Università degli Studi di Cagliari

Ph.D. DEGREE

Matematica e Informatica

Cycle XXXIV

ARTIFICIAL INTELLIGENCE DRIVEN SYSTEMS FOR FINANCIAL FORECASTING

Scientific Disciplinary Sector(s)

INF/01 -INFORMATICA

Ph.D. Student: Maria Mădălina Stanciu

Supervisors: Prof. Salvatore Carta
Prof. Diego Reforgiato Recupero

Final exam. Academic Year 2020/2021
Thesis defence: April 2022 Session

Abstract

In recent years, machine learning algorithms have been successfully employed to leverage the potential of identifying hidden patterns of financial market behavior and, consequently, have become the norm in financial applications.

This thesis proposes a statistical arbitrage trading strategy with two key elements: an ensemble of regression algorithms for asset return prediction, followed by a dynamic asset selection. More specifically, the extreme heterogeneity of the ensemble is achieved by ensuring *model diversity* by using state-of-the-art machine learning algorithms, *data diversity* by using diverse input features, and *method diversity* by using individual models for each asset, as well as models that learn cross-sectional across multiple assets. Moreover, the ensemble is constructed using a novel model selection approach. Then, the predicted results are fed into a quality assurance mechanism that prunes assets that have poor forecasting performance in recent history.

However, relying blindly on machine learning algorithms for decision-making can have negative consequences, especially in critical areas such as finance. At the same time, it is well acknowledged that converting data into actionable insights can be a complex task. As a particular example, the practitioners in the financial domain, find it difficult to manage large quantities of data linked to an impressive number of stocks. Given these motivations, this dissertation introduces machine learning approaches based on eXplainable Artificial Intelligence techniques that are integrated into a financial forecasting and trading pipeline. Specifically, there are presented three strategies for excluding irrelevant features for the prediction task, with the goal being to increase the prediction performance not only at the stock level but also globally, at the stock-set level. For the proposed trading strategies, the analysis that was carried out reveals that the use of the feature selection approaches improve the portfolio performance. This is achieved by using only the predictive signals which are less noisy than the original content of the entire feature set while preserving enough information.

To demonstrate the utility of the proposed approaches, their performance is evaluated in real-world scenarios, *i.e.*, historical data of stocks composing the S&P500 index or custom stock-set combining various other indexes.

The thesis contributes to the scientific literature in terms of results and

the novel manner of exploiting machine learning in an algorithmic trading context.

Acknowledgements

I would like to thank everyone that has helped shape the ideas of this thesis by either sharing their technical insights or by supporting and encouraging me to keep on going.

I had the great opportunity to be guided and supervised by Prof. Salvatore Carta and Prof. Diego Reforgiato Recupero. I would like to express my deep gratitude to both of them as this thesis would not have been possible without their constant support, ideas, suggestions, and advice. During these four years, I have learned many things from them, among which, how to ask questions, how to find ideas, how to shape them, how to present them, but most importantly, they taught me how to trust myself that I can always find a solution.

During my Ph.D. studies, I had the opportunity to work closely with Dr. Sergio Consoli at the European Commission Joint Research Centre (JRC). I would like to express my gratefulness as he provided me guidance and feedback so that I have made an extra-mile towards a more qualitative work.

Last but not least, I must say that I would not have been able to complete this dissertation without the help and support of my family, my colleagues, and friends. I am deeply grateful.

Statement of Authorship

The thesis entitled *Artificial Intelligence Driven Systems for Financial Forecasting* and the presented works have been done during my candidature for this PhD degree. The presented novel approaches were carried out by myself, under the coordination of my supervisors and the collaboration of the authors indicated in the respective published papers. During the elaboration of this thesis I have follow appropriate ethics guidelines to conduct this research, I have acknowledged all main sources of help and, when I consulted or quote the work published by others, the source is always given.

Acronyms

ARIMA Auto-Regressive Integrated Moving Average.

DT Decision Tree.

EMH Efficient Markets Hypothesis.

GB Gradient Boosting.

GICS Global Industry Classification Standard.

IS In-Sample period.

LGB Light Gradient Boosting.

LR Lagged price returns.

LSTM Long short-term memory.

MDD Maximum drawdown.

MDI Mean Decrease Impurity.

ML Machine Learning.

OOS Out-of-Sample period.

PI Permutation Importance.

RF Random Forests.

SR Sharpe Ratio.

StatArb Statistical Arbitrage.

SVM Support Vector Machines.

SVR Support Vector Regression Machines.

TI Technical Indicators.

XAI Explainable Artificial Intelligence.

Latin Expressions

e.g.	exempli gratia, ‘for the sake of example’
ergo	‘therefore’
et al.	et alii, ‘and others’
i.e.	id est, ‘that is’

Notations

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
$\mathbb{A} \backslash \mathbb{B}$	Set subtraction, i.e., the set containing the elements of $\mathbb{A}$ that are not in $\mathbb{B}$

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
a_i	Element i of the random vector $\mathbf{a}$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\|\|\boldsymbol{x}\|\|$	L^2 norm of $\boldsymbol{x}$
$\mathbf{1}_{\text{condition}}$	is 1 if the condition is true, 0 otherwise

Datasets

$\mathbb{X}$	A set of training examples
$\boldsymbol{x}_i$	The i-th example (input) from a dataset
y_i or $\boldsymbol{y}_i$	The target associated with $\boldsymbol{x}_i$ for supervised learning
$\boldsymbol{X}$	The $m \times n$ matrix with input example $\boldsymbol{x}_i$ in row $\boldsymbol{X}_{i,:}$

Publications

The research reported in this thesis has contributed to the following publications:

- [1] Salvatore Carta, Diego Reforgiato Recupero, Roberto Saia, and-Maria Madalina Stanciu. A general approach for risk controlled trading based on machine learning and statistical arbitrage. In The Sixth International Conference on Machine Learning, Optimization, and Data Science (LOD 2020), volume 12565, pages 489–503. Lecture Notes in-Computer Science, 2020.

- [2] S. M. Carta, S. Consoli, A. S. Podda, D. Reforgiato Recupero, and M. M.Stanciu. Ensembling and dynamic asset selection for risk-controlled statistical arbitrage.IEEE Access, 9:29942–29959, 2021.

- [3] Salvatore Carta, Sebastian Podda, Diego Reforgiato Recupero, and Maria Madalina Stanciu. Explainable AI for financial forecasting. In The 7th International Conference on Machine Learning, Optimization,and Data Science (LOD 2021). Lecture Notes in Computer Science, to appear, 2021.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

> *"I think it's hugely important to have a strong episode one; you can lose an audience so quickly now. You can't afford to take the attitude that you will use the first one as an introduction and save the high drama for later."*
>
> **Jed Mercurio**

1.1 Motivation and Research Contributions

Considered a branch of artificial intelligence, Machine Learning (ML) aims at designing algorithms that are able to learn from past experience in order to make predictions about future observations, without being explicitly programmed to do so [4]. Recent advances have made ML algorithms experts of their own kind, capable to uncover by themselves the predictive structure of data. Public and famous examples include the use of boosted decision trees in the statistical analysis that led to the detection of the Higgs boson at CERN [5], the use of random forests for human pose detection in the Microsoft Kinect [6] or the implementation of various ML techniques for building the IBM Watson system [7], capable to compete at the human champion level on the American TV quiz show Jeopardy.

With the advent of ML, the financial sector has experienced a revolution in some of its core practices. In a recent article [8], Bartram *et al.* provide a systematic overview of the wide range of existing and emerging ML applications in the financial domain. Their study delineates three major areas in which ML can play a central role: trading, portfolio management, and portfolio risk management.

In this thesis, broadly speaking, our focus is on the integration of ML models in financial forecasting systems responsible with performing algorithmic trading.

Typical trading systems are based upon mathematical models meant to monitor market behavior, trade activities in real-time in order to detect any factors that can force security's (*e.g.* stock, future, or any financial instrument) price fluctuations. Over the years, a plethora of *statistical and econometric techniques* have been developed to analyze and predict securities' behavior. The literature reports two main such directions. The first one is the *fundamental analysis* [9], which consists of evaluating the intrinsic value of a security and analyzing the factors that could influence its price in the future. This form of analysis is based on macroeconomic factors such as the state of the economy and industry conditions and microeconomic factors like the effectiveness of the company's management. The second direction is the *technical analysis* [10], which takes into account the securities' historical prices. Technical analysis relies on the consideration that a security's prices already include all the fundamental information that could affect its future prices and do not measure a security's intrinsic value. Instead, it relies on identifying patterns and trends that suggest stock's price fluctuations in the future. Any type of analysis appear to be attainable for a human being under the consideration that the complexity of the security's behavior is low, the observations are consistent, or the number of analyzed securities is low. However, the analysis task becomes more challenging when the data for the security of interest represents a realization of a complicated event influenced by several factors and when a large amount of information is available. To break this cognitive barrier, the traders geared up to exploit the benefits of ML and leverage a competitive business edge [11]. The nowadays speed and complexity of trades have made ML techniques an essential part of trading practice. It is well-known that ML models can capture nonlinear dependencies and interaction effects that may lead to superior predictions. Leveraging ML models, sophisticated systems can be trained to automatically execute trades on the basis of trading signals, which enabled a whole new concept of *smart* algorithmic (or algo) trading.

