairs. To understand why, let's start by considering the relationship between commodity prices and currencies of commodity-exporting countries. According to a research paper by Chen, Rogoff, and Rossi (2010), there is a strong link between commodity prices and the exchange rates of commodity-exporting countries, such as Australia, Canada, and New Zealand. The authors found that changes in commodity prices can predict the future direction of these countries' currencies. If commodity prices rise, the currencies of commodity-exporting countries typically strengthen because higher commodity prices

increase these countries' terms of trade. A stronger currency can then translate into higher returns for investors in currencies-linked ETFs that track these currencies. Secondly, consider the relationship between the U.S. dollar and commodities. Commodities are often priced in U.S. dollars. According to Frankel (2008), when the U.S. dollar depreciates, commodity prices usually increase. This is because when the dollar weakens, commodities become cheaper for holders of other currencies, which can increase demand for commodities and push up their prices. Thus, a negative shock to the U.S. dollar could lead to simultaneous increases in the prices of both commodities-linked ETFs and currencies-linked ETFs that track non-U.S. currencies (Frankel, 2008). However, it's crucial to note that many other factors can influence the prices of these ETFs, including changes in interest rates, economic growth, geopolitical events, and market sentiment. Additionally, different commodities and currencies can respond differently to the same factors. Therefore, while there can be a positive relationship between the prices of currencies-linked ETFs and commodities-linked ETFs, this relationship is not guaranteed and can vary over time and across different ETFs.

In the end, the set of securities based on which the analysis is carried out consists only of commodity-linked and currency-linked ETFs that are available for trading in January 2023. Here ETF.com has proved helpful: entering the preferred markets, the site returned 136 ETFs divided into the 7 categories (5 for commodities and 2 for currencies) listed in Table 2.1[3].

Table 2.1: Commodity and Currency categories considered in the dataset

Category	Description
Agriculture	Plants or animals, or parts thereof, used primarily for producing food or fabrics
Broad Market	The commodity fund does not focus on a specific sector
Energy	Products or byproducts that relate to the generation of power
Industrial Metals	Metals whose primary function is in industrial use
Precious Metals	Gold, silver and platinum group metals
Basket	The fund targets a basket of currencies
Pair	The fund targets a pair of currencies

The tickers obtained were then cross-referenced with a database that contained historical intraday bars of 1 minute, 5 minutes, 30 minutes, 1 hour (open, high, low, close, volume) for numerous liquid ETFs[4].
In a context like this, a "database" is nothing more than a folder containing multiple files (usually in .txt or .csv format), one for each security. Each of these files contains a relational dataset with multiple rows, one for each time interval (in our case, one every minute), and for each row a series of information including the closing price of that interval. In other words, a time series. In this phase of the analysis, we will therefore

[3] Please refer to https://www.etf.com/docs/FactSet_ETF_Classification_System_Rules_and_Methodology_June2018.pdf for details of the ETFs classification system.
[4] Please refer to https://firstratedata.com/b/17/etf-complete-historical-intraday for details on the database used

look at several datasets, one for each ETF. Here the *locals()* function was very useful. If inserted in a loop, the function will easily allow you to create multiple variables, one for each ETF. This of course is not the only solution. In fact, we could store the datasets in a dictionary whose keys will be the names of the various ETFs (the tickers), and the corresponding historical series for values. Or we could create a single dataset with as many columns as there are ETFs. The only problem here is if you need more information than just closing prices, but a multi-level index will do the job.

Finally, other important information on the database from which the data on which the analysis is based is extracted is listed below:

- All prices and volumes are fully adjusted for both splits and dividends[5]
- Timezone is US Eastern Time
- Timestamps run from the start of the period (e.g. 1min bars stamped 09:30 run from 09:30.00 to 09:30.59)
- Volume Numbers are in individual shares
- Times with zero volume are omitted (thus gaps in the data sequence are when there have been no trades)

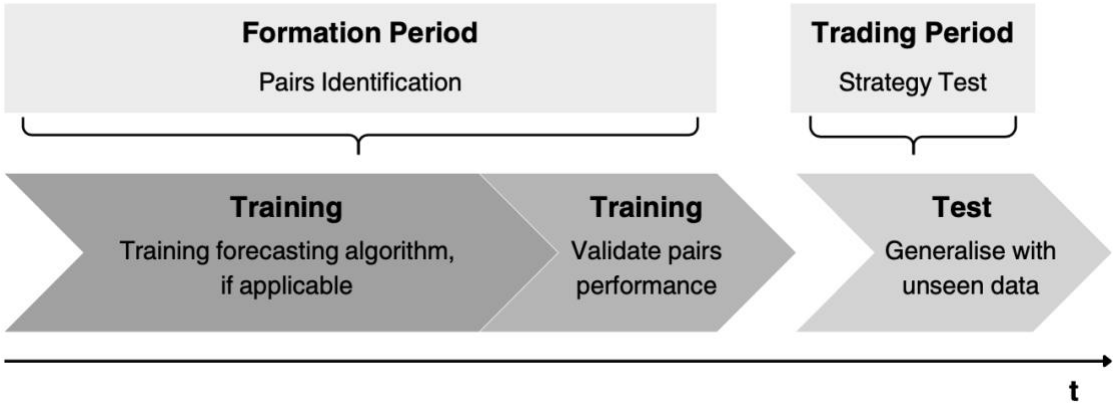
The last one is also the one to which more attention should be paid since, as we will see later, it forces us to put forward an important hypothesis, namely that in the absence of a specific minute of the day the price is assumed unchanged.

2.1.3 Time frame of choice

To better understand this section, it is helpful to first clarify some terminology. In every trading simulation, the data must be divided into two distinct periods: the formation period and the trading period. The formation period mimics the information accessible to an investor before partaking in any transaction, serving three primary functions. Firstly, it identifies the most attractive potential pairings. Secondly, a smaller subset of data, referred to as the validation set, is employed to approximate the strategy's recent performance. Thirdly, exclusive to forecasting-based models, a segment of the formation period, known as the training set, is utilized for training the prediction algorithm. The trading period is designed to emulate the performance of the adopted trading model when faced with previously unencountered data. Figure 2.2 visually represents the organization of these phases.

[5] Please refer to https://firstratedata.com/about/price_adjustment for details of the adjustments.

Figure 2.2: Period decomposition



We can now describe the reference time frame. As anticipated, the chosen price frequency is 1 minute. In general, the rationale for using intraday data is threefold. First, finer granularity allows for more precise definition of entry and exit points, potentially increasing profit margins. Second, it could reveal intraday mean reversion patterns that would otherwise be undetectable. Finally, it offers a larger dataset, making it easier to train sophisticated forecasting models while reducing the risk of overfitting. In addition, we have decided to further increase the frequency, going from the 5 minutes of Horta and Sarmento to 1 minute. This allows us to reduce the reference time frame while maintaining the same number of datapoints and, as we will see later, to find more pairs. As a result, we opted for a 6-month training period, a 1-year validation and a 1-year test period. We could have reduced these to six months too, however Do and Faff (2012) claims the profitability can be increased if the initial 6-month trading period proposed in Gatev et al. (2006) is extended to 1 year. Specifically, data will be extracted from June 2021 to December 2022.

2.2 Data Cleaning

Before delving into the search of promising pairs, some data cleaning steps are implemented, as depicted in Figure 2.3.

Figure 2.3: Data cleaning steps

First, ETFs with insufficient data are excluded. An ETF has “insufficient data” if it is not active throughout the reference period. It is important to acknowledge that focusing solely on these ETFs introduces survivorship bias. By excluding delisted ETFs, potential positions that may have been abruptly terminated

are not considered in the results. This occurs due to the unavailability of a survivorship-free dataset containing the desired ETFs. To mitigate this effect, recent periods are used whenever possible.

Regarding missing values, the issue becomes somewhat more intricate. As previously mentioned, data gaps occur when no trades have taken place. It follows that, at least theoretically, there shouldn't be any missing values. Thus, the challenge lies in determining how to fill these gaps. One possible approach is to presume a stable price and a trading volume of zero during these periods. This hypothesis may be bold but, in my opinion, it represents the optimal choice when faced with a database built in this way. Here the *fillna()* function with the "*ffill*" method was helpful.

Next, we remove ETFs that don't meet minimum liquidity requirements to ensure realistic transaction costs associated with bid-ask spreads[6]. The minimum liquidity criteria follow those used in Do & Faff (2012) and Gatev et al. (2006), which discard ETFs not traded for at least one day.

Ultimately, certain price series occasionally encompass outliers. Outliers are data points in a dataset that significantly differ from the others. They deviate from the general pattern and can impact data analysis. It's crucial to identify and handle them appropriately, as they could signal errors, anomalies, or important insights. Various techniques exist in the literature for anomaly detection, such as cluster analysis or other machine learning approaches. However, given the infrequency of these events in the dataset, I propose a simpler technique. For each price series, the return series is calculated and the points with percentage changes greater than 10% - an unlikely occurrence within 1 minute – are identified as outliers. If a single outlier exists in an ETF's time series, it is corrected using Winsorization. If multiple outliers are present, the ETF is discarded.

Winsorization is a statistical technique used to minimize the influence of extreme values on the data. It can be performed in different ways, but the most common method involves replacing values that fall outside of a specified percentile range with the nearest value within that range. For example, a 5% Winsorization of a dataset involves replacing values that fall below the 5th percentile with the value at the 5th percentile, and values that fall above the 95th percentile with the value at the 95th percentile. If your data has a wide range of values and few outliers, you can use a higher percentile range, such as 99%, to preserve more of the original data.

Winsorization can be a good option when you have a small number of outliers that are skewing the distribution of your data, but you want to retain the original data scale and avoid removing the outliers completely. In this case, Winsorization can help to reduce the impact of the outliers on your statistical measures while still preserving the shape of the data.

The *winsorize()* function from the scipy.stats.mstats module provides a pre-built implementation of Winsorization that can be used to Winsorize a given dataset.

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

Finally, during the process of training forecasting-oriented models, it is essential to normalize the data using the following formula: $St\_norm = (S_t - \mu_s) / \sigma_s$, where St_norm signifies the standardized spread St. This normalization ensures that the series has a mean of zero and a variance of one. By normalizing the input data, neural network training can be expedited and the likelihood of encountering local optima is reduced.

3. Research Stage 1: Pair Selection and Clustering

The following chapter consists of three sections: (i) pair space definition, (ii) pairs selection and (iii) performance evaluation. In the first section we will explain the three clustering methods we know to define the space of candidate pairs. Particular attention will be given to the one proposed by Horta & Sarmento (2020), i.e. the one that uses unsupervised learning. In the second we will explain the criteria by which certain pairs are chosen to be traded. Finally, we will implement a trading algorithm to understand which clustering method led to the choice of the most profitable pairs.

3.1 Pair space definition: clustering using unsupervised learning

As anticipated, the simplest method to identify candidate pairs is to compare each security with every other security in the dataset and generate all potential pair combinations. Two drawbacks of this method are the of multiple comparisons problem and the high computational power requirements.
The second method consists in comparing each security exclusively with those belonging to the same sector. In our case, the sectors are represented by the seven categories that we have previously described. This method is relatively simple to implement, but it is precisely from this simplicity that its main disadvantage derives: the more traders know about pairs, the more difficult it becomes to find pairs that are not already heavily traded, leaving less room for profit.
This perceived imbalance fuels the quest for an approach that achieves a harmonious equilibrium between the two situations: an efficient pre-segmentation of the asset universe that avoids restricting pair combinations to apparent solutions, without necessitating an excessive number of search combinations. In the present study, we suggest employing an unsupervised learning technique on the expectation that it will deduce significant groupings of assets for choosing pairs. The main goal is to enable the data to reveal its inherent patterns, as opposed to assigning predefined categories to each security individually.
However, in order to implement this method, it is first necessary to reduce the dimensionality of each time series using principal component analysis (PCA).

3.1.1 Dimensionality reduction

In the pursuit of identifying profitable pairings, our goal is to discover financial instruments with the same systematic risk exposure. As per the Arbitrage Pricing Theory (APT)[7], such securities yield identical long-term expected returns. Discrepancies from these theoretical expected returns can be interpreted as mispricing, providing a basis for trading decisions. To uncover the shared underlying risk factors for each

[7] The APT is a model according to which the return of a share is expressed as a function of the returns of a series of risk factors (e.g. factors linked to macroeconomic variables such as the price of oil or GDP; but also factors of a different nature).

security, it is suggested to apply Principal Component Analysis (PCA) to the return series, as detailed in Jolliffe (2011).

PCA is a statistical method that utilizes an orthogonal transformation to convert a set of potentially correlated variables into a set of linearly uncorrelated variables referred to as principal components. This transformation is structured in a way that the first principal component captures the highest achievable variability within the data. Subsequent components, while maximizing variance, are also designed to be orthogonal to the ones that came before them.

The PCA process is carried out as follows. Firstly, the return series for security i at time t, $R_{i,t}$, is derived from the security price series P_i. The return series must then be normalized, as PCA is sensitive to the relative scaling of the initial variables. This normalization is achieved by subtracting the mean, R_i, and dividing by the standard deviation σ_i, as in the equation we saw in Section 2.2. The correlation matrix ρ is subsequently calculated from the normalized return series of all assets. The function of ρ will be elucidated shortly. However, it should be emphasized that the application of PCA to return series arises from the fact that a return correlation matrix offers more information to evaluate price co-movements. Instead, the use of price series could lead to the identification of false correlations due to underlying time trends. The next step involves extracting the eigenvectors and eigenvalues in order to construct the principal components. The eigenvectors identify the directions of maximum variance, while the eigenvalues quantify the variance along the corresponding direction. Singular value decomposition (SVD) can be employed to determine eigenvalues and eigenvectors. For this purpose, we arrange the normalized return series for all n securities into matrix A. By directly applying the SVD theorem, A is decomposed as $A = USV^T$. Matrix U is orthogonal, with its columns representing the left singular vectors. Matrix S is diagonal and contains the singular values sorted in descending order along the diagonal, from the highest variance eigenvalue to the lowest. Matrix V is the transposed orthogonal matrix, with its rows comprising the right singular vectors.

Note that by multiplying A^TA, we obtain VS^2V^T. However, AT^A is precisely the correlation matrix ρ calculated earlier, so V can be determined. In this manner, it is possible to find the eigenvectors of the data matrix A. At this juncture, the k eigenvectors corresponding to the k directions of maximum variance are selected, where k represents the number of features to describe the transformed data. The more eigenvectors considered, the better the data is described. The matrix containing the selected eigenvalues in order of significance is referred to as the feature vector.

Ultimately, the new dataset is obtained by multiplying the original matrix A by the feature vector, resulting in an $n \times k$ matrix. It is important to note that the position (i, j) of the final matrix contains the product result between the normalized return series i and the eigenvector j. The resulting matrix is now reduced to the selected k features.