Algorithmic trading refers to the use of algorithms to make better trade decisions. In other words, algorithmic trading is a system of trading whereby advanced ML models and computer programs are used to facilitate trading and make decisions in the financial markets. The models learn various parameters such as timing, price, quantity, and other factors for placing trades which don not require active involvement of the human traders. Such ML-based trading systems use as input for their models a range of features, including technical and fundamental indicators, and plain text (like online posts and news articles), to predict future returns [8]. The models make forecasts of future securities' performance metrics, the most popular being returns. These predictions then form the basis of an investment strategy that usually favors stocks that will outperform and moves away from those that will underperform.

Unlike human traders, algorithmic trading can simultaneously analyze large volumes of data and make thousands of trades every day. ML makes fast trading decisions, which gives human traders an advantage over the market average. In addition, ML techniques can help minimize transaction costs.

Furthermore, algorithmic trading does not make trading decisions based on emotions, which is a common limitation among human traders whose judgment may be affected by emotions or personal aspirations [12]. A recent example of such human mistake which made world-wide tour is when Elon Musk told his followers to use 'Signal', a messaging app which is an alternative to WhatsApp. Investors pounced on a totally obscure and unrelated stock named 'Signal Advance Inc'. The frenzied buying caused the stock to surge 1100% in just two days.

However, the incorporation of ML in algorithmic trading faces important challenges.

Finance is sometimes viewed as a field awash with applicable data, spanning from financial and economic sources to more recent unstructured data such as online news and social media posts. Even though the breadth of financial data is relatively large, we usually face the challenge of having rather *short time series* (*e.g.* stock prices) by ML and especially by deep learning standards. For example, if an ML models is tasked at predicting the daily return using daily stock market information, we would have for each year only 252 observations as many as trading days in an year. A limited number of time series observations mean that any model using the data is confined to being less trained or improperly designed. Consequently, the ML models cannot operate near their full potential. Moreover, due to its complexity and intricate interdependencies, financial data cannot be synthesized in a similar manner to other fields. For example, in image processing, which is a field of successful ML application, researchers can produce photos using generative models for the models to train with. In the financial domain, however, there is no alternative but to wait for financial data to be produced over time.

There are exceptional cases in finance where data is available, *i.e.*, high-frequency trading. In such cases, ML models can be provisioned with a large number of observations across time from which to learn. However, even in these cases, ML in finance faces its second-biggest challenge: the *signal-to-noise ratio*. On the one hand, ML models are highly dependent on data quality. Poor quality and noisy data lead to unreliable ML models. On the other hand, financial market movement is considered difficult to forecast due to its high stochasticity and near random-walk behavior [13]. The reason for this, of course, is that when following the Efficient Markets Hypothesis (EMH) [14], one should only be able to predict one variable in fully efficient financial markets. That variable is risk premia [1], which is small and difficult

[1]Underlying the terminology is the notion that there should be a premium (higher

to capture on short horizons. In other words, data from more distant history may be less relevant for training ML algorithms, and experts have at their disposal an inherently short data sample with a low signal-to-noise ratio. In the absence of large and reliable datasets, ML models are otherwise tasked with finding a needle in a haystack.

Predicting time-series returns in financial markets is fundamentally different from other mainstream applications of ML, where the underlying data generation process is relatively stable over time. The key difference is *data evolution*. Taking image recognition again as an example, images of humans always have the same features and by using them, the ML models can learn to identify objects (*e.g.* animals, plants, persons, etc.). In contrast, the features in financial data in harmony with financial markets evolve and change with time as a result of competitive dynamics and structural shifts. This implies that financial variables will not carry the same meaning and significance over a long period, *e.g.* one decade. This dynamic make it particularly difficult for the ML model to derive a consistent explanation and learning over a reasonably long period of time. This issue is especially relevant to investment applications.

Because of such unique characteristics of financial information, some practitioners [16, 17] have called for establishing financial data science as a standalone field in its own right, wherein greater emphasis is placed on empiricism and data-driven expansions of traditional financial econometrics. The two primary goals of data analysis, as noted by Breiman [18], are to make a prediction and to obtain information that aids in understanding. However, most ML models are so-called "black boxes" and do not provide any insights regarding how they produce specific results. This lack of interpretability makes it difficult to understand whether an ML model captures economically meaningful patterns or pure noise. In this context and given the constraints of short and noisy financial data, it is at utmost importance to clearly identify the input features that strengthen the signal responsible for the ML model predictive power. Understanding which features contribute the most the the financial performance of a ML-driven system and selecting the most appropriate ones opens the proverbial "black-box".

Along the challenges previously outlined, we pose four distinct questions and answering them constitutes the objectives of this dissertation:

1. How can we mitigate the short and noisy datasets when using ML in a trading system?

2. How can we overcome the structural data-shift that is inherent to the trading activity?

expected return) for undertaken risk [15]

3. How can we select the appropriate input features for ML-driven trading system?

4. Are the same input features relevant to other related financial instruments?

In the following paragraphs, we sketch the overall approach to the ML-driven systems explored in this dissertation together with the proposals for tackling it, that constitute our research contribution.

The overall *aim* of this thesis is to design, develop, and validate a trading system that fulfills the widespread requirements of incorporating ML models provided scant and noisy data is available.

Chapter 4 objective is to address the first two questions laid above. In other words, we consider the problem of forecasting the price returns of a large set of stocks and perform daily trading. In order to mitigate the low signal to noise ratio we propose an ensemble methodology for financial forecasting, tackling the ensemble construction from three different perspectives:

- *model diversity*, by using ML models and even statistical models;
- *data diversity*, by considering lagged price returns and technical indicators to enrich the data used by the models;
- *method diversity*, by simultaneously training a single model across several assets and, conversely, individual models for each asset;

Chapter 5 addresses the last two questions and we aim to integrate ML and explainability technologies in order to increase the financial performances of a trading system. Specifically, for a series of ML models, we derive the feature importance score for each input feature. In this setup, we propose three strategies to select the best subset of features to remove *for each stock*, with the goal of improving the overall predictive performance. Moreover, the proposed strategies are able to *learn* which features can be removed both in cross-section (across multiple stocks) as well as at an individual level (for each stock) and in doing so increase the overall prediction performance (across all the stocks).

We carry out an empirical evaluation of the above on large set of stocks, *i.e.*, S&P500 index constituents, demonstrating potential of the proposed trading strategies. Our assessment includes various perspectives: ML models goodness of fit and risk-return perspective, all necessary when coupling ML models with algorithmic trading.

1.2 Dissertation Structure

The remainder of this thesis is organised as follows:

- Chapter 2 provides an overview of the type of problems and typical setups that we address throughout the dissertation, and defines the notion of a "ML-driven trading system" – the basic modeling steps and requirements that are shared among all of the problems under consideration. Moreover, we present the metrics used to evaluate our approaches. Finally, it describes which software tools and libraries have been adopted;

- Chapter 3 presents the state of the art in terms of ML applied to financial forecasting and algorithmic trading which are necessary for understanding the research context under which this work has been developed;

- Chapter 4 presents a methodology for building a heterogeneous ensemble taking into considerations various facets that contribute to a highly performing ensemble, *i.e.*, data diversity and model diversity. Moreover, acknowledging the ever-changing financial market's behaviour, we introduce a method of "asset selection". This work has been published in [1, 2];

- Chapter 5 presents a data-driven approach of financial forecasting and algorithmic trading through the lens of Explainable Artificial Intelligence (XAI). One of the key elements of efficient financial forecasting is the input features. Therefore we propose methods of automatic feature selection with the use of XAI techniques in order to enhance ML models predictions. Preliminary results of this work have been published in [3];

- Chapter 6 presents the concluding remarks about the use of ML as an integrative part of an algorithmic trading system.

Chapter 2

Machine Learning-driven Trading System Fundamentals

"Therefore, in the course of the work I have followed this plan: I describe in the first book all the positions of the orbits together with the movements which I ascribe to the Earth, in order that this book might contain, as it were, the general scheme of the universe."

Nicolaus Copernicus

Outline

This chapter introduces the various building blocks of an ML-driven trading workflow and fundamental concepts on which this work builds upon. In Section 2.1, we first describe the financial data. In Section 2.2, we proceed by presenting formally the models used thought this dissertation and in section Section 2.3 we describe procedures for determining the best input features for a ML model. In Section 2.4 we make a brief overview of some main methods of quantitative trading, whereas in Section 2.5 we describe how to test the trading methods and in Section 2.6 we proceed with a discussion on performance evaluation. We finalize by listing the software libraries that were used in the experiments of this disseration in Section 2.7.

A high-level abstraction of the typical ML-driven algorithmic trading workflow is illustrated in Figure 2.1. In what follows, we describe each stage of the workflow and its significance the proposed algorithmic approaches.

An algorithmic trading workflow involves the following steps/stages, with a specific investment universe and time horizon in mind [19]:

Data sources and engineering predictive features – ML algorithms promise to exploit market and fundamental data more efficiently than

Figure 2.1: The building blocks of an ML-driven trading strategy

human-defined rules and heuristics. However, given the intrinsic characteristics of market data, the stage of feature engineering plays an essential role as data quality might undermine the validity of any trading strategy.

Design, tune, and evaluate ML models – concerns making modeling choices that comprise the broad, preset assumptions on the nature of the model used to fit the data. For instance, we might seek to fit a regression model to predict a stock prices. Potential modeling choices would include: whether the model should be a simple linear model, or a complex non-linear one, and whether interactions between stock variables should be accounted for. This stage of the pipeline is crucial because it sets an upper bound on how well the ML model can accurately capture the data.

Predictions – concern based on the predictions of the previous step generate trading signals to buy or sell assets.

Asset selection – accounts for a series of rules to decide the "bet sizing", *i.e.*, the number of trades to execute on the market in order to minimize risk and increase the return.