To conclude, the number of features, k, must be determined. Typically, a common approach involves assessing the proportion of the overall variance explained by each principal component and selecting the number of components that collectively explain a fixed percentage, as described in Avellaneda and Lee

(2010). However, in this study, we employ a different methodology. However, this work adopts a different method. Since an Unsupervised Learning algorithm is applied using these data features, the data dimensionality should be considered. High data dimensionality poses a dual challenge. Firstly, it increases the likelihood of uncovering irrelevant features. Secondly, it introduces the issue known as the "curse of dimensionality," a concept coined by Bellman (1966) to describe the problem stemming from the exponential expansion of volume when adding extra dimensions to Euclidean space. This becomes particularly problematic when measuring the distance between seemingly similar data points, as they can suddenly appear very distant from each other. Consequently, this can severely hamper the effectiveness of the clustering process. Berkhin (2006) notes that the effect starts to become severe for dimensions greater than 15. Bearing this in mind, the number of PCA dimensions is capped at this value and is chosen empirically.

Table 3.1 showcases the variations in clusters and identified pairs as the number of principal components fluctuates.

Table 3.1: Clustering statistics when varying number of principal components

Number of principal components	3	5	8	12
Number of clusters	4	4	3	2
Average cluster size	7	7	7	7
Pair selected	21	13	6	2
Pair selected from distinct categories	11	11	2	2

The last line highlights the pairs made up of securities belonging to different categories, thus underlining the diversity offered by unsupervised learning.

In the end I settled on three main components. Given the lack of evidence in favor of higher dimensions, I preferred to represent the ETFs in a lower one. As we will see in the next paragraph, as long as it falls within an acceptable size range, this parameter should not significantly influence the results.

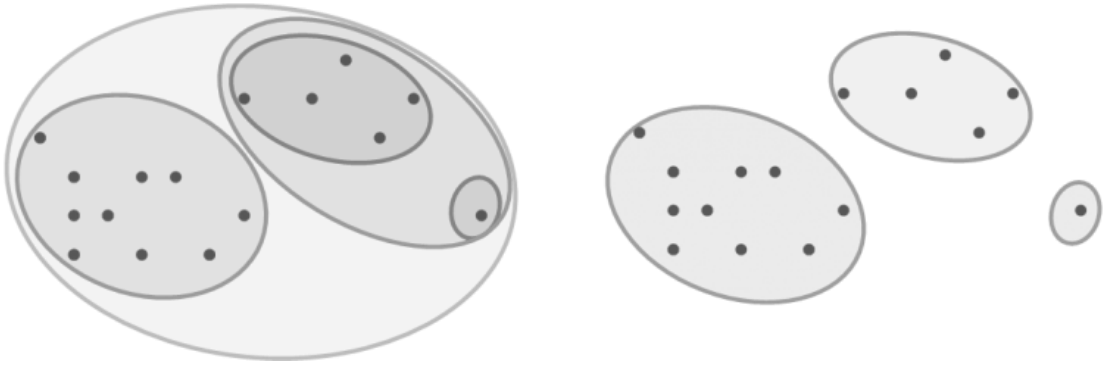
3.1.2 OPTICS

Unsupervised learning is a type of machine learning where the algorithm is not given any labeled data to learn from, but instead must discover the underlying structure or patterns in the data on its own. One common application of unsupervised learning is clustering, where the algorithm groups similar data points together based on some measure of similarity. According to the book “Hands-On Unsupervised Learning Using Python” by Ankur A. Patel, clustering “involves partitioning a dataset into groups, called clusters, such that data points within a cluster are similar to each other, while data points in different clusters are dissimilar.” Clustering methods can be divided into three broad categories: partition-based, hierarchical, and density-based clustering.

Partition-based clustering, also known as centroid-based clustering, attempts to partition the dataset into a predetermined number of clusters. The most well-known partition-based clustering algorithm is the K-means algorithm. K-means aims to minimize the sum of squared distances between each data point and the centroid of the cluster it belongs to (MacQueen, 1967). Partitioning techniques are generally regarded as inappropriate for what we are trying to do here. First, handling noisy data and outliers is a challenge for them. Secondly, partitioning algorithms impose convex cluster shapes and assuming that the data follows a normal distribution around the cluster center might be too rigorous, especially in the case of high data dimensionality. Third, determining the number of clusters beforehand is not optimal when aiming for models with minimal parameters. It may be contended that when performing a grid search to determine the optimal number of clusters, the resulting partition should conform to the configuration that achieves the lowest value of a designated criterion. Therefore, it would be satisfactory to indicate a cluster range. Nevertheless, the criteria utilized for identifying the most favorable configurations remain ambiguous. Although clustering integrity measures such as the silhouette coefficient proposed by Rousseeuw (1987) might seem fitting, the empirical findings of Horta & Sarmento (2020) suggest that a strong correlation between greater cluster integrity and more promising pairs doesn't necessarily exist.

Hierarchical clustering, on the other hand, builds a tree-like structure of nested clusters, which can be represented as a dendrogram. This method can be further divided into two sub-categories: agglomerative and divisive. Agglomerative hierarchical clustering starts by treating each data point as a separate cluster and iteratively merges the closest clusters until only one cluster remains (Johnson, 1967). Divisive hierarchical clustering, as the name suggests, begins with one cluster containing all data points and recursively splits it until each data point forms its own cluster (Sneath, 1957). Hierarchical clustering techniques offer the benefit of providing a dynamic termination criteria. This enables investors to choose a cluster separation that matches their preferred level of detail. However, this could inadvertently introduce investor-related biases. Consequently, the chosen clusters may resemble outcomes achieved through conventional search approaches, which is what we seek to avoid. Therefore, employing hierarchical clustering is only suitable if an automated stopping criterion is present.

Figure 3.1: Hierarchical and partitional clustering diagram

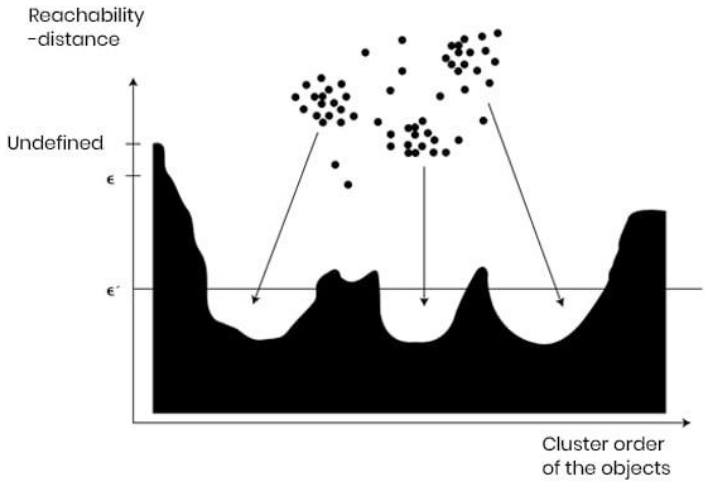


a) Hierarchical clustering b) Partitional clustering

In conclusion, density-based clustering algorithms identify clusters by searching for areas of high data point density and separating them from areas with lower density (Ester et al., 1996). Density-based clustering techniques showcase certain advantages relevant to this research. Firstly, these techniques allow for the formation of clusters with various shapes, eliminating the need for assumptions about data conforming to Gaussian distributions. Additionally, such methods exhibit resilience to outliers, as they do not mandate the inclusion of all data points within a cluster. Furthermore, there is no need to predetermine the quantity of clusters. Two prominent density-based clustering algorithms are DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points to Identify the Clustering Structure). DBSCAN was introduced by Ester et al. in 1996. As anticipated, it operates on the premise that a cluster is a dense region of data points separated by areas of lower point density. The algorithm has two key parameters: a distance threshold (Eps) and the minimum number of points required to form a dense region (MinPts). It works by selecting an arbitrary unvisited data point and determining if it's part of a cluster by examining its neighborhood within the Eps distance. If there are at least MinPts points in the neighborhood, a new cluster is formed, and the process continues by expanding the cluster iteratively to include all reachable points within the Eps distance. If the neighborhood has fewer than MinPts points, the data point is marked as noise. This process is repeated until all data points have been visited.

OPTICS (Ordering Points To Identify the Clustering Structure), developed by Ankerst et al. in 1999, is an extension of the DBSCAN algorithm that addresses some of its limitations. Unlike DBSCAN, OPTICS doesn't require specifying the Eps parameter, making it better suited for handling datasets with varying density. OPTICS generates an ordered list of data points, representing the density-based clustering structure of the dataset, with a reachability-distance value assigned to each point. It begins by selecting an unprocessed data point and calculating its core distance, which is the distance to the MinPts-th nearest neighbor. Then, it processes the data point's neighbors and calculates their reachability distances, updating them if a shorter distance is found. This process is repeated for each subsequent point in the dataset. The result is an ordered list of data points that can be visualized and analyzed to reveal the underlying cluster structure.

Figure 3.2: Obtaining clusters with OPTICS



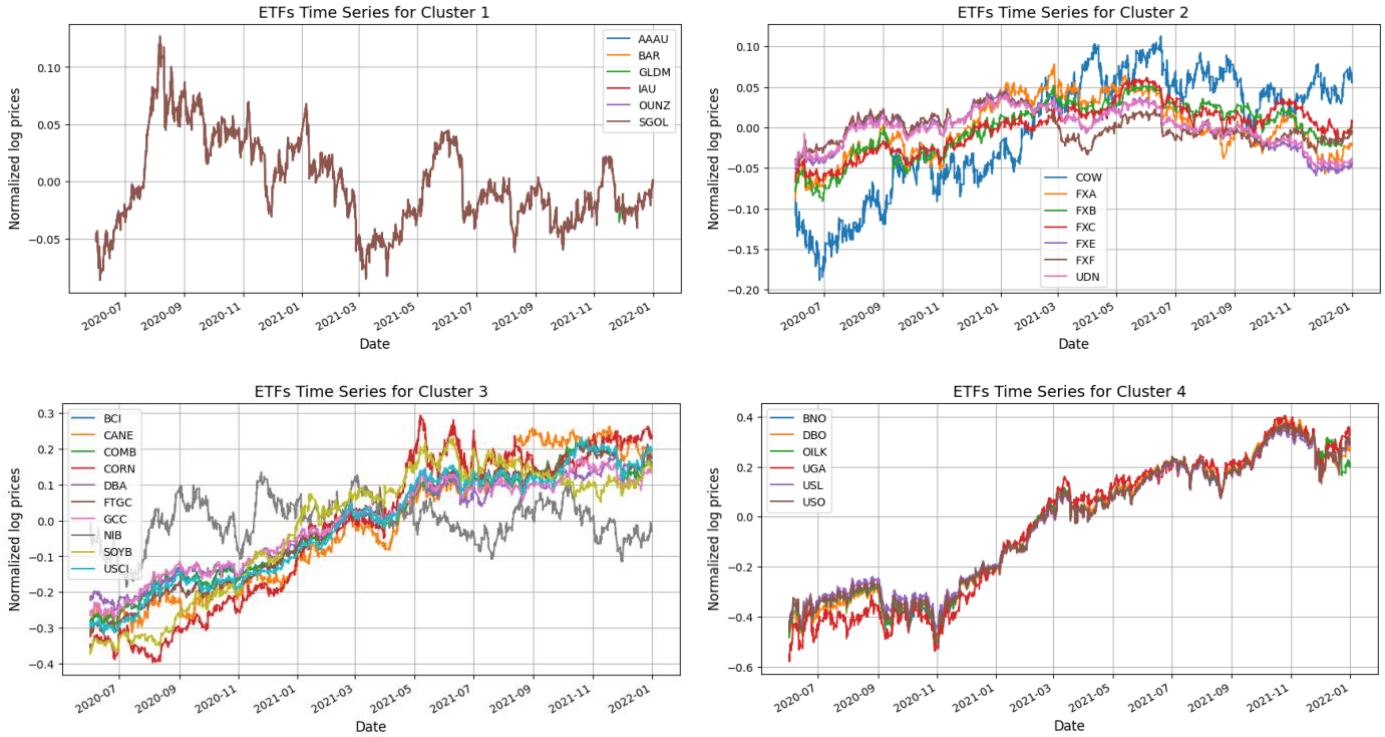
OPTICS offers some advantages over DBSCAN. Firstly, it can effectively handle datasets with varying densities, as it doesn't rely on a single Eps value. Instead, it examines the dataset's intrinsic clustering structure. For instance, the results presented in Table 3.1 reveal that the impact of modifying the number of principal components within the tested range is minimal. The consistent number of clusters and their average size across different principal component counts can be attributed to the stability provided by the OPTICS algorithm. If the number of principal components is chosen within a reasonable range, the OPTICS algorithm automatically adjusts its parameters to best fit the data across various dimensions. Achieving the same outcome using DBSCAN is significantly more challenging, as investors would need to identify the optimal value for the ε parameter in each case. Secondly, OPTICS provides an ordered list of data points that can be easily visualized and analyzed to determine the clustering structure, whereas DBSCAN produces a set of clusters that may require additional interpretation.

Now let's see the results derived from the application of OPTICS on our dataset. Below is the composition of the clusters identified by OPTICS after reducing the dataset to three principal components[8]. Each

[8] To identify which columns of the original dataset are within the clusters, it is first necessary to determine which original variables are most associated with each principal component. This can be done using principal component loadings, which are obtained by multiplying the original dataset by the eigenvectors of the principal components. For example, suppose you have a dataset with five variables (X1, X2, X3, X4, X5) and you have extracted three principal components. The loadings for the first principal component might be X1: 0.5, X2: 0.4, X3: 0.7, X4: 0.1, X5: 0.2. This means that variable X3 has the strongest association with the first principal component, followed by variables X1 and X2. Variables X4 and X5 have a weaker association with the first principal component. Once you have identified which variables are most strongly associated with each principal component, you can look at the cluster assignments for each observation in the dataset. The cluster assignments will indicate which observations are similar to each other in the principal component space. You can then examine the original variables associated with each observation to determine which variables are most commonly associated with each cluster. For example, suppose you have three clusters, labeled 1, 2, and 3. You can examine the observations in each cluster and count the number of times each variable appears. If you find that variables X1, X2, and X3 appear most frequently in cluster 1, you can say that these

visualization represents the logarithm of the price series for each ETF included in the cluster. The original price series has been adjusted by subtracting its average value for clarity.

Figure 3.3: Price series composition of the clusters formed in the formation period



Beginning with Figure 3.3 (a), it is fascinating to note the proximity of the series to one another, making them difficult to distinguish. This raises the question of whether a fundamental explanation exists for this pattern. Indeed, there is one. The four securities pinpointed are not only part of the Precious Metals category but also have a connection to Gold. This serves as initial proof that the OPTICS-driven method can delve further into categorizing ETFs, as it can discern not only general categories but also specific segments.
A comparable scenario can be found in Figure 3.3 (d). Here, the identified securities also fall within a single segment, Energy Crude Oil, which is a component of the broader Energy sector.
Figure 8.b reveals that the OPTICS clustering abilities extend beyond merely identifying ETFs within the same segment. In cluster 2, we can find ETFs related to the Agriculture sector (COW), ETFs connected to funds focusing on currency pairs (FXA, FXB, FXC, FXE), and ETFs linked to funds targeting a collection of currencies (UDN). Despite not all belonging to the same category, a discernible relationship exists among the identified price series. The same holds true for cluster 3, which consists of two distinct categories: Agriculture (CANE, CORN, DBA, NIB, SOYB) and Broad Market (BCI, COMB, FTGC, GCC, USCI). Unfortunately, in none of the clusters formed by OPTICS there is a security belonging to the basket and pairs categories or, more generally, to the currencies sector. This refutes the first of the hypotheses advanced

variables are inside cluster 1. Similarly, if you find that variables X4 and X5 appear most frequently in cluster 2, you can say that these variables are inside cluster 2.

at the beginning of this study according to which, the addition of a sector among those from which to draw would have determined the selection of a greater number of couples. Despite that, we can verify that the clusters formed achieve their goal of integrating the desired characteristics from the two other methods examined. Specifically, they display a tendency to group subsets of ETFs from the same category while still allowing for clusters containing ETFs from various categories.

3.2 Pairs selection

Once the clusters have been defined, it remains essential to establish a series of criteria to select the pairs to be traded. Ensuring the stability of couples is essential. With this in mind, we suggest combining techniques that have been used independently in previous studies. In particular, a pair is selected if it meets the following four conditions:

1. The components of the pair exhibit cointegration;
2. The spread Hurst exponent for the pair demonstrates a tendency towards mean reversion;
3. The divergence and convergence of the pair's spread occur within suitable timeframes;
4. The frequency of the pair's spread reverting to the mean is adequate.

We will now delve into each stage more comprehensively to elucidate the underlying rationale.

First, a pair a pair is only deemed eligible for trading if the two securities that form the pair are cointegrated. To better understand the concept of cointegration and why it is crucial in this context, we must first introduce the notion of stationarity. A random process is considered stationary if its mean and variance remain constant over time, as stated by Adhikari and Agrawal (2013). Consequently, a time series that is stationary exhibits mean-reverting behavior, with oscillations around the average exhibiting similar magnitudes. A stationary process can also be described in terms of its order of integration, I(d). In this sense, a stationary time series is identified as an I(0) process, while a non-stationary one as I(1). The order of integration, d, serves as a summary statistic representing the minimum number of differences needed to obtain a stationary series. To ascertain whether a time series is stationary or not, a widely-used method is the Augmented Dickey-Fuller (ADF) test. Developed by Dickey and Fuller in 1979, this test is a statistical technique designed to determine the presence of unit roots in a time series, which are indicative of non-stationarity[9]. The null hypothesis of the ADF test is that the time series contains a unit root (i.e., it is non-stationary), while the alternative hypothesis is that the series is stationary.

We can now move on to the concept of cointegration. Cointegration was first proposed by two econometricians, Engle & Granger (1987). A group of variables is considered cointegrated when it's possible

[9] In more technical terms, a unit root exists when the autoregressive parameter of a time series is equal to one (1). In such a scenario, the time series exhibits a random walk behavior, which results in the absence of mean reversion and constant variance over time (Enders, 2010).

to find a linear combination of these variables, typically of order 'd,' that leads to a reduced order of integration, denoted as I(d-1). In the context of this study, cointegration is verified when a set of I(1) variables, which are non-stationary, can be effectively utilized to model an I(0) variable, which is stationary. Formally speaking, the concept of cointegration involves two time series, y_t and x_t, which are both characterized as I(1). This implies that there exist coefficients, denoted by μ and β, such that the equation $y_t - \beta x_t = u_t + \mu$ holds, where u_t represents a stationary time series. The significance of this relationship lies in the fact that it provides a formal means of generating a stationary time series for trading purposes. In order to examine this condition, we suggest utilizing the two-step Engle-Granger cointegration test, as it offers a straightforward approach. The test is carried out as follows:

1. Employ the Augmented Dickey-Fuller (ADF) test to determine if a unit root exists in the series y_t and x_t. If confirmed, move to step 2.
2. Execute the regression defined as above using Ordinary Least Squares, and store the residuals, $\hat{u}_t$.
3. Utilize the ADF test (or a similar test) to examine the residuals $\hat{u}_t$ for a unit root.
4. Rejecting the null hypothesis of a unit root in the residuals (the null hypothesis of no cointegration) implies that the residual series is stationary and the two variables are cointegrated.

A significant drawback of the Engle-Granger method is that selecting the dependent variable may yield differing outcomes, as highlighted by Armstrong (2001). One solution to address this problem would be to perform the test for both potential choices of the dependent variable (i.e., for both securities that make up the pair), and then choose the option that yields the lowest t-statistic.

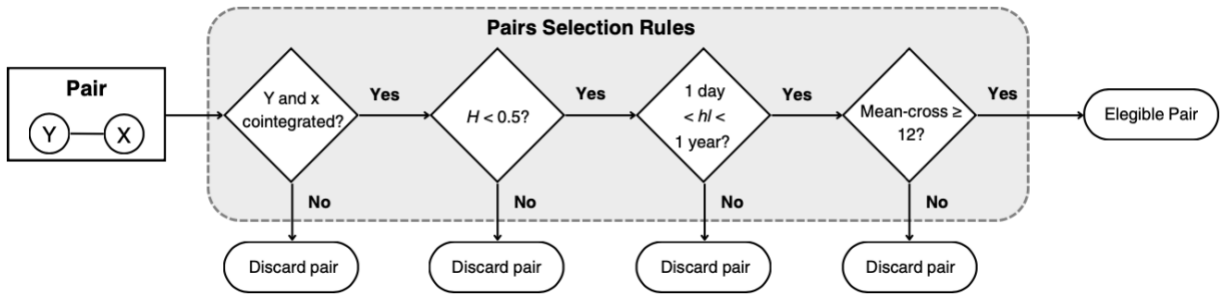
To enhance confidence in the mean-reversion behavior of the spreads in pairs and to mitigate the risk of false positives caused by the multiple comparisons issue, an additional validation step is suggested. This step involves enforcing a condition that the Hurst exponent associated with the spread of a given pair is less than 0.5. The Hurst exponent serves as an indicator of the long-term memory of time series. It relates to the autocorrelations present within the time series and the rate at which these diminish as the time lag between value pairs increases. The Hurst exponent ranges between 0 and 1, with three scenarios emerging based on its value. When $H > 0.5$, persistence is observed, suggesting that an increase (or decrease) in the time series values is more likely to be followed by a similar trend. If $H < 0.5$, the time series demonstrates anti-persistence or mean reversion, implying that an increase (or decrease) in the time series values is more likely to be followed by an opposing trend. Lastly, if $H = 0.5$, the time series follows a random walk and is considered to be uncorrelated (Peters, 1994). There are several methods for calculating the Hurst exponent, such as the rescaled range (R/S) analysis, the detrended fluctuation analysis (DFA), and the wavelet-based methods. In the R/S analysis, the time series data is divided into non-overlapping segments, and for each segment, the range (R) and standard deviation (S) are calculated. The rescaled range is obtained by dividing R by S, and the Hurst exponent is estimated by analyzing the scaling behavior of the rescaled range as a function of the segment size (Hurst, 1951).

It is important to recognize that the Hurst exponent evaluates the extent of mean reversion in a time series, rather than its stationarity. In other words, it should not be considered a direct substitute for the cointegration test. While it is true that most stationary time series also exhibit mean reversion, there can be exceptions. For instance, ponder the process X that follows the rule $X_t = X_{t-1}$ for $t > 0$, and X_0 is assigned the value 1 with a 50% probability and 0 otherwise. Although X is stationary, it does not display mean reversion, demonstrating that stationarity does not necessarily equate to mean reversion. However, such cases are infrequent, and thus, we disregard them in our approximation.

Thirdly, the stationarity of the spread is a sought-after attribute when selecting pairs. Nevertheless, a mean-reverting spread alone doesn't guarantee profits. The duration of mean-reversion and the trading timeframe must align. A significant concept in this context is the half-life, which can be viewed as an approximation of the anticipated time for the spread's mean-reversion (Bouchaud, 2018). Essentially, the half-life represents the time it takes for a spread to revert halfway back to its mean from its current position.
If a spread takes about 400 days for mean-reversion and the trading timeframe is one year, it's improbable that a profitable situation arises. Similarly, an extremely brief mean-reversion period is also unfavorable. As the half-life plays a crucial role in estimating the expected time for mean-reversion, we suggest eliminating pairs whose half-life doesn't align with the trading timeframe.

Moreover, we stipulate that each spread must intersect its mean on at least a monthly basis, a requirement put in place to ensure an adequate level of liquidity. It's worth noting that, even though there exists a (negative) correlation between the frequency of mean intersections and the duration of half-life - more mean intersections often correspond to a shorter half-life - these characteristics are not always interchangeable, as demonstrated by Rama & Tankov (2003). By incorporating this constraint, we not only strengthen the previous condition but also filter out pairs that, despite meeting the timing criteria for mean reversion, fail to intersect the mean. Such pairs offer no opportunities for exiting a position.

Figure 3.4: Pair selection criteria



After outlining the four steps for determining a suitable pair, we now detail the parameters employed in each phase. Figure 3.4 illustrates the four criteria a pair must satisfy to be eligible for trading. Firstly, pairs must exhibit cointegration with a 5% p-value. Next, the Hurst exponent of the spread, denoted by H, should be

less than 0.5. Moreover, the half-life duration, represented by hl, should fall between one day and one year. Lastly, it is mandated that the spread intersects a mean at least 12 times per year, ideally equating to a minimum average of one intersection per month.

3.2.1 Pairs selection performance

We commence by showcasing the quantity of pairs that the system chooses under the three distinct clustering techniques. These numbers are displayed in Table 3.2.

Table 3.2: Pairs selected for various clustering methodologies

	No clustering	By category	OPTICS
Number of clusters	1	7	4
Possible combinations	1035	155	96
Pairs selected	141	24	21

As anticipated, the single cluster method encompasses a wider range of ETFs to pick the pairs from. Naturally, this setup yields more pairs. Yet, it remains to be evaluated whether all the chosen pairs are similarly lucrative.
It can be seen that when we categorize the ETFs into seven groups (based on the classifications defined in Sect. 5.2.2), it leads to a decrease in the potential pair combinations.
When utilizing OPTICS, the number of potential pair combinations is significantly less. These outcomes are achieved using three main components to depict the reduced data. Moreover, the only parameter for OPTICS to be determined, minPts, is assigned a value of three, as we have empirically confirmed that this value facilitates a reasonable distribution of clusters. Even though the quantity of clusters surpasses that when sorted by category, their smaller size leads to less combinations. This not only minimizes the computational load of conducting multiple statistical tests, but it also lowers the chance of detecting spurious pairs.

Let's now explore in depth the process of selecting pairs. The pair elimination at each stage of the process is shown in Table 3.3.

Table 3.3: Progress in the selection of pairs within an unconstrained search environment.

Pairs eliminated per stage	
1. Cointegration	1004
2. Hurst exponent	69
3. Half-life	4
4. Mean-crosses	0
Not eliminated	186
Total Pairs	1286

It's crucial to acknowledge that these steps occur in a sequential manner. This implies that each row exhibits the pairs that have been removed from the subset created by the prior selection criteria. Consequently, it's natural to assume that the largest portion of pairs are weeded out at the initial stage, and our observations align with this assumption. Furthermore, the function of the Hurst exponent in removing pairs that pass the cointegration test, yet their spread didn't record a Hurst exponent under 0.5, thereby not portraying a mean-reverting process, is substantiated.
The data in Table 4 also suggests that certain pairs are disqualified due to the incompatibility between their convergence period and the trading timeframe, which leads to the non-fulfillment of the half-life requirement. However, it's clear that the mean-crossing stipulation is satisfied by the entire subset of pairs that met the earlier requirements, signifying that applying this final regulation is superfluous in this instance. To wrap up, we can assert that the Hurst exponent and the half-life constraints play a substantial role in sifting out pairs that would not have been ruled out if only cointegration was imposed.

We conclude the paragraph by noting how having increased the frequency to one minute has allowed us to validate another of the hypotheses on which this study is based. In fact, the possibility of reducing the training period allowed us to find, proportionally, many more couples than in Sarmento & Horta (2020), regardless of the clustering method used: 13.6% against 3% in the case of no clustering, 15.5% against 1.6% clustering by category and 21.9% against 21.2% in the case of OPTICS.

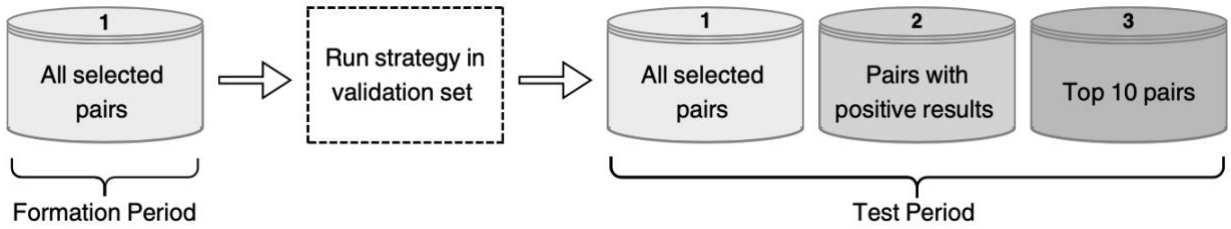
3.3 Performance evaluation

The objective of this last section is to evaluate the trading performance of the pairs that have been selected within each of the three cluster settings. By doing so we will be able to identify the most effective clustering approach to produce a high potential investment portfolio.

3.3.1 Test Portfolios

Let's start by saying that three separate test portfolios are used to simulate all potential trading circumstances. The first portfolio includes all pairs identified during the training period. The second portfolio capitalizes on the insights gained from executing the strategy within the validation set, picking only the pairs with favorable outcomes. The third and final portfolio reflects a scenario where an investor has to commit to a fixed number of k pairs. In this case, we propose to select the top-k pairs based on the returns obtained in the validation set. We adopt k = 10, which is an intermediate value between Gatev et al. (2006) choices of k = 5 and k = 20. Through the examination of various portfolio structures, we aim not only to identify the most effective clustering approach, but also to evaluate its ideal implementation conditions. Figure 3.5 illustrates the three portfolios.

Figure 3.5: Test portfolios



It should be highlighted that while choosing pairs according to their performance on the validation set may appear suitable, it is not certain that the top-performing pairs within the validation set will exhibit a similar pattern in the test set. The effectiveness of this guiding principle will be determined by the outcomes.

3.3.2 Threshold-based trading model

As the primary objective of this research stage is to assess the outcomes of the pair selection methods in relation to one another, optimization of the trading model is not our focus. Consequently, we employ the conventional threshold-based model with the parameters specified in Table 3.4.