Backtesting – which entails simulating the trading positions using historical market data. This steps aims to gather evidence from historical data about the performance of a candidate trading strategy.

Evaluation – quantifies thought specific risk-return metrics how the resulting positions (of the previous steps) would have performed it would have been used in the past. At the same time, assess how the ML models performed on "unseen" data. This step is essential as it offers a good outcast about the goodness of fit of ML models as well as the entire trading strategy in an close to real trading scenario.

With this in mind, we start by introducing the data and diverse methods to engineer informative features. Then, we briefly present various forecasting algorithms, with focus on the algorithms used thought this dissertation.

Then, we shift our focus to understanding how a model reaches its outcomes as is very important for several reasons, including trust, actionability, accountability, and debugging. For this reason, we present various methods to gain insight into the drivers of the ML models' predictions. We also present basic principles of algorithmic trading, more specifically of that of Statistical Arbitrage and the methodology of backtesting the trading strategies against historical data. Finally, the evaluation metrics used throughout this work will be provided and explained.

2.1 Data and Feature Engineering

Data has always been a critical component of trading, and traders have long sought to gain an advantage by accessing better data. These efforts stretch at least as far back as tales that the House of Rothschild profited greatly from bond purchases based on information about the British victory at Waterloo conveyed across the channel by pigeons [19].

Returning to more modern times, Clive Humby famously stated in 2006 that "Data is the new oil. It's valuable, but if unrefined it cannot really be used. It has to be changed into gas, plastic, chemicals, etc to create a valuable entity that drives profitable activity; so must data be broken down, analyzed for it to have value." Since then, it has become a common refrain. In 2011, Peter Sondergaard, senior vice-president of Gartner, added another nuance by saying if data is the new oil, then "analytics is the combustion engine". This metaphor has its merits as data may be considered a valuable resource, but only if we can figure out how to extract value from it appropriately. In other words, like oil, data is only valuable if it is in a usable. Similarly to oil, raw data needs to be preprocessed before it can be used for analytics or serve as input for forecasting. That is because raw financial data may suffer from some of the following flaws:

- The data contains inconsistent or inaccurate information.
- The data contains missing information.
- The data does not represent the population that it was intended to represent.
- The data is not in a form that is ready for predictive analytics.

As such, in the financial domain, the investors seek to transform raw financial data into alpha factors. That is, identifying information or variables that help to reliably predict asset returns. Alpha factors are market, fundamental, and alternative data transformations that contain predictive signals.

Thus, an important objective is, using past data, to develop such factors that reflect main return drivers as well as the risk embodied by the return.

Typical techniques of alpha factors extraction revolve around two directions. The first one concentrates on securities' fundamental data and construct financial models based on reported financials, which may be supplemented with industry or macro data to forecast stock prices. Fundamental data pertains to the economic drivers that determine the value of securities [9]. The nature of the data depends on the asset class[1]. For example, for equities and corporate credit, it includes corporate financials as well as industry and economy-wide data. For commodities, it includes asset-specific supply-and-demand determinants, such as weather data for crops. The goal is to ultimately identify which assets are priced correctly or incorrectly.

Contrasting with fundamental analysis is the technical analysis that can draw signals from market data by using indicators derived from price and volume data [20, 21]. Technical analysis operates under the assumption that past trading activity and price changes of a security can be valuable indicators of the security's future price movements when paired with appropriate investing or trading rules. Technical analysis, as known today, was first introduced by Charles Dow and the Dow Theory in the late 1800s [21].

Charles Dow, in his writings, made two basic assumptions that have continued to form the framework for technical analysis trading: (i) markets are efficient with values representing factors that influence an assets' price; (ii) however, even random market price movements appear to move in identifiable patterns and trends that tend to repeat over time.

Today the field of technical analysis builds on Charles Dow's work. Professional analysts typically accept three general assumptions for the discipline:

The market is informationally coherent: Everything from a company's fundamentals to broad market dynamics to market psychology, according to technical analysts, is already priced into the stock. This viewpoint is congruent with the EMH [14] which posits a similar conclusion about prices. The only thing remaining is the analysis of price movements, which in technical analysts view is the product of supply and demand for a particular stock in the market.

Prices have trends: Even in random market movements and regardless of the observed time range, technical analysts expect prices to show trends. In other words, the price of a stock is more inclined to keep a previous

[1]Securities are assets specific to the capital markets. A security is a financial instrument that represents an ownership position in a publicly-traded corporation, *i.e.*, equity assets (stock), a creditor relationship with governmental body or a corporation, *i.e.*, debt assets (bond). Throughout this dissertation, the terms security and asset are used interchangeably, but equity assets are being considered.

trend than to move erratically. This premise underpins the majority of technical trading systems [22, 23].

History tends to repeat itself: According to technical analysts, history tends to repeat itself. The repeated nature of price movements has been attributed to market psychology which tends to be somewhat predictable based on emotions like fear or excitement [24, 25]. To comprehend trends technical analysts employ chart patterns to examine these emotions and subsequent market movements. While many forms of technical analysis have been around for almost a century, they are still thought to be useful because they show patterns in price movements that frequently repeat.

ML algorithms promise to exploit data more efficiently than human-defined rules and heuristics. The basic idea behind ML is that a given set of data contains enough information to be used to predict an output. An ML model must therefore, find the relationships and patterns hidden within the data. However, for ML-driven systems, feature engineering is a key ingredient for successful predictions. This is no different in trading. That is because raw financial information is organized as time-series where each observation is also timestamped. In the data processing and feature engineering it is crucial to avoid look-ahead bias. Therefore, data should be carefully managed and curated by adjusting it to the desired frequency on a point-in-time basis. This implies that data should only reflect information that was available and known at the time. Forecasting algorithms that have been trained on biased historical data are almost certain to fail in live trading.

In the following sections, further details will be provided about the features that are used thought this dissertation, and details about raw financial data is processed to avoid the look-ahead bias.

Lagged Returns as Features

In finance, a return defines a profit on an investment. A return covers any change in the value of the investment and/or cash flows (or other investments) investor receiving from investment (*e.g.* interest payments, coupons, cash dividends). Moreover, a return measures either in absolute terms (*e.g.* dollars) or as a percentage of the amount invested over a period of time, called the holding period return. A loss contrary to a profit describes as a negative return, assuming the amount invested is greater than zero. Calculating the *Return* (or the holding period *Return*) over a single period of any length of time is expressed as:

$$Return = \frac{V_f - V_i}{V_i},$$

where V_f is the final price value, including dividents and interest and V_i is the initial price value.

Throughout this dissertation, there are used as input features the Lagged price returns (LR), that is the returns of an asset computed based on and *Close* price as V_f and *Open* price as V_i with different holding periods or starting at different moments in time with respect to t, a reference moment in time. That is, for an asset (it can be a stock or any other financial instrument) the *Return* as feature with respect to time t and a holding period of 1 day would be expressed as:

$$Return_{t,1} = \frac{Close_{t-1} - Open_{t-1}}{Open_{t-1}} \tag{2.1}$$

Moreover, the features are identified as being *lagged* to indicate that the return time series contains its own prior (*i.e.*, lagged) values with respect to the instance of time t.

Technical Indicators as Features

The second type of input features include technical analysis indicators. Table 2.1 shows their list and formulas.

Roughly, the indicators can be divided into four categories[2]:

- Trend followers - identify the main movements of stock prices in recent past (EMA(10)). **EMA(10)**, is a type of moving average that places a greater weight and significance on the most recent data samples. The use of this type of indicator has been considered as the goal is predicting the short-term future.

- Divergence identifiers - identify possible regime switches, e.g. the current trend ends and the prices that will start moving in the opposite directions (AccDO, MACD). **AccDO** provides insight into how strong a trend is by using both price and volume information. If the price is rising but the indicator is falling this indicates that accumulation volume may not be enough to support the price rise, thus, the price might drop in the near future. **MACD** follows the trend of the asset, *i.e.*, if its value goes up then asset price also goes up and vice-versa.

- Momentum indicators - determine the momentum that a stock has with respect to its upward or downward trajectory in the market (ROC, RSI, Disp (5), and Disp (10)). **ROC** measures the percentage change in price between the current price and the price a certain number of

[2]Information about technical indicators use and their interpretation has been collected from https://www.investopedia.com/.

Table 2.1: Technical Indicators their acronyms and formula

Name of technical indicator	Formula
Exponential Moving Average ($EMA(10)$)	$(C_t \times a) + (EMA_{t-1} \times (1 - a))$ where $a = 2/(n+1)$
Stochastic %K (%K)	$\%K = \frac{(C_t - LLt-n)}{(HH_{t-n} - LL_{t-n})} \times 100$
Price rate of change (ROC)	$\frac{C_t - C_{t-n}}{C_{t-n}} \times 100$
Relative Strength Index (RSI)	$100 - \frac{100}{1+(U/T_n)}$
Accumulation Distribution Oscillator (ADO)	$\frac{(C_t - LL_{t-n}) - (HH_{t-n} - C_t)}{HH_{t-n} - LLt-n} \times V$
Moving Average Convergence - Divergence (MACD)	$EMA_{12}(t) - EMA_{26}(t)$
Williams %R	$\frac{HH_{t-n} - C_t}{HH_{t-n} - LL_{t-n}} \times 100$
Disparity 5 (D(5))	$\frac{C_t}{MA_5} \times 100$
Disparity 10 (D(10))	$\frac{C_t}{MA_{10}} \times 100$

C_t is the closing price at time t, L_t the low price at time t, H_t high price at time t, LL_{t-n} lowest low in the last $t-n$ days, HH_{t-n} highest high in the last $t-n$ days, MA_t the simple moving average of t days, U represents the total gain in the last n days and T_n represents the total loss in last n days. In this paper n has been set to 10.

periods ago. ROC is used to spot divergences, overbought and oversold conditions. **Disp (5)** and **Disp (10)** have a similar interpretation, *i.e.*, a value greater than zero suggests that the asset is gaining upward momentum, whereas, a negative value is a sign that selling pressure is increasing, forcing the price to drop. **RSI**, ranges between 0 and 100 and is generally used for identifying the overbought and oversold assets. That is, if **RSI** exceeds the level of 70, it is an indication that the asset is overbought, so, the asset's price may go down in near future, whereas values below 30 level indicate that the asset is oversold, so, the asset's price may go up in near future.