Within the framework suggested by Gatev at al. (2006), the initiation of a trade is contingent on the divergence of the spread. When the spread between a pair's two price series exceeds twice the historical standard deviation, a trade is initiated. Trades are then closed when the spread converges to the average, at the conclusion of the trading period, or in the event of delisting. This model is likely to be familiar to readers, as it is the approach introduced in Section 1.2.1.

The model can be formally outlined as follows:

1. Establish the model thresholds: the long position-triggering threshold (α_L), the short position-triggering threshold (α_S), and the exit threshold (α_{exit}) that determines when to close a position.
2. Track the spread (S_t) development and check for threshold crossings.

3. If α_L is crossed, go long on the spread by purchasing Y and selling X. If α_S is crossed, short the spread by selling Y and buying X.
4. Exit the position when α_{exit} is crossed and a position is being held.

It is crucial to acknowledge that the definition of the spread, S_t, is closely associated with the method employed to identify the pairs. For instance, using the minimum distance approach, the spread formed by securities Y and X is straightforwardly expressed as $S_t = Y_t - X_t$.[10] Conversely, with cointegration, the spread takes on a slightly altered form, represented as $St = Y_t - \beta X_t$. This nuance impacts how positions are established.

If the spread is defined as $S_t = Y_t - X_t$, entering a long position involves purchasing Y and selling X in equal share quantities or allocating the same amount of capital to both, assuming the investor desires a dollar-neutral position. However, if the spread is expressed as $S_t = Y_t - \beta X_t$, a decision must be made regarding the treatment of the cointegration factor. Various suggestions have been put forth in academic literature, but no consensus on the most suitable option has been reached. We will see which of these has been adopted in this work in the next paragraph.

Table 3.4: Threshold-based model parameters

Parameters	Values
Long Threshold	$\mu_s - 2\sigma_s$
Short Threshold	$\mu_s + 2\sigma_s$
Exit Threshold	μ_s

Finally, it is important to note that there is existing research that specifically investigates the optimization of trading thresholds, such as the studies by Göncü and Akyıldırım (2016), and Huang et al. (2015). Moreover, some researchers, like Dunis et al. (2010), delve into the utilization of sliding windows to continuously update the mean (μ_s) and standard deviation (σ_s). Nonetheless, we maintain this framework for the sake of simplicity.

3.3.3 Trading Simulation

This segment provides a comprehensive account of the trading simulation. We delve into the various elements that were considered while constructing the portfolio, transaction costs, as well as the position settlement conditions.

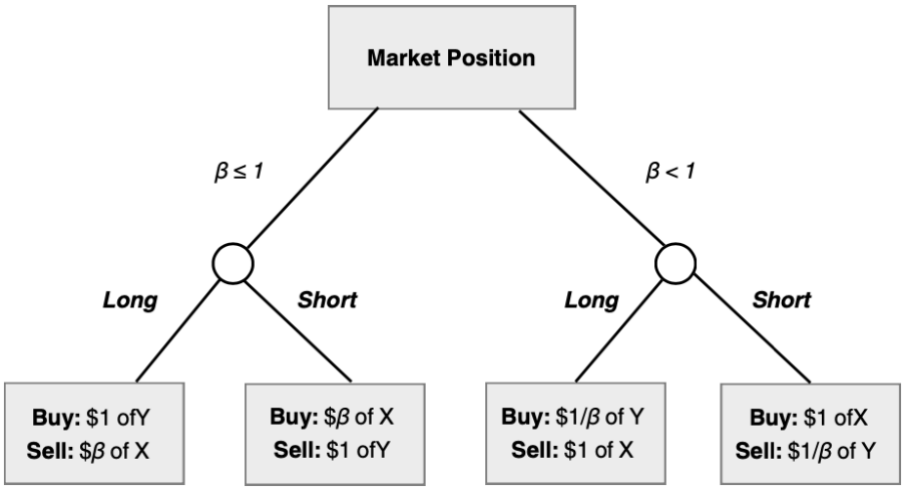
[10] The minimum distance approach is an alternative selection criterion to cointegration which is based on distance and has its foundations in Gatev et al. (2006). In their study, they create an aggregate total return index for each equity, which is then standardized to the initial day of a formation period spanning 12 months. This interval is utilized to compute the Sum of Squared Euclidean Distances (SSD) across the time series they created. Following this, they recommend ordering these pairs based on the least historical SSD.

We begin by observing that the portfolios analyzed in this study may differ in size. Regardless, we determine that all pairs within the portfolio should have equal weight. This allows the portfolio returns to be calculated by simply averaging the performance of all pairs, without worrying about the initial investment's relative proportion.

A logical question that arises is the allocation of capital for each pair. Theoretically, if equal values are maintained in the long and short positions of a pair, the capital gained from the short position can completely cover the cost of entering the long position, requiring no initial financing. This concept is known as a self-financing portfolio. While this notion is theoretically sound, it is not applicable in practice, as collateral is always necessary for borrowing the security being shorted, making zero initial investment impossible. Thus, the required 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 of equal value. By using proceeds from short positions to finance long positions, the investor leverages their investment. This leverage strategy is common in hedge funds to increase the portfolio's absolute return and is therefore considered in this study.

For the sake of simplification, it is beneficial to assume a one-dollar investment in each pair. This method is commonly used by researchers in the field (Avellaneda & Lee (2010), Caldeira & Moura (2013), Dunis et al. (2006), Gatev et al. (2006), Rad et al. (2016)), making it appropriate for comparison. It should be noted that this approximation assumes that an investor can buy fractional shares of trading ETFs since the security is likely not valued at exactly one dollar. In this study, we adopt this assumption based on the idea that the investor can always find the least common multiple between the two security prices, allowing them to take equal-sized positions in both ETFs.
In contrast to some studies that aim to be dollar-neutral, such as Gatev et al. (2009) and Dunis et al. (2010), which invest $1 in both long and short positions, our methodology adheres to the cointegration ratio β between the two securities. We suggest that the value invested in X is β times the value invested in Y. Under these conditions, to maintain a $1 initial investment, it must be ensured that neither the long nor the short position costs more than $1. Formally, the imposed condition is as follows: $\max(leg_1, leg_2) = \$1$, where leg_1 and leg_2 denote the capital invested in each leg of the pair. Based on this, we develop a framework depicted in Figure 3.6.

Figure 3.6: Market position definition



Throughout the trading process, we assume that all capital earned by a pair during the trading period is reinvested in the subsequent trade. For example, if a pair initially trades with a 5% return, the next time a trade is opened for that pair, the starting capital will be $1.05 instead of $1. This mechanism streamlines the calculation of the final return.

We confirm that all outcomes presented in this research take transaction expenses into account. The transaction fees taken into consideration are based on approximations from Do & Faff (2012). Their research delves into the effects of transaction costs on Pairs Trading and offers cautious calculations of all inherent costs. Other researchers in the field, like Huck & Afawubo (2015), have also utilized these estimates. The costs can be separated into three components, as depicted in Table 3.5. The fee is presented in relation to the position's size.

Table 3.5: Transaction costs considered

	Commission costs	Market impact	Rental costs
Description	Value charged when buying or selling a security	Indicator that reflects the cost faced by a trader due to market slippage	Constant loan fee for the short position, payable over the life of each trade
Charge	8 bps	20 bps	1% per annum

A crucial nuance in trading execution pertains to the price points considered when initiating a position. This applies to both threshold models and, as we will see later, forecast-based models, where the system constantly monitors price movements to see if a specific limit is violated, prompting you to establish a position. However, due to the unavoidable delay in a trading system's reaction, a slight variation in the entry price may arise, linked to the time taken to set a position. The way this issue is tackled in the proposed simulation setting merits discussion.

Drawing from the work of Do and Faff (2010), Dunis et al. (2010), and Gatev et al. (2006), we adopt a cautious delay of five periods before entering a position. Therefore, if a position is activated at t = i, we only step into the position at time t = i + 5 min (considering the dataset in question has a 1-minute frequency). This measure is designed to ensure the strategy is implementable in real-world scenarios. If it turns out to be profitable under these conditions, the results could only be enhanced by assuming a less-than-five-minute delay, which is entirely plausible in practice.

In terms of specifying exit points, this study does not incorporate a stop-loss system in any situation. This implies a position is only relinquished if one of two conditions is met. Either the pair reaches convergence or the trading period concludes. In the latter situation, a recently established position might diminish the overall return, despite its potential profitability in the future. However, this closely mirrors how a hedge fund would declare its earnings at the conclusion of a specific period.

3.3.4 Evaluation metrics

We will now discuss the most suitable financial evaluation metrics for assessing the suggested strategies, which include Return on Investment (ROI), Sharpe Ratio (SR), and Maximum Drawdown (MDD).

In its most basic form, return on investment can be determined by dividing net profit by the initial investment: ROI = Net Profit / Initial Investment × 100.

It should be noted that the portfolio creation method used in this study ensures that the required initial investment is always $1. As a result, the return at any given time can also be viewed as the net profit.

When determining the returns for the entire portfolio, a question arises regarding whether the portfolio returns should be averaged across all chosen trading pairs or solely across those that opened positions during the trading period. Gatev et al. (2006) referred to the first approach as return on committed capital (RCC) and the second as fully invested return (FII). This study adopts the first approach, as it is more conservative and takes into account the opportunity cost of hedge funds committing capital to a strategy even if no trades are made.

An additional detail should be highlighted. Dunis et al. (2010) argue that in a realistic setting, the financial institution lending shares would charge interest on the collateral, which should be added to the strategy's net profit. However, we have chosen to disregard this aspect for the sake of simplicity, as it would only account for a small fraction of the net profit.

As explained before, the returns calculated are based on a leveraged position. In the case of an unleveraged position, the initial capital corresponds to the initial gross exposure (the sum of the long position and the short position), resulting in slightly lower returns.

The Sharpe Ratio is a well-established metric in financial analysis that helps investors understand the return of an investment compared to its risk. Named after William F. Sharpe, the ratio is a measure of the excess

return (or risk premium) per unit of deviation in an investment asset or trading strategy, typically referred to as risk (Sharpe, 1966). The formula for calculating the Sharpe Ratio is: Sharpe Ratio = $(R_p - R_f) / \sigma_p$, where Rp represents the expected portfolio return, which is the return that an investor anticipates from a portfolio over a specific period. This value can be calculated by summing the products of the expected return of each asset and its weight in the portfolio.

Rf stands for the risk-free rate. The risk-free rate represents the projected rate of return for an hypothetical investment that carries no financial risk. When evaluating the returns generated by a specific strategy, it is necessary to subtract the risk-free rate to account for the potential interest that could have been earned on the same amount of cash with no risk involved. In this study, we adopt the common practice of setting the risk-free rate equal to the interest paid on the 3-month US government treasury bill. Table 3.6 demonstrates the risk-free annualized rates considered for the tested period. These rates are calculated by averaging the rates of the 3-Month Treasury bill during the corresponding periods. The data is sourced from the Board of Governors of the Federal Reserve System (US).

Table 3.6: Risk-free rates considered per reference year

	2018	2019	2020	2021	2022
R_f	1.94%	2.06%	0.35%	0.05%	2.02%

σ_p, the denominator, represents the standard deviation of the portfolio's excess return, also known as the total risk. Standard deviation provides a measure of the dispersion of a set of values. In finance, it's used as an indicator of investment risk, or volatility. A low standard deviation suggests a tendency towards a mean, while a high standard deviation signals that data points are spread out over a large range of values. However, a significant issue with the Sharpe Ratio arises when considering the annualization factor. Standard practice is to annualize the Sharpe Ratio when the data frequency is not on an annual basis. This is typically accomplished by multiplying the Sharpe Ratio by the square root of the number of periods in a year. This process, though, is founded on the assumption that returns are independently and identically distributed, which is often not the case in the financial markets. To achieve a more precise approximation, we embrace the correction factor proposed by Lo (2002). This involves measuring the serial correlation[11] of portfolio returns and subsequently applying a scale factor based on the findings.

The Sharpe ratio, which normalizes returns by factoring in the associated risks, is contrasted in this study with the Maximum Drawdown. The latter metric signifies the largest noted downturn from a high point to a

[11] Serial correlation refers to the correlation between values of a time series that are separated by a given lag or time interval. Serial correlation is important to consider when analyzing time series data because it can impact the accuracy of statistical inference and prediction. If there is significant serial correlation in a time series, it can violate the assumption of independence between observations that underlies many statistical methods. This can lead to biased parameter estimates, incorrect confidence intervals, and reduced predictive accuracy.

subsequent low in a data sequence, before a fresh peak is reached. Throughout the trading duration, this metric is specifically evaluated in relation to the balance of the account.

3.3.5 Trading performance

We start by discovering the results derived from the threshold-based trading model during the validation period.

Table 3.7: Validation results for each clustering technique

	No clustering	OPTICS	Category
SR	5.71	4.21	4.89
ROI (%)	15.09	11.33	11.25
MDD (%)	-0.88	-1.29	-0.79
Profitable pairs (%)	95.04	90.48	87.5
Number of trades (positive-negative)	468 (415-53)	55 (47-8)	80 (63-17)

The data in Table 3.7 warrant a careful interpretation due to an inherent bias sourced from the strategy being scrutinized within the same timeframe used to pinpoint the pairs under examination, a situation not reflecting authentic trading conditions. As such, the outcomes don't accurately depict genuine performance. Although one could anticipate more gratifying outcomes, given the additional assurance on the stability of the selected pairs, the outcomes could also deteriorate, as we are unable to discard pairs that haven't yielded profits in the past due to the absence of existing records. Regardless, the validation data are instrumental in facilitating the creation of the test portfolios, which employ these findings as a heuristic for pair selection. Consequently, we opt to exhibit these data. A salient point to grasp from Table is the consistent profitability of the strategy across all scenarios, with a commendable margin. This solidifies the success of the pair selection process up to this stage.

The findings related to the unobserved data, i.e., the testing periods, are disclosed in Table 3.8 Here, we present the outcomes for every testing duration, assuming one of the three test portfolios we've developed, as outlined in Section.