- Oscillators - follow the price variation of the stocks in order to identify possible reverse points (%K and Williams %R). To note that when stochastic oscillators are increasing, the asset prices are likely to go up and vice-versa.

2.2 Forecasting Algorithms

In [26] Tom Mitchell, provides a succinct definition of what a forecasting algorithm is:"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". One can imagine a wide variety of experiences, tasks, and performance measures, but in the following sections, while a broad context of the tasks, experiences, and performance measures is provided, the focus will be on the concepts used extensively in this dissertation.

Differently put, the objective which is sought is to find a systematic way of predicting a phenomenon given a set of observations. In ML terms, this goal is formulated as the learning Experience of inferring from collected data, a model that predicts the value of an output variable based on the observed values of input variables. Finding an appropriate model is based on the assumption that the output variable does not take its value at random and that there exists a relation between the inputs and the output. To give an example, in finance, an asset's price represents a phenomenon. The observations are the values of the price at a finite number of time points, collected with a given frequency, *e.g.* hourly, daily, etc. For an asset, the goal is to find a decision rule (*i.e.*,, a model) to predict the future prices (*i.e.*, the output variable) given a set of observations such as past prices (*i.e.*, the input variables). Note, in finance, differently from other domains, the observations are also indexed by time.

Before proceeding with more details on the forecasting algorithms, a series of preliminary definitions are introduce. Let us consider a set of observations of a stochastic process $\mathcal{X} \subset \mathbb{R}^D$ be an input space and $\mathcal{Y} \subset \mathbb{R}^D$ be an output space, where D is the dimension of the input. The dimension of the output space is considered to be 1. We assume that the data is given in terms of time series $(\boldsymbol{x}_t)_{t\in[1:T]}$, where the inputs $\boldsymbol{x}_t \in \mathcal{X}$ are indexed by a time parameter $t \in [1:T]$ with $T \in \mathbb{N}^*$. We consider the problem of predicting a sequence $(y_t)_{t\in[1:T]}$ of outputs $y_t \in \mathcal{Y}$ indexed by the same time parameter. By definition, both the input and the output spaces are assumed to respectively contain all possible input vectors and all possible output values. Note that input variables are sometimes known as features, input vectors as instances or samples and the output variable as target.

Most ML algorithms simply experience a dataset. A dataset can be described in many ways, depending on the type of experience. It can be defined as $\mathcal{D}$, a set of T pairs of input vectors and output values $\mathcal{D} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$, where $\boldsymbol{x}_t \in \mathcal{X}$ and $y_t \in \mathcal{Y}$. In all cases, a dataset is a collection of examples, which are in turn collections of features.

The Task - is usually described in terms of how the forecasting algorithms

system should process each observation. Given $\mathcal{D}$ a time-series dataset, the task is to predict a target variable $\{y_t\}_{t=1}^{T}$ based upon the multivariate predictive variable $\{\boldsymbol{x}_t\}_{t=1}^{T}$ by using a learning algorithm that outputs a *learned* model function $f(\cdot)$. Depending on the type of output, the forecasting algorithms tasks can be roughly divided into two categories: (i) *classification* and (ii) *regression*. For classification tasks, the forecasting algorithm is asked to specify which of the c categories an observation belongs to. Formally the task is to learn a function $f : \mathbb{R}^D \longrightarrow \{1, \ldots, c\}$, thus, y_t will be a discrete numerical value. In turn, in regression tasks, the forecasting algorithm has to learn a function $f : \mathbb{R}^D \longrightarrow \mathbb{R}$, therefore y_t is a continuous numerical value. Going back to the stock forecasting example above of predicting the future prices/returns given a set of measurements such as past prices/returns, the Task is a regression one. If the goal would be to predict the direction on price movement (up or down) the Task is a classification one. In this dissertation, the focus lies on the former type of forecasting, *i.e.*, regression tasks.

The Experience - defines the context in which the forecasting algorithms operate, or in more generic terms what kind of experience they are allowed to have, *i.e.*, *unsupervised* or *supervised*. *Unsupervised learning* refers to such experience in which the algorithms try to learn some inherent structure from input data. On the other hand, the *supervised learning*, defines the manner in which algorithms are fed, in the sense that each observation in the training dataset it also associated with a *target* or *label*. With respect to the dataset definition above, in an unsupervised context, $\mathcal{D}$ is represented only by the input features, whereas in the supervised context the dataset is given by both input vectors and output values. On a more philosophical perspective, supervised context originated from the idea that a teacher who guides the learning process and provides the target variable, whereas in the unsupervised learning, there is no teacher and the algorithm must make sense of the data without this guide.

The Performance Measure - defines a series of quantitative metrics to assess the learning performance of the algorithm. Usually the learning is specific to the learning task. Section 2.6 presents in more detail the performance metrics used in this work.

Data representation

For optimal implementations of ML algorithms, data needs to be represented using structures which allow for high performance numerical computation. One common way of describing a dataset is with a *design matrix* [27].

A *design matrix* is a matrix containing a different example in each row. Each column of the matrix corresponds to a different feature. As such, the set of D-input vectors $\boldsymbol{x}_t$ (for $t = 1, \ldots, T$) can be denoted by a $T \times D$ matrix $\boldsymbol{X}$, whose rows $t = 1, \ldots, T$ correspond to the input vectors $\boldsymbol{x}_t$ and its columns $j = 1, \ldots, D$ to the input variables. The design matrix can be formalized as

$$\boldsymbol{X} = (x_{t;j})_{(t;j)\in[1:T]\times[1:D]},$$

where for a single feature j, an observation at time t is indicated by $x_{t,j}$. Similarly the corresponding output values y_t can be written as a vector

$$\boldsymbol{Y} = (y_1, \ldots, y_T) = (y_{t,i})_{(t)\in[1:T]}.$$

Note, the ML algorithms are described in terms of how they operate on the design matrix.

2.2.1 Statistical Learning Models

Auto-Regressive Integrated Moving Average

In statistics and econometrics, and in particular in time series analysis, Auto-Regressive Integrated Moving Average (ARIMA) and Autoregressive Moving Average (ARMA)[3] models are widely used. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). The ARIMA was first introduced in [28] and, since then, it has proven to be robust and efficient for short-term prediction [29, 30]. ARIMA models are applied the cases where data show evidence of non-stationarity in the sense of mean, where an initial differencing step should be applied one or more times to eliminate the non-stationarity of the mean function (*i.e.*, the trend). The algorithm captures a suite of different time-dependent structures in time series and as its acronym indicates $ARIMA(p, d, q)$ comprises three parts:

Auto-Regression model that uses the dependencies between an observation and a number of lagged observations (p). This component of the model indicates that the model regresses the variable of interest on its own prior values.

Integration differencing of observations with the degree d, to make the time series stationary. In other words the term I indicates the data values have been replaced with difference between their values and the previous values.

[3] ARIMA is a generalization of ARMA

Moving Average model that accounts the dependency between observations and the residual error terms when a moving average model is used to the lagged observations (q).

Mathematically, ARIMA models can be estimated with the Box-Jenkins approach [31]:

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L^i)^d x_t = \delta + (1 - \sum_{i=1}^{q} \theta_i L^i)\varepsilon_t, \tag{2.2}$$

where ϕ_i are coefficients for the auto-regressive part of the model, θ_i are coefficients of the moving average part of the model, and ε_t is the error term at time t, δ denotes the intercept, and L is the lag operator. The lag operator operates on the elements of a time-series to produce previous elements and is defined as $L^i x_t = x_{t-i}$. Finally, the ε_t terms are assumed to be i.i.d. variables and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

2.2.2 Machine Learning Models

One of the most used supervised learning algorithms in financial forecasting [16] are Decision Tree (DT) based ensembles. Briefly, the learning of a DT consists of a hierarchical rule-set, where each rule represents a node in an arborescent structure. The DT splits the dataset into smaller and smaller subsets (usually hyper-rectangles) while at the same time, the associated DT is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches, each representing values for the attribute tested. If a node does not have any children, it is called a leaf node. A leaf node represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node.

Based on decision-trees,there were developed two methods of ensembling the so-called weak-learners. That is: Random Forests (RF) [18], which entail bagging and random subspace search, and Gradient Boosting (GB) [32], based on boosting [33] or differently put greedy gradient descent search in function space. Each of the two methods has its own variations.

Mathematically, an ensemble of weak learners can be expressed as:

$$\varphi(\boldsymbol{x}) = \sum_{j=1}^{M} \alpha_j h_j(\boldsymbol{x}), \text{ with } \alpha_j \in \mathbb{R} \text{ and } h_j(\boldsymbol{x}) : \chi \longrightarrow \mathbb{R},$$

where χ is the family of learners, $h_j(\cdot)$ a weak learner, α_i the weight of the weak learner in the final strong-learner, $\varphi(\cdot)$. Thus, learning means solving

the following problem:

$$\{\alpha_i^*, h_j^*\}_{j=1}^M = \arg\min_{\alpha_i, h_j} \sum_{i=1}^{T} L(y_t, \varphi(\boldsymbol{x}_t)),$$

where L is the loss function.