Table 3.8: Test results for each clustering technique

	No clustering			OPTICS			Category		
Test Portfolio	1	2	3	1	2	3	1	2	3
SR	**2.26**	**2.27**	**2.07**	**2.87**	**2.85**	**2.61**	**1.86**	**2.08**	**1.73**

ROI (%)	**10.31**	**10.60**	**18.60**	**14.16**	**13.67**	**15.68**	**9.48**	**10.27**	**10.56**
MDD (%)	**-1.69**	**-1.78**	**-4.67**	**-2.03**	**-1.97**	**-2.54**	**-1.84**	**-1.75**	**-3.03**
Profitable pairs (%)	78.72	80.59	100	83.33	85.71	100	90.47	94.74	100
Total trades	245	229	12	42	38	18	36	34	16
Profitable Trades	184	177	12	46	33	18	30	29	15
Unprofitable trades	61	52	0	6	5	0	6	5	1

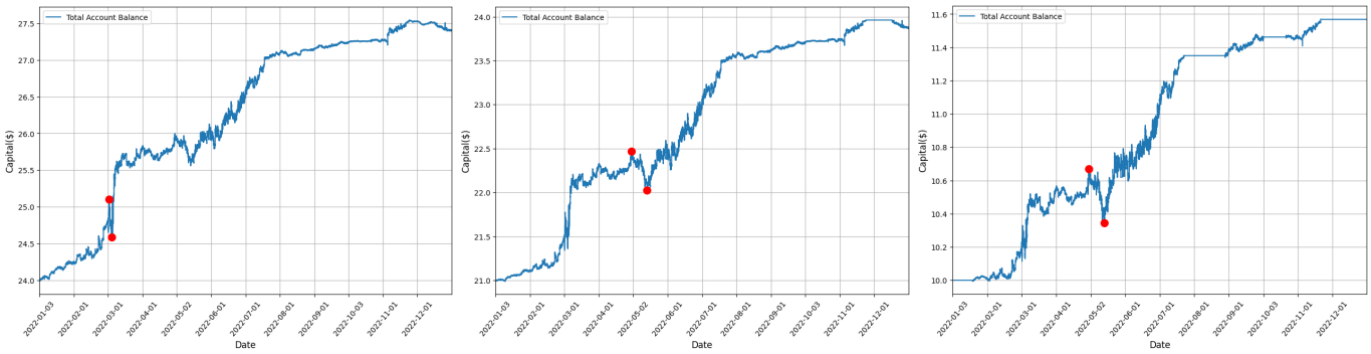
It is immediately noticeable that, in terms of pure returns, OPTICS is the method that achieves the best result. In fact, this beats the other two both in terms of ROI and in terms of Sharpe ratio. This means that whether or not the investor wants to factor in the risk associated with his investment portfolio, clustering by category or not clustering at all leads to a worse outcome. The result changes if we look at the consistency in terms of the proportion of profitable pairs within the portfolio. Here is the grouping by category that seems to have the top spot. However, it should be noted that situations where 100% of the pairs in the portfolio were profitable involved all three clustering methods.

Looking at the maximum drawdowns, it is evident that there is a certain degree of volatility when clustering by category or when not clustering at all. The data suggests that these strategies, when employing the top ten best performing pairs of the validation period (Portfolio 3), often experience a significant increase in the portfolio's maximum drawdown amplitude. In contrast, unsupervised learning strategy demonstrates its ability to maintain favorable MDD consistently across portfolios, without showing irregularities. This is validated by the lower MDD, emphasizing the stability imparted by this technique. Lastly, given that the OPTICS algorithm generally clusters pairs that appear more reliable, it is less likely to identify pairs that exhibit strong validation performance that then deteriorates in the test period. This point can be confirmed through an examination of the Sharpe ratio progression from the validation to the test results.

Interestingly, between the two conventional grouping methods, no clustering at all delivers weaker outcomes than clustering by category. This implies that without OPTICS, an investor might be better off restricting pairs to securities from the same sector rather than refraining from any grouping. On another aspect, the results reveal that there's no clear superior method for creating the test portfolio.

We conclude the chapter with a representation of the trend of the three test portfolios obtained by clustering through unsupervised learning, which is also the method that has been chosen as the basis of the forecasting-based algorithm.

Figure 3.7: Performance of the three test portfolios deriving from OPTICS following the application of the threshold-based model



a) Portfolio 1 b) Portfolio 2 c) Portfolio 3

In each graph we see how the account balance changes over time based on the transactions carried out during the reference period. The red colons represent the start and end of the Maximum Drawdown period. The differences in the starting capital are a direct consequence of the way in which the three portfolios were constructed: all pairs in the first, those with positive results during the validation period in the second and the top 10 for results obtained in the validation period in third.

4. Research Stage 2: Forecasting-based Trading

Once again we divide the chapter into three sections: (i) Forecasting-based Trading Model, (ii) Time Series Forecasting and (iii) Performance Evaluation.
In the first section we will analyze what are the motivations that prompted the authors to explore a trading model that exploits projected information. In the second section we will see which the different methods available are to obtain this information. Finally, in the third section, we will compare these methods based on their profitability.

4.1 Forecasting-based Trading Model

The threshold-based trading model, as described in Section 3.3.2, has a notable drawback. Since the only criterion for entering a position is the crossing of a pre-set threshold, regardless of the current direction of the spread, the entry points lack a precise definition. Consequently, this approach may lead to unfavorable periods of portfolio decline when a pair keeps diverging. Enhancing the trading model's robustness could involve incorporating predicted future data to determine market entry positions. By leveraging projected information, the model could make more informed decisions about when to enter trades, potentially reducing the negative impact of imprecise entry points and improving overall portfolio performance.

An interesting strategy to consider for creating a more robust trading model is the approach employed by Dunis et al. (2006). This particular method involves directly modeling the predicted returns of the spread, which the author defines as

$$R_t = \frac{Y_t - Y_{t-1}}{Y_{t-1}} - \frac{X_t - X_{t-1}}{X_{t-1}}, \tag{4.1}$$

where X_t and Y_t represent the prices of the two legs composing the pair, at time t. By doing so, the model can evaluate whether the forecasted return exceeds a predefined threshold (optimized during in-sample testing). When the predicted return surpasses this threshold, the model takes a position in the market. This approach may offer greater precision in determining entry points and potentially lead to improved trading performance.
An alternative approach to enhance the trading model's robustness is to concentrate on modeling the spread series itself and utilize this information for trading decisions. The investor can monitor recent values of the spread and search for patterns of local trendiness. When an uptrend is recognized, the investor may choose to take a long position on the spread. Conversely, if a downtrend is identified, the investor may opt to take a short position on the spread. The position can be maintained until the predicted value for the next time instant contradicts the direction of the trend, at which point the position is exited. By focusing on the

spread's behavior and trend patterns, this approach aims to provide a more reliable and adaptive trading strategy, potentially leading to improved performance in varying market conditions.

However, Sarmento & Horta (2020) demonstrates how, although theoretically appealing, both methodologies described do not work well in practice. The authors propose a third approach, which also revolves around predicting the series of spreads defined as per Eq. (4.1). In this case, the approach continuously monitors the percentage change between the spread at the current time (S_t) and the predicted spread in the next time-step (S^*_t). If the absolute value of the predicted change exceeds a predefined threshold, a position is entered. The expectation behind this decision is that the spread will experience a sudden movement from which the investor can benefit.

To elaborate more formally on the methodology, let's define St as the true value and S^*_t as the predicted value for the spread at time t. The predicted change can be calculated using Eq. 4.2 (a).

$$\text{Predicted Change} : \Delta_{t+1} = \frac{S^*_{t+1} - S_t}{S_t} \times 100 \qquad 4.2\ (a)$$

Once the spread's predicted change is computed, the position settlement conditions are applied as described in Eq. 4.2 (b). Specifically, α_L and α_S represent the long and short position trigger thresholds, respectively. When a position is entered, it is held as long as the predicted spread direction persists and is closed once there is a shift in the predicted direction. This approach aims to capitalize on abrupt spread movements, leading to potentially profitable trading opportunities.

$$\text{Market entry conditions} : \begin{cases} \text{if } \Delta_{t+1} \geq \alpha_L, & \text{open long position} \\ \text{if } \Delta_{t+1} \leq \alpha_S, & \text{open short position} \\ \text{otherwise,} & \text{remain outside market} \end{cases} \qquad 4.2\ (b)$$

The definition of the thresholds (α_L, α_S) has not been discussed thus far. One potential method involves formulating an optimization problem to determine the profit-maximizing values. However, this approach is not considered suitable due to the risks of data-snooping and the introduction of unnecessary complexity. Instead, Sarmento & Horta (2020) suggests a more straightforward, non-iterative, data-driven. For each spread series, we first obtain the distribution *f(x)* of the spread percentage changes during the formation period. The spread percentage change at time t is defined as $x_t = ((S_t - S_{t-1}) / S_{t-1}) * 100$. From this distribution *f(x)*, the positive and negative percentage change values are considered separately, leading to two distributions: $f^-(x)$ for negative percentage changes and $f^+(x)$ for positive percentage changes.

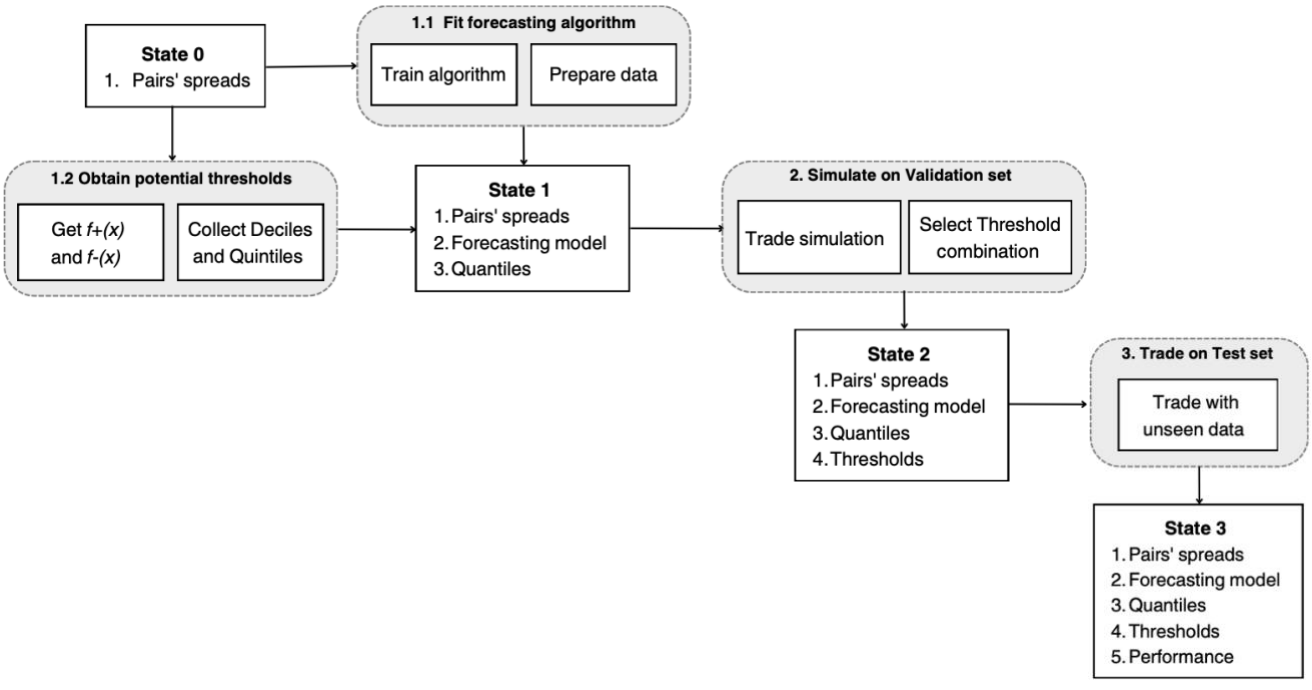
Since the proposed model aims to target abrupt changes that occur frequently enough, the authors suggest looking for extreme quantiles from the distributions. By selecting the top decile and quintile from $f^+(x)$ as candidates for defining αL and the bottom decile and quintile from $f^-(x)$ for defining α_S, the model ensures that the thresholds adapt to the spread’s volatility and properties.

Both the quintile-based and decile-based thresholds are then tested in the validation set, and the most optimistic combination is adopted. Considering both quintiles and deciles is essential as the authors fear that relying only on deciles might be too restrictive in capturing potential trading opportunities.
It is crucial to emphasize that this methodology is fundamentally different from the benchmark trading strategy presented in Sect. 3.3.2. The underlying motivation for entering a position in both models is distinct, leading to no expected relationship between the entry and exit points of the two strategies.

4.1.2 Model diagram: regression and classification

The reader is reminded that one of the modifications examined in the following study involves training the neural network within a classification framework. Specifically, the goal is to move from predicting the spread to predicting a decision to enter a position, long or short, or not to enter at all.
One of the main consequences of this change is the scheme that the investor will have to follow. In the setting of Horta & Sarmento (2020), in which what is forecasted is the spread (i.e., regression setting), the proposed model can be applied following the process represented in Fig. 4.1. To begin, the investor should train the forecasting algorithms to predict the spreads accurately. Subsequently, the decile-based and quintile-based thresholds should be gathered to incorporate into the trading model. Once the forecasting algorithms have been fitted, and the two combinations of thresholds are obtained (State 1), the model can be applied to the validation set. Based on the validation performance, the best threshold combination is selected (State 2). At this stage, the model is fully prepared to be applied to unseen data, allowing the investor to assess its performance effectively (State 3). By following this step-by-step process, the model can be confidently deployed in real-world trading scenarios while ensuring its reliability and effectiveness in making predictions on new data.

Figure 4.1: Proposed model application diagram



In the case of classification setting, the scheme changes. As anticipated, the algorithm will no longer have to be trained to predict future spreads but will have to directly predict what the investor should do at a given moment. So, the first thing the investor needs to do is collect the decile-based and quintile-based thresholds. To do this Eq. 4.2a will no longer be needed. In fact, it will be sufficient to calculate the percentage variation of the spread during the formation period. Based on this information he can now label the data on which the model will be trained: 1 for a long position, -1 for a short position and 0 for no position. Once the model has been fitted and the forecasts obtained, one proceeds as before: the investor applies the trading model to the validation set and, based on the results obtained from this, does the same with the test set.

4.2 Time Series Forecasting

Driven by the potential of implementing a trading strategy based on time series forecasting, we explore several alternatives that might be better suited for this purpose. Time series forecasting models can be broadly categorized into two primary classes: parametric and non-parametric models.

Parametric models were some of the early time series prediction methods, assuming that the underlying process follows a specific structure that can be described using a limited number of parameters. The primary objective of these approaches is to estimate the model's parameters that best describe the stochastic process. However, these models have limitations in their representation capacity, as highlighted in the work by Chen et al. (1997). Due to these limitations, there is a need to explore alternative modeling approaches that do not rely on specific structural assumptions.

Non-parametric modeling is particularly attractive because it does not impose any specific structural assumptions about the underlying process. Artificial Neural Networks (ANNs) are an example of non-parametric models. There are several reasons why ANNs are considered for this context. First, ANNs have received attention in various fields, making them an interesting case study for this application. Second, ANN-based models have shown promising results in predicting financial time series data in general, as shown by Cavalcante et al. (2016). Finally, the field of financial time series forecasting using Machine Learning techniques is relatively unexplored, as discussed in Sect. 2.4, which encourages the application of ANNs in this work.

In conclusion, Sarmento & Horta (2020) proposes applying a benchmark parametric approach alongside other relevant non-parametric models. The details of each of the proposed algorithms will be elaborated in the following paragraphs of the study. This approach aims to leverage the strengths of both parametric and non-parametric models to achieve accurate and robust time series forecasting for the trading strategy.

4.2.1 Autoregressive Moving Average

Si & Yin (2013) points out that financial time series are inherently complex, highly noisy, and nonlinear, which makes Artificial Neural Networks (ANNs) an attractive choice for forecasting due to their ability to address such conditions. However, the financial time series under analysis in this study may not necessarily exhibit the characteristics mentioned above. The spread series is constructed to be stationary during the formation period, as a pair is only selected if both constituents are cointegrated, ensuring stationary spreads. Though stationarity cannot be guaranteed during the trading period (unseen data), the series is more likely to maintain some level of stationarity. Given this observation, Sarmento & Horta (2020) questions whether a simpler model could perform the forecasting task just as effectively. The authors are interested in determining whether the added complexity of neural networks is warranted. To explore this, they propose the application of an autoregressive moving average (ARMA) model for forecasting.

The ARMA model, introduced by Whitle (1951), describes a stationary stochastic process as a combination of two polynomials. The first polynomial, known as autoregression (AR), is designed to model the variable at time t based on its historical lagged values. The second polynomial, referred to as moving average (MA), models the prediction error as a linear combination of past error terms and the expected value of the time series.

We will now provide a formal description of how each of the polynomials models a time series X_t. The AR(p) model is defined as follows:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t, \tag{4.3}$$

where p is the polynomial order, φ_1 , …, φ_p are the model parameters, c is a constant, and ε_t is a random variable representing white noise. The MA(q) model is defined as

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}, \tag{4.4}$$

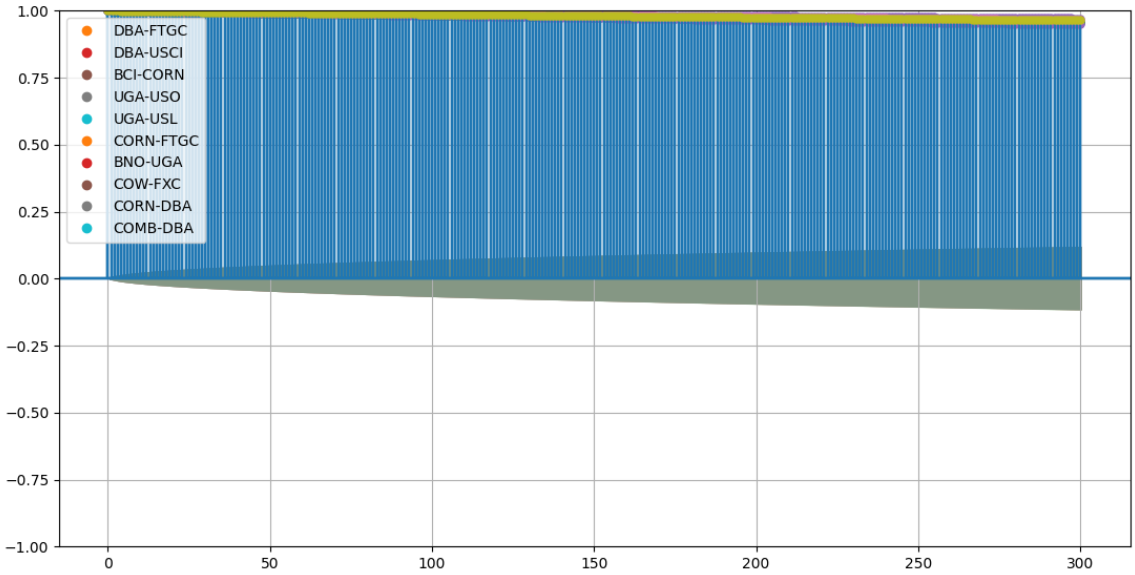
where q is the polynomial order, θ_1, …, θ_q represent the model parameters and μ represents the mean value of X_t. The variables ε_t, ε_{t-1}, …, ε_{t-q} correspond to white noise error terms at the respective time instants. Finally, the ARMA(p, q) model results from the combination of (4.4) and (4.5), and can be represented as:

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}. \tag{4.5}$$

The performance of the ARIMA model relies on the selection of the appropriate values of the autoregressive component (AR) and moving average (MA) parameters. Indeed, these determine the complexity of the model, the stationarity of the data and the residual dependence.

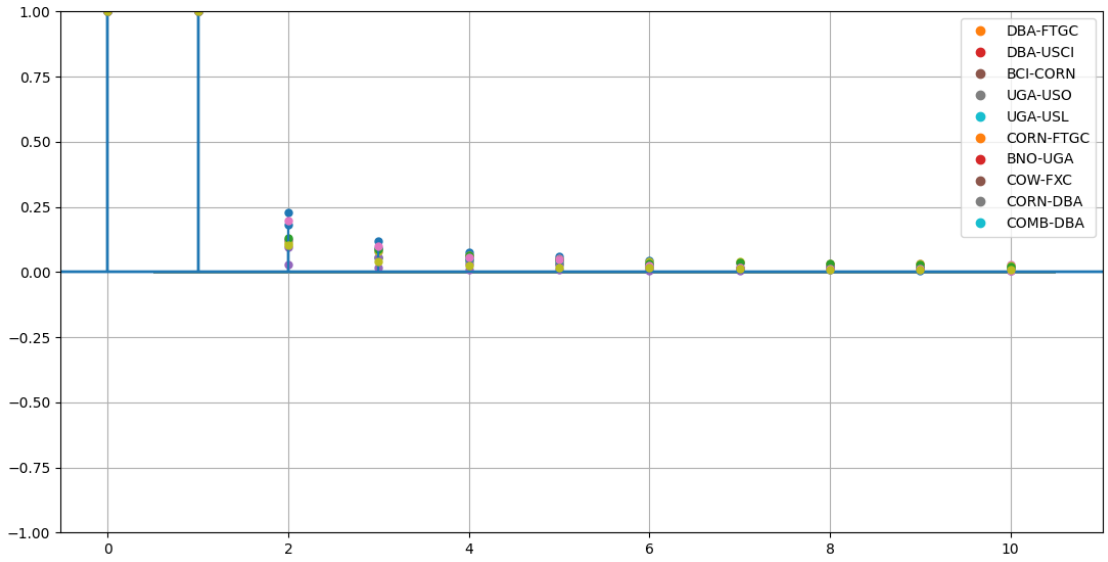
To determine the order of the AR component (p), we use the autocorrelation function (ACF) plot. The ACF measures the correlation between a time series and its lagged values. A significant correlation at a specific lag suggests a relationship between the time series and its lagged values at that lag. If this correlation decays slowly as the lag increases, an AR model may be suitable. The order of p is determined by the largest significant lag in the ACF plot. Figure 4.2 shows the ACF plots for the selected pairs during this research stage. It is evident that the correlation, although slowly decaying as the lag increases, remains significant for high values of p, demonstrating the fact that the historical series have a strong correlation with their past values over a wide range of time lags. While a larger p-value allows the model to capture more complex patterns in the data, it also increases the risk of overfitting.

Figure 4.2: ACF plot for selected pairs



To find the order of the MA component (q), we use the partial autocorrelation function (PACF) plot. The PACF measures the correlation between a time series and its lagged values, with the effect of intermediate lags removed. Significant partial autocorrelation at a specific lag indicates that the current value of the time series is influenced by the error at that lag, after accounting for the effect of intermediate lags. If this partial autocorrelation decays slowly as the lag increases, an MA model may be appropriate. The order of q is determined by the largest significant lag in the PACF plot. Figure 4.3 shows the PACF plots for the selected pairs during this research stage. The PACF shows a dramatic drop in partial autocorrelations after just one lag. In other words, after considering the autocorrelation of only one lag, subsequent autocorrelations become insignificant, demonstrating that previous values beyond the first lag have negligible influence on the current time series.

Figure 4.3: PACF plot for selected pairs



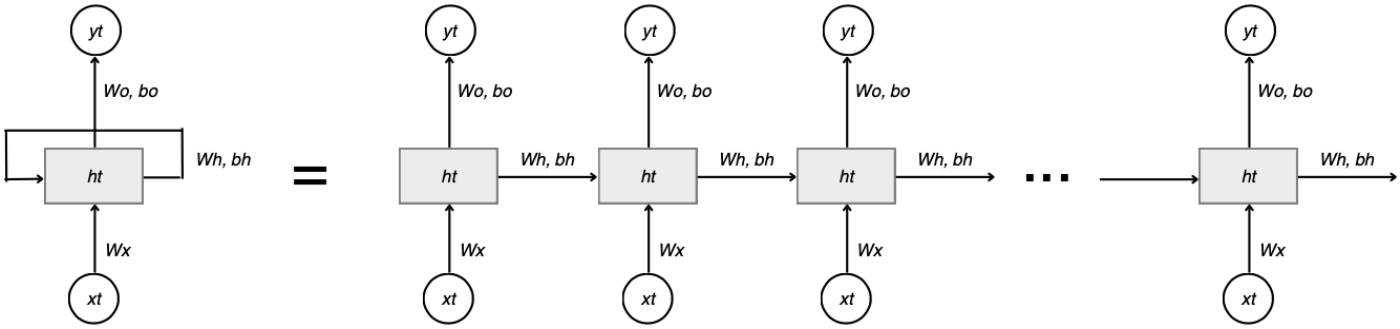
We conclude by noting how the results obtained from the ACF and PACF plots seem to indicate that when dealing with high-frequency time series, such as minute-by-minute prices, it may be necessary to opt for a non-parametric model. Indeed, fitting an ARMA model with extremely large p values can lead to an overly complex model that may not generalize well and may be sensitive to small changes in the data, making it less robust for prediction purposes.

4.2.2 Artificial Neural Network

The ARMA model's simplicity, which was discussed earlier, offers practical advantages. However, it could also be seen as a limitation when attempting to learn more intricate data patterns. In such cases, Artificial Neural Networks have demonstrated their remarkable efficiency, providing a more suitable solution.

One suitable configuration that addresses the limitation of learning intricate data patterns is the Long-Short Term Memory (LSTM) model. LSTMs belong to a category of Recurrent Neural Networks (RNNs), which differ from standard feedforward neural networks (FNNs) in their ability to process not only individual data points but also entire sequences of data, as mentioned by Goodfellow et al. (2016). In contrast to traditional FNNs, where inputs are assumed to be independent of each other, RNNs assume a sequential dependency among inputs. This implies that previous states can influence the decision-making process of the neural network at different points in time, as depicted in Fig. 4.4. RNNs achieve this functionality through inner loops, which enable them to retain a memory of their previous computations.

Figure 4.4: Structure of a Recurrent Neural Network



In Fig. 15, the symbols W_i and b_i represent the weights and bias term of the neural network, respectively. The subscript “i” can take the values of "x" for the input, "h" for hidden, and "o" for output. The grey rectangle represents the hidden layer's RNN cell, which unfolds along the time dimension. This cell takes an input and, along with the previous state, computes the next state and the output at the current time step. The two cell outputs can be described as follows:

$$\begin{cases} h_t = \sigma\left(W_h h_{t-1} + W_x x_t + b_h\right) \\ o_t = \sigma\left(W_o h_t + + b_o\right) \end{cases}, \tag{4.6}$$

where “σ” represents the activation function of the layer. One issue that arises with the RNN architecture described above is the difficulty in propagating the prediction error during the training process. To understand why, let's first describe the standard backpropagation in regular FNNs. In a feedforward neural network, the backpropagation algorithm moves backward from the final error through the outputs, weights, and inputs of each hidden layer. It calculates the partial derivatives $\partial E/\partial wi$, where "E" represents the error function, and considers these derivatives to be responsible for a portion of the error. The backpropagation algorithm then uses these derivatives to adjust the weights in a direction that decreases the error, thus enabling the network to learn from the data.

For training RNN models, the same backpropagation process applies, considering the network in its unfolded version, as shown in Fig. 4.4. However, when propagating the error along the entire network's time dimension, a problem can occur: the gradient might either vanish or explode. The vanishing gradient problem happens when the derivative of the state update function decreases the backpropagation error too drastically, making the learning process very slow. On the other hand, the exploding gradient problem occurs when the derivative increases the error too drastically, leading to numerical issues during training. These challenges in propagating the error over time can make training RNNs difficult and limit their effectiveness in certain applications. To address these problems, more advanced RNN architectures like LSTMs were introduced, as they provide a more effective way of handling long-term dependencies and mitigating the vanishing and exploding gradient issues.

The issue mentioned above is addressed, in part, by LSTMs (Long-Short Term Memory) proposed by Hochreiter & Schmidhuber (1997). LSTMs use a gating mechanism to selectively update the internal state

based on the current state and the input. This architecture involves three gates: the input gate, the output gate, and the forget gate, which collectively regulate the flow of information.
The input gate controls how much influence the new input has on the internal state. The forget gate determines which aspects of the internal state are preserved over time. Lastly, the output gate decides which part of the internal state contributes to the network's output.
In this way, LSTMs efficiently manage long-term dependencies and overcome some of the challenges associated with traditional RNNs when propagating error during training.

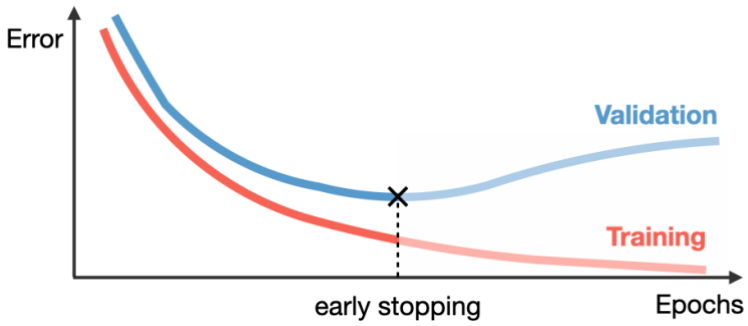
Deep Learning models are criticized for the intricate process involved in optimizing their configurations. Tuning hyperparameters significantly impacts their outcomes, but this undertaking is also exceptionally time-intensive. Thus, a balance must be struck. Given the constrained computational resources, the tuning of LSTM models is confined to a subset of the most pertinent variables: (i) sequence length, (ii) count of hidden layers, and (iii) nodes within each hidden layer. We systematically test progressively more intricate configurations until indications of overfitting become apparent.

Neural network weights are frequently initialized randomly, which might require numerous iterations to converge to the optimal values. Unfortunately, this kind of initialization is susceptible to problems like vanishing or exploding gradients. One strategy to mitigate these issues is to approach random weight initialization with careful consideration. In this study, we employ the glorot weight initialization, introduced by Glorot & Bengio (2010). This approach entails an initialization algorithm that factors in the network's size (comprising the number of input and output neurons). The proposed methodology not only decreases the likelihood of encountering gradient-related difficulties but also expedites the convergence process. This initialization technique, also referred to as Xavier initialization, involves sampling layer weights in a manner that maintains the input variance. This ensures that the variance remains consistent as information traverses through successive layers of the neural network.