Random Forests

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging [34], to tree learners. Given the training set $\mathbb{D}$, the algorithm performs the steps, as shown by Algorithm 1.

Algorithm 1: Bagging algorithm

Input:
Training samples $\mathcal{D} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^T$
Predictor learning procedure $\mathcal{L} : \{\chi\} \longrightarrow \mathcal{H}$
Number of learners M
Output: Trained decision trees h_j;

1: **begin**
2: **for** $m = 1, \ldots, M$ **do**
3: Sample with replacement, n training examples from $\boldsymbol{x}_t, y_t$, called x_b, y_b
4: Learn $h_m(\cdot) = \mathcal{L}(x_b)$

By bagging repeatedly (M times) the algorithm selects a random sample with replacement of the training set and fits trees to these samples. After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\varphi(x') = \frac{1}{M} \sum_{m=1}^{M} h_m(x').$$

The key idea is that bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. Although the predictions of a single tree are highly sensitive to noise in its associated training set x_b, y_b, the average of many trees is not, because the trees are decorelated. This is achieved by bootstrapping the training data, thus, showing the trees different training sets.

Gradient Boosting

Boosting is another algorithm in the area of supervised learning that builds ensemble of weak learners for reducing bias and also variance [35]. While boosting is not algorithmically constrained, Gradient Boosting (GB) consist of iteratively learning weak learners with respect to a distribution and adding them to a final strong learner. Algorithm 2 presents in pseudocode the generic gradient boosting method. It starts by an initial weak learner, φ_0, to which in M iterations adds weak-learners. When they are added, they are weighted in a way that is related to the weak learners' forecasting performance (line!4). After a weak learner is added, the data weights are readjusted, known as "re-weighting" [36]. After each boosting iteration, the results of the prediction are evaluated according to a decision function and data samples are re-weighted in order to focus on examples with higher loss in previous steps (line 5).

Algorithm 2: Boosting algorithm

Input:
Training samples $\mathcal{D} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
A differentiable loss function $l(y, \varphi(\boldsymbol{x}))$
Number of learners M
Output: Trained decision trees h_j;

1: Initialize the model φ_0 with a constant value
2: **for** $m = 1, \ldots, M$ **do**
  `/* Compute pseudo-residuals */`
3: $r_{tm} = -\frac{\partial \ell(y_i, \varphi_{m-1}(\boldsymbol{x}_t))}{\partial \varphi_{m-1}(\boldsymbol{x}_t)}$
  `/* Fit a base learner` $h_m(x)$ `to the pseudo-residuals */`
4: $h_m = \arg\max_{h \in \mathcal{H}} \sum_{t=1}^{M} r_{tm} h(\boldsymbol{x}_t) - \lambda_t h(\boldsymbol{x}_t)^2$
  `/* Using line-search, compute the multiplier` α_m `that minimizes the cost` ℓ `*/`
5: $\alpha_m = \arg\min_{\alpha} \sum_{i=1}^{N} \ell(y_i, \varphi_{m-1}(x_i) + \alpha h_m(x_i))$
  `/* Upgrade the model */`
6: $\varphi_m(x) = \varphi_{m-1}(x) + \alpha_m h_m(x)$
7: Output $\varphi_M(x)$

To note that GB algorithm is fitting the weak learners to the residuals and is minimizing the loss l indirectly by applying a gradient descent algorithm (hence the requirement that the loss function to be differentiable). In this manner, after each iteration, the loss function is closer to the minimum. Moreover, the algorithm does not compute the cost function for each weak-learner split (assuming that the base learners are DTs), but it uses the residuals and the gradient of the loss as a proxy. In other words the, the

residuals offer an approximation of the loss direction (increase of decrease) by their sign and of the loss value by their magnitude.

Depending on the manner of fitting the trees and approximating the loss after each iteration a series of boosting algorithms were developed. One of the most popular is Light Gradient Boosting (LGB) proposed by Ke *et al.* [37]. The novelty of LGB lies in the construction of trees, in the sense that LGB does not grow a tree level-wise. Instead it grows trees leaf-wise. It chooses the leaf it believes will yield the largest decrease in loss. Besides, LGB does not use the widely-used sorted-based DT learning algorithm, which searches the best split cut-point on sorted feature values. Instead, it implements a highly optimized histogram-based DT learning algorithm, which yields great advantages on both efficiency and memory consumption. The basic idea of the histogram algorithm is to discretize successive floating-point eigenvalues into s integers and construct a histogram of width s. When traversing the data, the statistic is accumulated in the histogram according to the discretized value as an index. After traversing the data once, the histogram accumulates the required statistic and then traverses to find the optimal segmentation point according to the discrete value of the histogram.

Support Vector Machines

One of the most influential approaches to supervised learning is the Support Vector Machines (SVM).This type of algorithms was initially proposed for classification tasks [38], and later revised for regression in [39] and are known as Support Vector Regression Machines (SVR).

Given the input $\boldsymbol{x}$, the goal is to find a function that deviates from actual data, $\boldsymbol{y}$, by a value no greater than ϵ for each training point, and, at the same time, being as flat as possible. That is, to find the linear function $f(\boldsymbol{x}) = \boldsymbol{x}'\beta + b$. This is formulated as a convex optimization problem to minimize $J(\beta) = \frac{1}{2}\beta'\beta$ subject to all residuals having a value less than ϵ; or, in equation form:

$$\forall t \ : |y_t - (\boldsymbol{x}_t'\beta + b)| \leq \epsilon$$

It is possible that no such function $f(\boldsymbol{x})$ exists to satisfy these constraints for all training points $\boldsymbol{x}_t$. To deal with otherwise infeasible constraints, there were introduced the slack variables ξ_t and ξ_t^* for each point. This approach is similar to the "soft margin" concept in SVM classification, because the slack variables allow regression errors to exist up to the value of ξ_t and ξ_t^*, yet still satisfy the required conditions.

Including slack variables leads to the objective function:

$$J(\beta) = \frac{1}{2}\beta'\beta + C\sum_{t=1}^{T}(\xi_t + \xi_t^*),$$

subject to:

$$\forall t : y_t - (\boldsymbol{x}_t\beta + b) \leq \epsilon + \xi_t$$
$$\forall t : (\boldsymbol{x}'_t\beta + b) - y_t \leq \epsilon + \xi^*_t$$
$$\forall t : \xi^*_t, \xi_t \geq 0$$

where $C > 0$ is the regularization parameter, and $\epsilon > 0$ is an error sensitivity parameter. The training, for each observation $\boldsymbol{x}_t$ finds the non-negative multipliers α_t and α^*_t such that the parameter β can be described as the linear combination of the training observations as $\beta = \sum_{t=1}^{T}(\alpha_t - \alpha^*_t)\boldsymbol{x}_t$. Thus, the linear function used by the learning algorithm can be rewritten as dependent only on the support vectors as:

$$f(\boldsymbol{x}) = b + \sum_{t=1}^{T}(\alpha_t - \alpha^*_t) \cdot \langle \boldsymbol{x}_t, \boldsymbol{x} \rangle$$

This enables us to replace $\boldsymbol{x}$ with the output of a given feature function $\phi(\boldsymbol{x})$ and the dot product with a function $\kappa(\boldsymbol{x}, \boldsymbol{x}_t) = \phi(x) \cdot \phi(\boldsymbol{x}_t)$ called a **kernel**. In this manner, the above concepts can also be extended to the non-separable case (linear generalized SVR).

$$f(x) = b + \sum_{t=1}^{T}(\alpha_t - \alpha^*_t) \cdot \langle \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}_t) \rangle$$
$$f(x) = b + \sum_{t=1}^{T}(\alpha_t - \alpha^*_t) \cdot k(\boldsymbol{x}, \boldsymbol{x}_t)$$

This function is nonlinear with respect to x, but the relationship between $\phi(\boldsymbol{x})$ and $f(\boldsymbol{x})$ is linear. Also, the relationship between α and $f(x)$ is linear. The kernel-based function is equivalent to preprocessing the data by applying $\phi(\boldsymbol{x})$ to all inputs, then learning a linear model in the new transformed space.

In our study, the radial basis kernel function was selected, which can be formalized as $\kappa(\boldsymbol{x}_{t1}, \boldsymbol{x}_{t2}) = exp(-\gamma\|\boldsymbol{x}_{t1} - \boldsymbol{x}_{t2}\|^2)$. Here $\boldsymbol{x}_{t1}$ and $\boldsymbol{x}_{t2}$ represent two feature vectors of the input space, γ is a free parameter, considered as a design parameter of the SVR.

Concluding, SVR solves an optimization problem that involves two parameters: the regularization parameter, C, and the error sensitivity parameter ϵ. C controls the trade-off between model complexity and the number of non-separable samples. A lower C will encourage a larger margin, whereas higher C values lead to a hard margin [38].