Neural network topologies are typically defined by non-convex functions, as highlighted by Goodfellow et al. (2016). Consequently, the training process lacks a formal guarantee of converging to the optimal solution. To address this concern, regularization terms are introduced during the optimization process. Regularization involves imposing additional constraints, often derived from problem-specific information, to manage ill-posed problems and prevent overfitting.
In particular, regularization aids in diminishing the generalization error by narrowing the gap between errors observed in the training set and validation set. Simultaneously, by incorporating supplementary information, the convergence rate can be improved, as outlined by Goodfellow et al. (2016). This discussion now delves into two regularization techniques adopted within this study: Early-stopping and Dropout.
Early-stopping stands out as a frequently employed regularization approach in Deep Learning, favored for both its effectiveness and simplicity, as mentioned by Goodfellow et al. (2016). The essence of Early-

stopping lies in the following rationale: when training substantial models with ample representational capacity to potentially overfit, a consistent reduction in training error is usually observable. However, the validation set error begins to rise after a certain point, a trend depicted in Fig. 4.5. This pattern is particularly evident in the mentioned figure.

Figure 4.5: Illustration of early stopping



By recognizing this behavior, it becomes feasible to attain a model with superior validation set error (and ideally, better test set error). To achieve this, it is necessary to revert to the parameter configuration associated with the lowest validation error, indicated by the black triangle in Fig. 16. This can be achieved by maintaining a memory copy of the model parameters that yielded the best validation error up to that point. At the conclusion of the training process, the algorithm retrieves these stored parameters instead of the most recent ones. This process can be expedited if the training procedure is terminated when no enhancement in validation loss is detected for a predefined number of consecutive iterations following the best parameterization; this termination mechanism is often denoted as "patience."
The second regularization technique employed is Dropout. Its fundamental concept revolves around the notion that the risk of overfitting can be mitigated by training distinct neural networks on the same dataset and then using the average prediction of all these networks as the final prediction. This approach is rooted in the belief that noise learned by a specific model architecture is attenuated when averaged across multiple models. However, training numerous models simultaneously can be intricate, especially when dealing with intricate models. Dropout presents a more pragmatic alternative for emulating this process. It approximates training numerous neural networks with diverse architectures in parallel while effectively training only one neural network. This is achieved by randomly disregarding certain layer nodes during training, effectively treating layers as if they possess varying node counts and connectivity to the prior layer. This dynamic perspective adjustment during training imbues each update with a distinct layer configuration.

We conclude the paragraph by seeing the differences in the architecture of the model passing from a classification setting to a classification one.
In the original implementation of the model, that of Sarmento & Horta (2020), the neural network is optimized for regression problems. Specifically, the model uses a mean squared error (MSE) loss function, which aims to minimize discrepancies between predicted and actual continuous target values. The

architecture of the network and its final output level have been designed to produce a single continuous value, corresponding directly to the target or, in other words, to the spread. The model's performance was then evaluated based on metrics such as mean absolute error, which measured the average magnitude of errors made by the model.

In our case, however, that of classification, the target variable becomes categorical, with three distinct classes represented as -1 (short position), 0 (no position) and 1 (long position). This required several changes:

- The target series was subjected to one-hot coding[12], translating the three discrete values into a binary matrix format suitable for classification.
- The model architecture was updated, especially its final level, which now needed to provide probabilities for three distinct classes rather than a singular value.
- The loss function has been moved from MSE to categorical cross-entropy, which is more suitable for classification tasks.

In terms of evaluating model performance, accuracy, has become a primary metric, highlighting the proportion of correctly classified instances to the total. For transparency, precision[13] and recall[14] will also be exposed along with this.

After generating the predictions, an extra step was added to convert the one-hot encoded outputs back to the original labels (-1, 0, 1). This ensured that the results remained interpretable and consistent with the categorical goals of the study. These changes moved the framework from its original regression focus to a classification-centric approach, meeting evolving research needs."

4.3 Performance Evaluation

This section is dedicated to the analysis of the outcomes that enable us to address the second research inquiry: "Is it feasible for a forecasting-based trading model to attain enhanced and consistent performance?" To address this, we assess the effectiveness of the introduced forecasting-based model by juxtaposing its performance with that of the threshold-based model employed thus far in our simulations. This analysis systematically traces the simulation's evolution, commencing with pairs identification and extending through to the actual trading process.

---

[12] One-hot encoding is a fundamental technique in data preprocessing within the field of machine learning and data science. It is used to represent categorical variables as binary vectors, where each category is transformed into a unique binary code. This encoding allows algorithms to process categorical information, which is typically not directly compatible with mathematical operations, in a format that can be effectively utilized for predictive modeling and analysis. Each binary digit in the one-hot encoded vector corresponds to a specific category, and the presence of '1' in a particular position indicates the category to which the observation belongs. One-hot encoding ensures that the categorical data is properly prepared for input into machine learning algorithms, contributing to more accurate and reliable model outcomes.

[13] Precision is a metric that measures how often a machine learning model correctly predicts the positive class. You can calculate precision by dividing the number of correct positive predictions (true positives) by the total number of instances the model predicted as positive (both true and false positives).

[14] Recall is a metric that measures how often a machine learning model correctly identifies positive instances (true positives) from all the actual positive samples in the dataset. You can calculate recall by dividing the number of true positives by the number of positive instances. The latter includes true positives (successfully identified cases) and false negative results (missed cases).

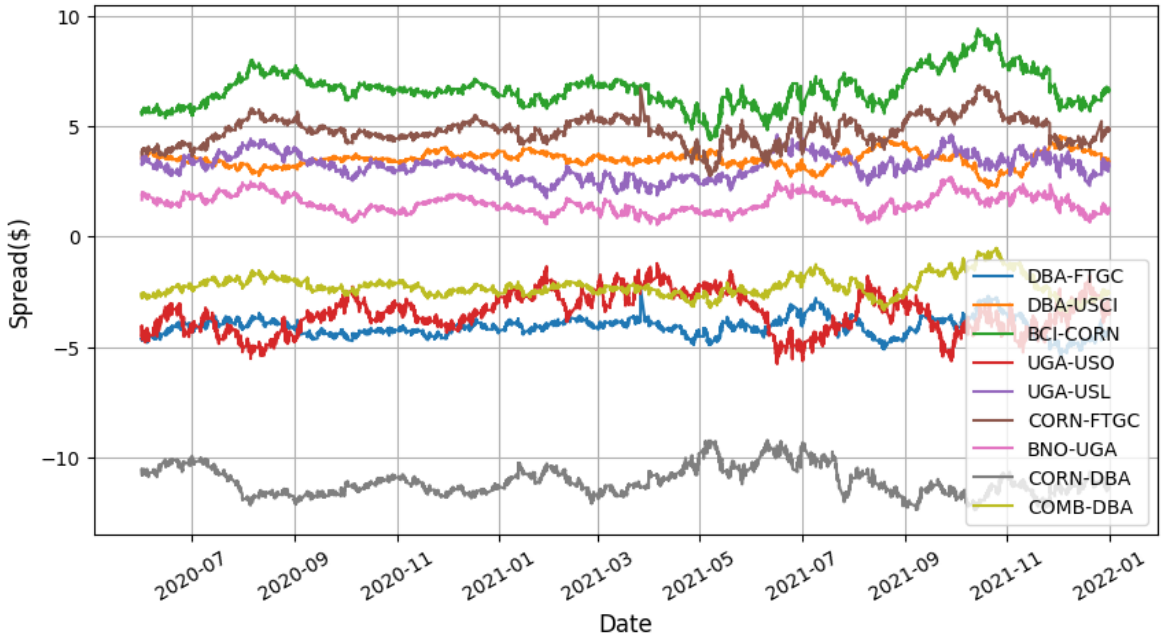
4.3.1 Eligible Pairs

We begin by outlining the pairs that were chosen during this phase of the research.
To address the potential high cost associated with comparing prediction-based models through three distinct clustering techniques, a more pragmatic strategy is adopted: among the clustering methodologies evaluated, the one that showed the most promising results was chosen. Therefore, for this particular scenario, we exclusively use the OPTICS clustering technique to rank potential pairs. This decision is backed by its proven ability to build strong portfolios. After the clustering process, we proceed with the conventional pair selection procedure, similar to our previous methodology. As anticipated in paragraph 3.2.1, of the 96 couples identified by the OPTICS, 21 passed all the steps of the selection process. Among these we have chosen the top ten best performing pairs of the validation period (referred to as Portfolio 2 in Fig. 3.5). 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 identified ETFs, one by one. Table lists the selected pairs, along with the corresponding segment, and their description. Both the segment and the description information are retrieved from ETF.com.

An intriguing observation is that among the ten pairs, six belong to distinct categories. To gain deeper insights into this phenomenon, we conduct a more comprehensive analysis of the DBA-FTGC pair. Given that FTGC is categorized as a Broad Market ETF, tracking the overall commodity market, we delve into additional data from ETF.com that outlines its sector composition. Upon examining Table, it becomes apparent that Agriculture serves as the predominant driver for FTGC. Consequently, the equilibrium between FTGC and DBA can be attributed to their significant exposure to the same sector. Any deviations from this equilibrium might potentially find explanation in market fluctuations within the remaining sectors.

We conclude the paragraph by analyzing the trend of the spreads of the pairs during the formation period, represented in the Figure 4.6.

Figure 4.6: Spread from the ten pairs identified during the formation period



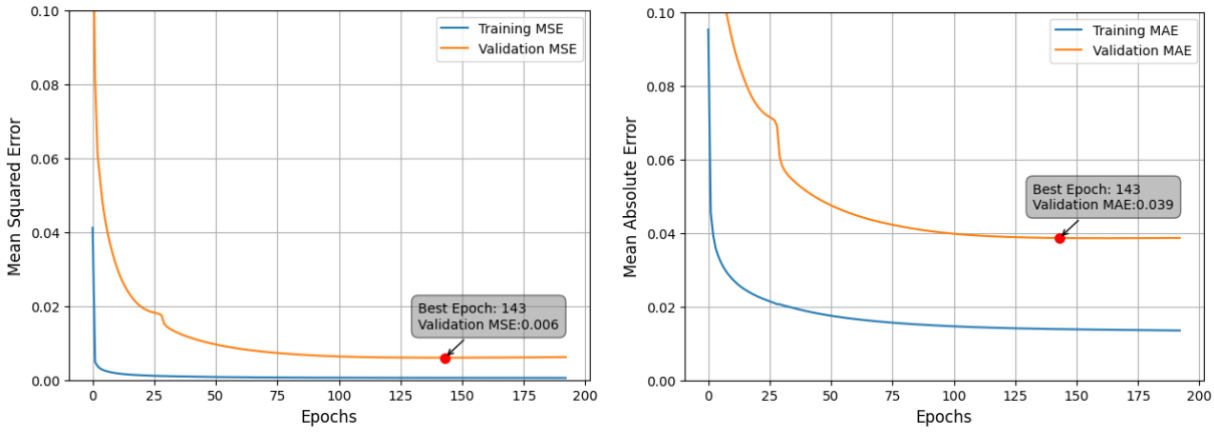
An intriguing observation is the apparent stationarity of the spreads, aligning with expectations. Additionally, a distinct variance exists among the various time series' volatilities. This contrast further reinforces the notion that data-oriented model parameters, such as the suggested thresholds for the predictive trading model, appear well-suited within these circumstances.

4.3.2 Forecasting performance

Before proceeding with the analysis of the forecast results obtained, it is important to note that while both the parametric and non-parametric approaches utilize the formation period for training, there are subtle differences in the methodology for each approach. Specifically, the ARMA model leverages the entire formation period to acquire the optimal combination of coefficients that minimizes forecasting errors. In contrast, the LSTM exclusively utilize the training period for this purpose. In the latter case, a distinct validation set is employed to effectively manage and prevent overfitting.

To counter overfitting, the system employs an early-stopping strategy, detailed in Section 4.2.2. This involves continuous monitoring of the validation score, and if indications of overfitting become apparent, the training process is promptly halted. To provide a visual depiction, Figure illustrates a prototypical training scenario using an example from the LSTM, demonstrating its application in fitting the spread generated by the DBA-FTGC pair.

Figure 4.7: LSTM training loss history for DBA-FTGC spread



In Figure 4.7 (a), the evolution of training and validation Mean Squared Error (MSE) can be observed. As anticipated, both metrics decrease as the training advances. Notably, the algorithm appears to reach its optimal configuration around epoch 143. Beyond this point, there is a lack of improvement in validation MSE, despite the ongoing reduction in training MSE. This behavior strongly suggests overfitting. It's important to clarify that the figure actually encompasses a total of 193 epochs, although this entirety might not be visually evident. The early-stopping mechanism is set with a patience of 50 epochs, thus allowing training to continue for 50 additional epochs. Consequently, the training concludes at the 193rd epoch due to the absence of enhanced scores during this period.
Furthermore, attention is drawn to the adjacent image, Figure 4.7 (b), which portrays the evolution of Mean Absolute Error (MAE). Despite the model's primary focus on minimizing MSE, the progression of MAE exhibits a comparable pattern.

We can now focus on the results from the best performing architectures of the models employed. Clarification is essential regarding the definition of the “best architectures”. In our context, these architectures are identified based on their ability to minimize validation mean squared error. The rationale for utilizing the validation score lies in its similarity to the information accessible to an investor before commencing trading – a critical decision juncture. It's worth noting that mean squared error serves as the designated metric for comparison since it constitutes the loss function targeted for minimization by the models.
Aligned with these criteria, the optimal performing architectures are detailed in Table 4.1. The outcomes attributed to each model pertain to the average forecasting performance over the forecasts of the ten distinct spreads mentioned above.

Table 4.1: Forecasting results comparison

	Configuration	Parameters	Test	Validation

					MSE (E-03)	RMSE (E-02)	MAE (E-02)	MSE (E-03)	RMSE (E-02)	MAE (E-02)
ARMA	p: 30	q: 1		31	0.4	1.86	0.96	1.5	3.38	1.77
LSTM	input length: 72	hidden layers: 1	hidden nodes: 50	10651	0.3	1.73	1.14	1.3	3.39	2.08

Table 4.1's results clearly show the LSTM model's superiority over the ARMA across the specified metrics. These findings bring into focus the appropriateness of using ARMAs given the study's conditions. Existing literature has pointed out some of ARMA's limitations in high-frequency data scenarios. Notably, the stationarity assumption and the effects of microstructure are primary concerns. Harris et al. (2013) noted that the foundational stationarity assumption integral to ARMA models might be unsuitable for high-frequency financial data, which often displays volatility clustering and variable mean and variance. Such discrepancies can adversely affect model performance and forecasting accuracy. Additionally, microstructure effects, including factors like bid-ask spreads and market liquidity, can distort ARMA estimates, making them less adept at deciphering data dynamics (Gatheral et al., 2010). Nevertheless, it is crucial to note that we accounted for both of these scenarios in this study. Section 1.1 delineates the extensive criteria implemented to guarantee the stationarity of the identified pairs. Meanwhile, Section 1.2 elaborates on our methodology for excluding less liquid ETFs, specifically to sidestep the aforementioned complications. Yet, the consistently significant parameter p in the ARMA model remains a point of contention. As explored in Section 1.3, even at high values, the significance of parameter p remains, potentially pointing to subpar modeling results. Such behavior might hint at long-memory or heavy-tailed patterns in the high-frequency data, traits not rare in financial time series (Cont, 2001; Lillo & Farmer, 2004). This continued significance for high p values might indicate the model's difficulty in encapsulating complex patterns beyond basic autoregressive relationships. Exploring more intricate models or other approaches, like GARCH-based extensions, could offer solutions to enhance predictive accuracy (Bollerslev, 1986; Engle, 1982). Hence, while your meticulous attention to stationarity and microstructure noise is praiseworthy, the pronounced significance of the parameter p suggests a need for more in-depth data analysis to bolster the ARMA model's efficacy in high-frequency financial contexts.