2.2.3 Cross-Validation

Most ML algorithms have hyperparameters, settings that allow controlling the algorithm's behavior. The values of hyperparameters are not adapted by

the learning algorithm itself. Moreover,the setting must be a hyperparameter because it is not appropriate to learn that hyperparameter on the training set. If learned on the training set, such hyperparameters would always choose the maximum possible model capacity, resulting in overfitting [27]. Usually, for ML models hyperparameter optimization, a held-out test set is used called validation set, composed of examples coming from the same distribution as the training set, can be used to estimate the generalization error of a learner, after the learning process has completed. Since the validation set is used to "train" the hyperparameters, the validation set error will underestimate the generalization error, though typically by a smaller amount than the training error does. After all hyperparameter optimization is complete, the generalization error may be estimated using the test set.

Dividing the dataset into a fixed training set and a fixed test set can be problematic if it results in the test set being small. A small test set implies statistical uncertainty around the estimated average test error, making it difficult to claim that algorithm A works better than algorithm B on the given task. When the dataset has hundreds of thousands of examples or more, this is not a serious issue. When the dataset is too small, such as financial time-series usually are, alternative procedures enable one to use all the examples in the estimation of the mean test error, at the price of increased computational cost. These procedures are based on the idea of repeating the training and testing computation on different randomly chosen subsets or splits of the original dataset. The most common of these is the k-fold cross-validation procedure, in which a partition of the dataset is formed by splitting it into k non-overlapping subsets. However, k-fold cross validation is unsuitable for time-series data where the temporal dependency between observations should be taken into accounts. Usually, for time series a more suitable approach is *TimeSeriesSplit*. That each for for k trials split the training data i into two folds, on condition that the training fold is always ahead of the validation fold. Then, the test error may then be estimated by taking the average validation error across k trials. On trial i, the i^{th} subset of the data is used as the validation set, and the rest of the data is used as the training set. One problem is that no unbiased estimators of the variance of such average error estimators exist, but approximations are typically used.

2.3 Explainable Artificial Intelligence

Although employing ML techniques for decision making seems to be the best viable option, practitioners require to understand how the employed ML models undertake their decisions. Often referred to as "black boxes", ML models are developed with a single goal of maximizing predictive performance [40]. To fill this gap, researchers have developed frameworks to "X-ray" the ML

models and increase their transparency [41], and the field of Explainable Artificial Intelligence (XAI) has surged. Under the umbrella of XAI, in the literature [42] different perspectives and motivations can be found: explanations to justify, explanations to control, explanations to discover, and finally, explanations to improve classification or regression tasks. Throughout this work the focus is on the latter. In the following paragraphs the most prominent XAI methods in the literature are presented. At the same time, they are extensively used throughout this dissertation.

Preliminaries Given a forecasting Task (as described in Section 2.2) with a training set $\mathcal{D}$ our goal is to predict a target variable $\boldsymbol{Y}$ based on the multivariate predictive variable $\boldsymbol{X}$. A learning algorithm can be used to output a model function $f(\cdot)$ representing the estimate of $\boldsymbol{Y}$ based on $\boldsymbol{X}$. The goal is to explain the prediction $\boldsymbol{Y} = f(\boldsymbol{X})$ of a black-box f that has been pretrained for the aforementioned prediction task. In other words, our method aims to explain individual predictions of a given black box. More specifically, our purpose is to identify the parts of the input $\boldsymbol{X}$ that are the most relevant for f to produce the prediction Y. In this context, in the remainder of this section, three methods for determining the feature importance and related to our work are introduced.

Feature importance is defined as an array of numbers where each number corresponds to a feature, indicating how much the feature contributed to the model prediction [43]. It can be said that a feature $\boldsymbol{X}_{:,j}$ is a noisy feature if $\boldsymbol{X}_{:,j}$ and $\boldsymbol{Y}$ are independent, and a relevant feature otherwise.

Permutation Importance

The permutation feature importance measurement was introduced by Leo Breiman [18] for Random Forests. Based on this idea, Fisher, Rudin, and Dominici [44] proposed a model-agnostic version of the feature importance and called it model reliance. Note, in the literature can be found under different naming standards, *i.e.*, Permutation Importance, Mean Decrease Accuracy, Model Reliance.

The Permutation Importance (PI) provides a score for each feature based on how much the replacement of the feature with noise impacts the predictive power of the estimator. Given the matrix of feature values $\boldsymbol{X}$ and the corresponding response variable $\boldsymbol{Y}$, let $\boldsymbol{X}^{\pi,j}$ be a matrix achieved by randomly permuting the j^{th} column of $\boldsymbol{X}$ and $L(\boldsymbol{Y}, f(\boldsymbol{X}))$ be the loss for predicting the target variable $\boldsymbol{Y}$ from the data $\boldsymbol{X}$ using the model $f(\cdot)$. Algorithm 3 presents the PI feature importance computation steps.

Given that each feature is permuted N times, the feature importance can be mathematically expressed as follows:

$$VI_j^{\pi} = \frac{1}{N} \sum_{\pi} L(\boldsymbol{Y}, f(\boldsymbol{X}^{\pi,j}) - L(\boldsymbol{Y}, f(\boldsymbol{X})), \tag{2.3}$$

Algorithm 3: Permutation Importance algorithm

Input:
Feature matrix $\boldsymbol{X}$
Target vector $\boldsymbol{Y}$
Trained model f
Loss measure $L(\boldsymbol{Y}, f(\boldsymbol{X}))$;
Output: Feature Importance VI_j^{π};

1: **begin**
    `/* Estimate the original model error */`
2: $e_{orig} = L(\boldsymbol{Y}, f(\boldsymbol{X}))$ (e.g. mean squared error)
3: **for** $j = 1, \ldots, D$ **do**
4: Generate feature matrix $\boldsymbol{X}^{\pi,j}$ by permuting feature j in the data $\boldsymbol{X}$. This breaks the association between feature j and true outcome $\boldsymbol{Y}$
        `/* Estimate the error based on the predictions of the permuted data */`
5: $e_{perm} = L(\boldsymbol{Y}, f(\boldsymbol{X}^{\pi,j}))$
        `/* Calculate permutation feature importance */`
6: $VI_j^{\pi} = e_{perm} - e_{orig}$

where N represents the number of permutations applied to each feature. Specifically, the importance of the feature j is given by the increase in loss due to replacing $\boldsymbol{X}_{:,j}$ with values randomly chosen from the distribution of feature j.

Mean Decrease Impurity

Mean Decrease Impurity (MDI) is a tree-specific feature importance method computed as the average of feature importance across all decision trees in an RF model. The structure of the trees has a direct impact on determining the feature importance. As pointed out in Section 2.2.2, the RF algorithm constructs multiple decision trees where each DT is a set of internal nodes and leaves. An internal node n has a pair of child nodes and divides the feature space in a hyper-rectangle R_n with respect to the feature set into two hyper-rectangles corresponding to the left child and right child. For a node n in a tree DT, $N_n = |\{t \in \mathcal{D}^{(DT)} : \boldsymbol{x}_t \in R_n\}|$ denotes the number of samples falling into R_n and their associated response is:

$$\mu = \frac{1}{N_n} \sum_{\forall t | \boldsymbol{x}_t \in \mathrm{R}_n} y_t$$

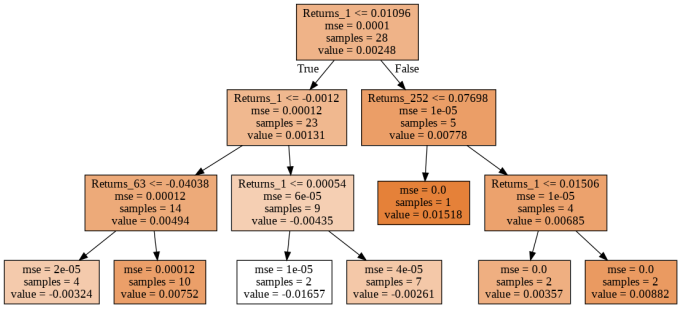


Figure 2.2: Example of a decision trees from a RF regressor with three features (lagged returns of Apple stock). The maximum depth of of the trees is set to 3. The nodes are color coded to reflect the magnitude of the prediction where darker colour represent higher values.

The algorithm splits the data by taking an input feature (from a randomly picked subset of features) and determining which cut-point minimizes some impurity measure of the target variable. The optimal split is determined by maximizing the decrease:

$$\Delta_{\mathcal{I}} = \text{Impurity}_n - \frac{N_{n\,left}}{N_n}\text{Impurity}_{n\,left} - \frac{N_{n\,right}}{N_n}\text{Impurity}_{n\,right}$$

Some popular choices of the impurity measure Impurity_n include variance, Gini index, entropy. Given that thought this work, the center of attention is represented by regression tasks, the variance is considered as impurity measure and it can be expressed as:

$$\text{Impurity}_n = \frac{1}{N_n}\sum_{\forall t|\boldsymbol{x}_t \in R_n}(y_t - \mu)^2$$

Variance minimization occurs when the data points in the nodes have very similar values or when all variables X in the node are locally constant [45]. After determining the optimal cut-point per feature, the algorithm selects the best input feature for splitting, *i.e.*, the feature that would result in the best partition with the lowest variance. Finally, it adds this split to the tree. The feature importance is computed by iterating all the splits generated by a feature. Thus, the feature importance of a feature $\boldsymbol{X}_{:,j}$ with respect to a single tree DT and all M trees in a forest (RF) can be written as:

$$\begin{aligned} MDI_j(DT) &= \sum_{n \in DT, \nu(n) = \boldsymbol{X}_{:,j}} \frac{N_n}{T}\Delta_{\mathcal{I}} \\ MDI_j &= \frac{1}{M}\sum_{m=1}^{M} MDI_j(DT_m) \end{aligned}, \tag{2.4}$$

where $\nu(n) = \boldsymbol{X}_{:,j}$ represents the node (split) where the variable $\boldsymbol{X}_{:,j}$ was used. Thus, MDI measures how much that feature reduced the variance with respect to the parent node. The more the variance decreases, the more significant the input feature is. Figure 2.2 shows a decision tree structure in an RF. In this example, one can appreciate that the most important is Return_1 as it is used 4 times to perform splits of the input samples.