The reader may have noticed that Table 4.1 lacks neural network results in the classification setting. This is because the metrics used in this setting are not comparable to the regression one. As anticipated in the previous section, accuracy, precision and recall were used to evaluate the predictive performance of the neural network in the classification setting.

Table 4.2: Forecasting results of LSTM in a classification setting

	Configuration			Parameters	Test			Validation		
					Accuracy	Precision	Recall	Accuracy	Precision	Recall
LSTM	input length : 72	hidden layers : 1	hidden nodes : 50	10753	0.98	0.97	0.98	0.96	0.93	0.96

Here it is emphasized that the results presented in the table are only relatively good as the dataset on which the algorithm was trained and then tested is highly unbalanced. In fact, of all signals, only 2% are long or short. The rest consists exclusively of no-entry ones. To judge the predictive capabilities of the model we must therefore wait for the next section.

4.3.3 Trading performance

Let us begin by revisiting the core motivation behind adopting the forecasting-based pairs trading approach – the intention to enhance portfolio resilience by minimizing the instances in which the portfolio experiences declines in value. With the aim of assessing the efficacy of this proposed methodology, we proceed to outline the outcomes achieved through the employment of the forecasting-based trading model utilizing various forecasting algorithms. This assessment covers the period spanning January 2022 to December 2022. The obtained results, as outlined in Table 6.12, are juxtaposed with the outcomes derived from utilizing the conventional threshold-based model.

It's important to emphasize once again that the presentation of results in the validation set serves primarily for comprehensive reporting. Its core purpose, however, lies in identifying pairs capable of generating profits during the validation period, rather than serving as a test of the strategy itself.

Table 4.3: Trading result comparison

Trading Model	Threshold-base model	ARMA-based model	LSTM-based model	
			Regression	Classification
Parameters	See Table 3.4	$\alpha_S = Q_{f-(0.10)}$ $\alpha_L = Q_{f+(0.90)}$	$\alpha_S = Q_{f-(0.10)}$ $\alpha_L = Q_{f+(0.90)}$	$\alpha_S = Q_{f-(0.10)}$ $\alpha_L = Q_{f+(0.90)}$
Validation				
SR	4.32	-0.13	2.85	-1.2
ROI (%)	11.33	-0.12	9.39	-1.98
MDD (%)	-1.29	-1.91	-1.54	-3.03
Days of decline	86	48	54	40
Trades (Pos.-Neg.)	65 (57-8)	90 (21-69)	105 (72-33)	102 (19-83)
Profitable pairs	16	3	9	7
Test				
SR	2.85	0.34	0.59	-1.32
ROI (%)	13.67	0.53	7.13	0.18

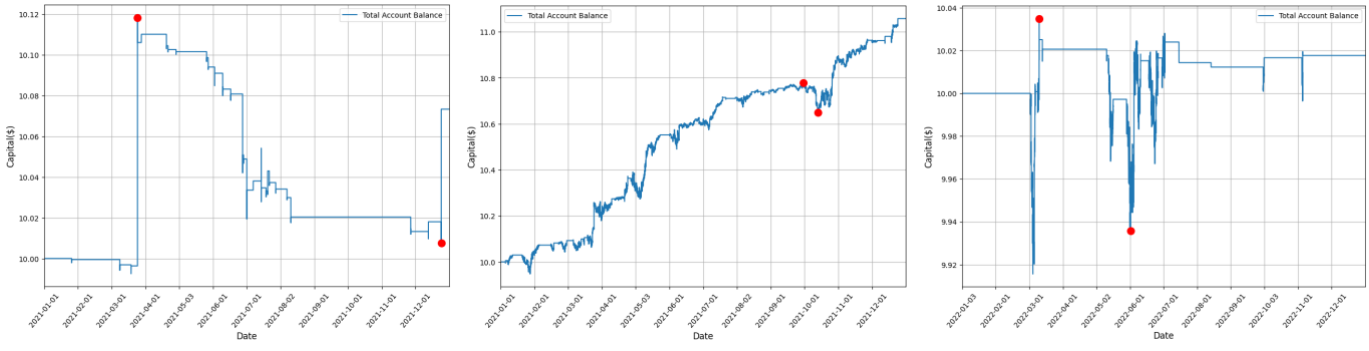
MDD (%)	-2.03	-1.09	-5.25	-0.99
Days of decline	94	21	68	57
Trades (Pos.-Neg.)	38 (33-5)	42 (11-31)	207 (129-78)	49 (19-30)
Profitable pairs	15	2	5	2

Directing our attention to the test results, which mirror a real-world trading scenario, we can affirm that the period of portfolio decline is notably shorter when employing the forecasting-based models, in contrast to the standard trading model. The longest decline period associated with the LSTM in the regression setting covers 71 days, which is still 25% less than the total decline period observed with the standard trading model of 87 days. However, it's crucial to note that this reduction in decline duration comes at a cost. There is a trade-off in terms of performance metrics, with the forecasting-based models demonstrating lower total returns on investment and portfolio Sharpe ratios in comparison to the standard trading model. However, it should be underlined that this performance is poor only in relative terms since, as regards return on investment and Sharpe ratio of the portfolio, our LSTM model far exceeds the best results obtained by the threshold-based one of Sarmento & Horta (2020). This result also allows us to validate another of the hypotheses advanced at the beginning of this study, namely that the forecasting-based model performance increases when reducing the formation period.

Shifting our focus to a different aspect, as we had anticipated based on the forecasting results outlined in Section 4.3.2, it becomes evident that the ARMA-based trading model fall short of surpassing the LSTM-based one in terms of overall performance, at least regarding the regression setting. In fact, the same cannot be said of the case of classification in which the absence of a fairly balanced database would seem to have precluded the performance of the model. Therefore, it remains an intriguing avenue for future research to explore how an LSTM model would behave when trained on more volatile time series.

We conclude the chapter with a representation of the performance of Portfolio 2 for each of the trading models implemented. As in Section 3., in each graph we see how the account balance changes over time based on the transactions carried out during the reference period.

Figure 4.8: Performance of Portfolio 2 following the application of the forecasting-based models used



a) ARMA b) LSTM (Regression) b) LSTM (Classification)

5. Conclusions and Future Work

In this last section we revisit how the modifications explored in this study allowed us to validate and extend the results obtained by Sarmento & Horta (2020).

Starting with Research Stage 1, having used higher frequency data allowed us to reduce the formation period while maintaining the same level of data points, which had as its main consequence the possibility of finding more profitable pairs and improving the results of the threshold-base model. In particular, for an investor singularly focused on maximizing returns, regardless of the accompanying risk, the exploration of all possible pair combinations emerges as an attractive prospect. However, when risk enters the equation, the strategy rooted in OPTICS demonstrates superior promise. Specifically, the OPTICS-based approach exhibits the capacity to yield the highest average portfolio Sharpe ratio, quantifying to 2.85. By contrast, this outperforms the alternatives, which yield ratios of 2.27 when no clustering is applied and 2.08 when categorization is employed. Additionally, the OPTICS strategy showcases a more stable trajectory in terms of portfolio drawdowns. It manages to maintain Maximum Drawdown (MDD) values within acceptable bounds, even when the other two techniques exhibit significant deviations. Consequently, in addressing the initial research question, it is reasonable to conclude that Unsupervised Learning has the potential to identify pairs with greater promise, particularly when risk is factored into the equation.
However, the expansion of the sectors from which the ETFs were chosen has not had the same success. In fact, in none of the pairs selected from the clusters identified by OPTICS was a currency-linked ETF present. It would therefore be interesting to see the behavior of the algorithm compared to another sector, such as Equity. It would therefore be interesting to see the behavior of the algorithm compared to another sector such as Equity[15], composed of a higher number of ETFs, all highly liquid.

We now come to Research Stage 2. The findings suggest that when assessing robustness in terms of the duration in which the portfolio value remains stable, the proposed trading model offers an enhancement, , confirming what was seen in Sarmento & Horta (2020). Within the timeframe spanning from January 2022 to December 2022, the LSTM-based model (in the regression setting) exhibits a total of 68 days of portfolio decline. In contrast, the standard threshold-based model records a significantly higher number of 94 such days. However, it's important to note that this increased robustness comes at the expense of reduced overall portfolio profitability. Neverthless, it should be underlined that the model far surpasses that of Sarmento & Horta (2020) which, once again, is due to the reduction of the timeframe following the increase in frequency. The same cannot be said for the LSTM-based model in the classification setting or for the ARMA-based model. In the first case, the more efficient optimization process was overshadowed by a highly unbalanced dataset. Here the possibility of using the equity-linked ETFs mentioned above could come in handy. Since

[15] ETF.com defines the Equity category as the set of ETFs that have the stated purpose of providing exposure to a set of companies, drawn from the broad market, grouped and defined by sources of revenue.

these are more volatile, in fact, the number of long and short positions that can be derived from the respective price series could increase, therefore reducing the model's biases. In the second case, however, we are faced with a limit of the model. Although we controlled for series stationarity and microstructure effects, the model appears to have difficulty encapsulating complex patterns beyond basic autoregressive relationships. Exploring more complex models or other approaches, such as GARCH-based extensions, could therefore offer solutions to improve predictive accuracy.

Bibliography

Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative value arbitrage rule. Yale ICF Working Paper No. 08-03.

Chan, E. (2013). Algorithmic trading: Winning strategies and their rationale (1st ed.). Wiley Publishing.

Russell, S. J., & Norvig, P. (2016). Artificial intelligence: A modern approach. Pearson Education Limited.

Bostrom, N. (2014). Superintelligence: Paths, dangers, strategies. Oxford University Press.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210–229.

Bishop, C. M. (2006). Pattern recognition and machine learning. New York: Springer.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT Press.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521, 436-444.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Neural Information Processing Systems, 25.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 5998-6008.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Chen, H., Chen, S., Chen, Z., & Li, F. (2017). Empirical investigation of an equity pairs trading strategy. Management Science.

Perlin, M. (2007). M of a kind: A multivariate approach at pairs trading. MPRA Paper 8309, University Library of Munich, Germany.

Chen, Y., Rogoff, K., & Rossi, B. (2010). Can exchange rates forecast commodity prices? Quarterly Journal of Economics, 125(3), 1145-1194.

Frankel, J. A. (2008). The effect of monetary policy on real commodity prices. In Asset Prices and Monetary Policy (pp. 291-327). University of Chicago Press.

Do, B. H., & Faff, R. W. (2011). Are pairs trading profits robust to trading costs?

Avellaneda, M., & Lee, J. H. (2010). Statistical arbitrage in the US equities market. Quantitative Finance, 10(7), 761-782.

Bellman, R. (1966). Dynamic programming. Science, 153(3731), 34-37.

Berkhin, P. (2006). A survey of clustering data mining techniques. In Grouping multidimensional data (pp. 25-71). Springer.

Sneath, P. H. A. (1957). The application of computers to taxonomy. Journal of General Microbiology, 17, 201-226.

Patel, A. A. (2019). Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data. O'Reilly.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability (Vol. 1, pp. 281-297).

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20, 53-65.

Johnson, S. C. (1967). Hierarchical clustering schemes. Psychometrika, 32, 241-254.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise.

Ankerst, M., Breunig, M. M., Kriegel, H. P., & Sander, J. (1999). Optics: Ordering points to identify the clustering structure. ACM Sigmod Record, 28, 49-60.

Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. arXiv:1302.6613.

Engle, R., & Granger, C. (1987). Cointegration and error correction: Representation, estimation, and testing. Econometrica, 55, 251-276.

Armstrong, J. S. (2001). Principles of forecasting: A handbook for researchers and practitioners (Vol. 30). Springer Science & Business Media.

Hurst, H. E. (1951). The long-term storage capacity of reservoir. Transactions of the American Society of Civil Engineers, 116, Paper 2447, Published in 1950 as Proceedings-Separate No. 11.

Bouchaud, J. P., et al. (2018). An introduction to statistical finance. Cambridge University Press.

Cont, R., & Tankov, P. (2003). Financial modelling with jump processes. Chapman and Hall/CRC.

Göncü, A., & Akyıldırım, E. (2016). Statistical arbitrage with pairs trading. International Review of Finance, 16(2), 307-319.

Huang, C. F., Hsu, C. J., Chen, C. C., Chang, B. R., & Li, C. A. (2015). An intelligent model for pairs trading using genetic algorithms. Computational Intelligence and Neuroscience, 2015, 16.

Dunis, C. L., Giorgioni, G., Laws, J., & Rudy, J. (2010). Statistical arbitrage and high-frequency data with an application to Eurostoxx 50 equities. Liverpool Business School, Working paper.

Rad, H., Low, R. K. Y., & Faff, R. (2016). The profitability of pairs trading strategies: distance, cointegration, and copula methods. Quantitative Finance, 16(10), 1541-1558.

Huck, N., & Afawubo, K. (2015). Pairs trading and selection methods: Is cointegration superior? Applied Economics, 47(6), 599-613.

Do, B., & Faff, R. (2010). Does simple pairs trading still work? Financial Analysts Journal, 66(4), 83-95.

Lo, A. W. (2002). The statistics of Sharpe ratios. Financial Analysts Journal, 58(4), 36-52.

Sharpe, W. F. (1966). Mutual fund performance. The Journal of Business, 39, 119-138.

Jolliffe, I. (2011). Principal component analysis. Springer, Berlin.

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., & Oliveira, A. L. (2016). Computational intelligence and financial markets: A survey and future directions. Expert Systems with Applications, 55, 194-211.

Chen, G., Abraham, B., & Bennett, G. W. (1997). Parametric and non-parametric modeling of time series—an empirical study. Environment: The International Journal of Environmental Society, 8(1), 63-74.

Si, Y. W., & Yin, J. (2013). OBST-based segmentation approach to financial time series. Engineering Applications of Artificial Intelligence, 26(10), 2581-2596.

Whitley, P. (1951). Hypothesis testing in time series analysis, vol. 4. Almqvist & Wiksells, Stockholm.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press, Cambridge.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).