Local Interpretable Model agnostic Explanation (LIME)

LIME was proposed by Ribero *et al.* in [46] and its core idea, though exhaustive, is based on three key concepts:

Model agnosticism - refers to the property of LIME using which it can give explanations for any given supervised learning model by treating as a "black-box" separately. This means that LIME can handle any type of tasks although it limits itself to the supervised learning models.

Interpretability - refers to the property of LIME to provide explanations which make the explained model predictions understandable by humans, therefore trustworthy. As such, LIME's explanations use a data representation (called *interpretable representation*) that is different from the original feature space.

Local explanations - LIME gives explanations that are locally faithful within the surroundings or vicinity of the observation/sample being explained. Although a simple model may not be able to approximate the "black box" model globally, approximating it in the neighborhood of the individual observation may be feasible. In other words, LIME relies on the assumption that every complex model is linear on a local scale.

To formalize the above concepts, let p be the dimension of the original feature space $\boldsymbol{X}$ and let p' be the dimension of the interpretable space X'. The interpretable inputs map to the original inputs through a mapping function $h^y : \boldsymbol{X'} \rightarrow \boldsymbol{X}$, specific to the instance $y \in \boldsymbol{X}$ for which the explanations are sought. Moreover, to obtain observation by observation explanations, LIME gets the locality around the explained observation by using a weight function $w^y : \boldsymbol{X} \rightarrow \mathrm{R}^+$ that gives the most weight to the observations closest to the instance y and the least weight to the observations that are furthest away.

Then, LIME locally approximates a black-box model $f : \boldsymbol{X} = \mathrm{R}^p \rightarrow \mathrm{R}$ around the instance of interest y with a local linear model $g : \boldsymbol{X'} \rightarrow \mathrm{R}'$, such that $g \in G$, where G is a class of potentially interpretable models. Finally, let $\mathcal{L}(f, g, w^y)$ be a loss function that measures how unfaithful g is in approximating f in the locality defined by w^y, and let $\Omega(g)$ be a measure of

complexity of the explanation $g \in G$. In order to ensure both interpretability and local fidelity, LIME must minimize $\mathcal{L}(f, g, w^y)$ while having $\Omega(g)$ be low enough to be interpretable by humans. Thus, the explanation $\xi(y)$ produced by LIME is obtained by the following:

$$\xi(y) = \arg\min_{g \in G} \{\mathcal{L}(g, f, w^y) + \Omega(g)\}.$$

Note that this formulation can be used with different explanation families G, loss functions $\mathcal{L}$ and regularization terms Ω.

2.4 Algorithmic Trading

Two distinct approaches have evolved in active investment: discretionary and systematic (or quant) investing. A discretionary method entails a thorough examination of a limited number of securities. Systematic techniques, on the other hand, rely on algorithms to find investing opportunities across a wide range of assets in a repeatable and data-driven manner. This is because algorithmic trading techniques are guided by signals that indicate when to buy or sell assets in order to outperform a benchmark such as an index. Quantitative strategies show three envolutionary periods:

- In the 1980s and 1990s, signals often emerged from academic research and used a single or very few inputs derived from market and fundamental data. Large quantitative funds such as AQR were founded in this period and start implementing research-based strategies at scale.

- In the 2000s, the pioneering work by Eugene Fama and Kenneth French [47] opened the steam of factor-based investing. Funds used algorithms to identify assets exposed to risk factors like value or momentum to seek arbitrage opportunities.

- The third era is driven by investments in ML capabilities and alternative data to generate profitable signals for repeatable trading strategies. Factor decay is a major challenge: the excess returns from new anomalies have been shown to drop by a quarter from discovery to publication, and by over 50 percent after publication due to competition and crowding [48].

Nowadays, the trading practice follows different objectives when using algorithms to execute rules. That is, short-term trades that aim to profit from small price movements. All strategies aim to exploit relative value opportunities due to arbitrage, through the implementation of long-short positions. However, the concept of arbitrage is fundamental and has been extensively in the financial literature [49, 50].

2.4.1 Statistical Arbitrage as Algorithmic Trading Strategy

Statistical arbitrage trading, or StatArb for short, exploits some statistical patterns in the dynamics of asset prices, thus obtaining, with a high probability, a return larger than the risk-free return. StatArb roots back from pairs trading strategy [51], and was first developed at Morgan Stanley by a quantitative trading group under the lead of Nunzio Tartaglia in the mid-1980s on Wall Street [52]. Pairs trading, a simplified form of StatArb, involves forming portfolios of two related assets with relatively close pricing.

The most common pairs trading model was introduced by Gatev *et al.* [53], and can be described as follows:

1. During the formation period calculate the pair's spread as defined by the authors $S_t = \boldsymbol{A}_t - \boldsymbol{B}_t$, with mean, μ_s, and standard deviation, σ_s.

2. Define the model thresholds: the threshold that triggers a long position, α_L, the threshold that triggers a short position,α_S, and the exit threshold, α_{exit}, that defines the level at which a position should be exited.

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

4. In case α_L is crossed, go long the spread by buying stock $\mathcal{B}$ and selling stock $\mathcal{A}$. If α_S is crossed, short the spread by selling stock $\mathcal{B}$ and buying stock $\mathcal{A}$. Exit position when α_{exit} is crossed.

The simplicity of this model is particularly appealing, motivating its frequent application in the field. However, during the years there were several other approaches have been developed and according to Krauss *et al.* [54] there are five different approaches to identify pairs of assets:

- **Distance approach**: This technique is the most thoroughly explored pair trading framework. Various distance metrics are used throughout the formation period to detect comoving securities. Simple nonparametric threshold criteria are used to trigger trading signals during the trading period. This strategy's main strengths are its simplicity and transparency, which allow for large-scale empirical applications. The key findings show that trading distance pairs can be beneficial in a variety of markets, asset classes, and time durations.

- **Cointegration approach**: In this technique comoving securities are identified trough cointegration tests during the formation period. Simple algorithms are employed to create trading signals during the trading period, with the majority of them based on Gatev *et al.* threshold criterion [55]. The main advantage of these procedures is that the

equilibrium relationship of identified pairings is more econometrically reliable [54].

- **Time series approach**: In the time series approach, generally the authors do not make used of any formation period and rely on pairs identified in previous analyses. Instead, they focus on the trading period and how alternative approaches of time series analysis, such as modeling the spread as a mean-reverting process, might yield optimum trading signals. Mean-reversion is a financial term for the assumption that a stock's price will tend to return to the average price over time. Some asset prices are naturally mean-reverted: commodities, foreign exchange rates, volatility indices, equities - all of those can be modeled with mean-reverting processes, alongside with interest rate and default risk. Facilitated by many hedge-fund managers, creation of the mean-reverting portfolios gave the ability to both pick almost any type of asset for trading and simultaneously rely on rigid mathematical concepts in the decision-making process. The only requirement to be is that the pair chosen has to be correlated or co-moving.

- **Stochastic control approach**: The formation period is neglected, as it is in the time series method. This body of research seeks to determine the best portfolio holdings in the legs of a pair trade in comparison to other available assets. The optimal policy functions for this portfolio problem are determined using stochastic control theory.

- **ML approaches**: This category covers additional pairs trading frameworks with only a limited collection of supporting literature and only a tenuous relationship to previously stated approaches. The ML models and combined predictions approach, the copula approach, and the Principal Components Analysis (PCA) approach are all included in this category.

Traders soon began to think of these "pairs" not as an isolated block to be executed and its hedge, but rather as two sides of the same trading strategy, where profits could be made rather than simply as a hedging tool. These pair trades eventually evolved into several more sophisticated strategies aimed at taking advantage of statistical differences in security prices due to liquidity, volatility, risk, or other fundamental or technical factors. These strategies can be collectively classified as StatArb. These strategies typically tend to make a large number of individual independent trades with a positive expected return, thereby reducing the risk of the strategy. The arbitrage opportunities exist as a consequence of the market inefficiency and the profits are realized by taking trading positions when the mispricing of the assets correct themselves in the future. Moreover, because the spread between assets' prices

is considered to be uncorrelated with market returns, pairs trading and, by extension, StatArb, are market-neutral strategies.

On a high level, StatArb implies automatically trading a set of assets that construct a portfolio [56] and comprises two phases: (i) the *scoring* phase, where each asset is assigned a relevance score, with high scores indicating assets that should be held long and low scores indicating assets that are candidates for short operations; and (ii) the *risk reduction* phase, where the assets are combined to eliminate, or at least significantly reduce the risk factor [49, 57]. StatArb strategies, as they become more widely used and automated, tend to push the market toward greater efficiency. As arbitrage opportunities between assets arise, they are quickly eliminated through the use of these strategies. As a result, StatArb can lead to a more liquid, more stable market.

However, besides being extremely appealing to the individual traders or large trust funds, StatArb has also caused some major problems. For instance, the collapse of Long Term Capital Management back in 1998 almost left the market in ruins. StatArb algorithms have also been blamed in part for the "flash crashes"[4] that the market has started to experience over the past decade.

2.5 Strategy Backtesting

Algorithmic trading stands apart from other types of investment classes because we can more reliably provide expectations about future performance from past performance. The process by which this is carried out is known as **backtesting**. Backtesting simulates an algorithmic strategy based on historical data with the goal of producing a set of trading signals. Each trade will mean to be a 'round-trip' of two signals: long or short actions and their corresponding entry/exit criteria (used interchangeably with opening and closing positions, respectively). A **long** action which consists of buying the stock, and then selling it before the market closes (exit criteria). Conversely, a **short** action consists of selling the stock (using the mechanism of the *uncovered sale*), and then buying it before the market closes.

The trading signals will have an associated profit or loss. The accumulation of this profit/loss over the duration of the strategy backtest will lead to the total profit and loss (also known as the 'P & L' or 'PnL'), performance results that will offer an outlook of how the strategy will generalize to new market conditions.

[4] A flash crash is an event in electronic securities markets wherein a rapid sell-off of securities leads to a negative feedback loop that can cause dramatic price drops over a matter of minutes.

For strategy backtesting, throughout this dissertation common approach for validating time-series data in finance is used, namely the *walk-forward* validation, which consists of splitting the study period into overlapping training periods, also known as *In-Sample period (IS)* and non-overlapping test (trading) periods or *Out-of-Sample period (OOS)*, as shown in Figure 2.3. Each tuple IS and OS data constitutes a walk. The IS data is further split into two chunks: training and validation. The example depicts values for the closing

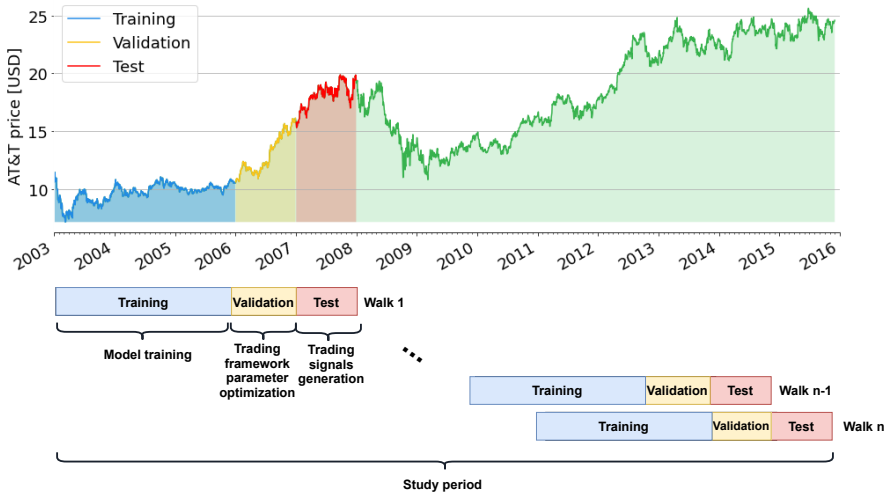


Figure 2.3: Illustration of walk-forward procedure, considering a study period starting from January 2003 to January 2016. On the y-axis it is presented the development of the closing price for AT&T (asset) across time and how this overlaps, in time, to each walk and each training,validation and test period, respectively.

price of AT&T (asset) over the whole study period. In this example, four years of IS and a year of OOS are considered. The period of IS is further split into three years of training, one year of validation and one year of test.

It is important that the test examples are not used in any way to make choices about the ML model, or trading strategy parameters for that matter. For this reason, no example from the test set can be used in the validation set. Therefore, the validation set is constructed from the IS data. Specifically, the IS is split data into two disjoint subsets. One of these subsets is used to learn the parameters. The other subset is our validation set, allowing for the parameters to be updated accordingly. The subset of data used to fit the ML model is still typically called the training set, even though this may be confused with the larger pool of data used for the entire training process. The subset of data used to guide the selection of parameters is called the validation set. Notably, the walk-forward optimization is different from the cross-validation procedure used for ML learning generalization error esti-

mation and hyper-parameter search, although they share the same principle, *i.e.*, learning data-set is different from validation set and the two should not be overlapping.

2.6 Evaluation Metrics

Ultimately, the goal of active investment management is to generate alpha, defined as portfolio returns in excess of the benchmark used for evaluation. The fundamental law of active management postulates that the key to generating alpha is having accurate return forecasts combined with the ability to act on these forecasts [58]. This section will explain the evaluation metrics used to assess the performance of the novel approaches proposed within this thesis. In this regard, the evaluation metrics can be split into two categories: metrics that asses the goodness of fit of the ML models, as well as metrics that assess the financial performance of a trading strategy from a risk-return perspective.

2.6.1 Evaluation metrics for Machine Learning

(i) Mean squared error (MSE) - one of the most popular goodness of fit measure for ML models for a continuous dependent variable, such as in our case. The MSE is defined as:

$$MSE(f, X, Y) = \frac{1}{T}\sum_{t}^{T}(\widehat{y}_t - y_t)^2 = \frac{1}{T}\sum_{t}^{T} r_t^2,$$

where $\widehat{y}_t$ is the forecast of the model f at time t (see also Section 2.1 for further details on notations), and r_t is the residual for the observation at time t. Thus, MSE can be seen as a sum of squared residuals. As the measure weighs all differences equally, large residuals have a large impact on MSE. Thus, the measure is prone to outliers. For a "perfect" model, which predicts (fits) all y_t exactly, MSE value would be 0.

2.6.2 Evaluation metrics for Finance

(i) Cumulative Return or *Portfolio Return* or *Net Profit* is the effective gain or loss realized by an investment. The return is expressed in %.

(ii) Annual return - represents the return on an investment generated over a year and calculated as a percentage of the initial amount of investment. Formally, it is expressed as:

$$\text{annual return} = 100 \times \left(\left(\prod_{i=1}^{n} (1 + R_n) \right)^{1/n} - 1 \right),$$

where n represents the number of years, and R_n the corresponding return at year n. High values are to be sought.

(iii) Maximum drawdown (MDD) - the maximum amount of wealth reduction that a cumulative return has produced from its maximum value over time. Formally, the drawdown is defined in two steps: first, the maximum up to time t is identified, *i.e.*, $M_t = \max_{u \in [0,t]} R_u$, where R_t are the cumulative returns up to time t. The drawdown up to time t (as a percentage) is, then: $DD_t = \frac{R_t}{M_t} - 1$. The maximum drawdown is thus the largest drop in cumulative returns from its maximum rolling over a given time frame:

$$MDD_t = \max_{u \in [0,t]} DD_u.$$

Smaller absolute values are an indication of lower incurred risk.

(iv) Sharpe Ratio (SR) - also known as the *Sharpe index*, the *Sharpe measure*, and the *reward-to-variability ratio*,it is the average return earned in excess of the risk-free rate per unit of volatility or total risk. Volatility is a measure of the price fluctuations of an asset or portfolio; the greater the value of the Sharpe Ratio, the more attractive the risk-adjusted return;

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p},$$

where R_p is the actual or expected portfolio return; R_f is the Risk-free rate; σ_p is the standard deviation of the portoflio's excess return

2.7 Software libraries

During the various research works the following toolkits have been employed for the development and evaluation of the proposed solutions.

Scikit-learn[5] is a Python library that provides implementations of supervised and unsupervised algorithms, and metrics to evaluate results.

Numpy[6] is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms,

[5] `https://scikit-learn.org/stable/`

[6] `https://numpy.org`

basic linear algebra, basic statistical operations, random simulation and much more.

Pandas[7] is a Python library that provides easy-to-use data structures and data analysis tools. It allows to efficiently manage great matrices of data and perform classic operations that are typically adopted on tables (e.g., *join* and *selection* of rows based on their values).

Matplotlib[8] is one of the most popular Python library. It is a comprehensive library for creating static, animated, and interactive visualizations in Python.

[7] `https://pandas.pydata.org/`
[8] `https://matplotlib.org/`

Chapter 3

Related Work

> *"If I am walking with two other men, each of them will serve as my teacher. I will pick out the good points of the one and imitate them, and the bad points of the other and correct them in myself."*
>
> **Confucius**

The prediction of the stock market is one of the most challenging problems in time series forecasting research and, hence, the interest in performing efficiently such a task from the academic finance community is growing. Despite the establishment of the EMH by Malkiel and Fama in [47], which was later revised in [59], according to which financial markets follow random pathways and therefore are unpredictable, in recent years, clear evidence was built that ML techniques are capable of identifying (non-linear) structures in financial market data. Such applications of ML and neural networks in finance have been presented and analyzed in several works, such as [60, 61, 11, 62, 63, 64].

In this chapter, the work on financial forecasting and algorithmic trading is reviewed, by considering four perspectives (1) the input variables investigated in each model, (2) techniques utilized to build the prediction model, (3) the trading strategies used with a focus on StatArb, and finally (4) XAI techniques applied to trading systems with an emphasis on StatArb.

3.1 Machine Learning and Financial Applications

This section discusses the work of general ML applications to financial forecasting in the sense that their main goal is either predicting stocks prices (regression) or their movement (classification) without taking into consideration their financial application, either perform forecasting on a limited

number of stocks and their framework does not include portfolio creation or optimization. The first stage in stock market forecasting is the selection of input variables. As pointed out in Chapter 2 Section 2.1, the two most common types of features that are widely used for predicting the stock market are fundamental indicators and technical indicators. Moreover, Kumar *et al.* in their study, point out that 39% of the work reviewed use a set of TI as technical indicators [64], followed by stock prices with 31%, 11% fundamental indicators (FI), and only 7% a combination of TI or FI.

The work in [65] uses standardized technical indicators to forecast the rise or fall of market prices with the AdaBoost algorithm, which is used to optimize the weight of these technical indicators. Chien *et al.* [66] build a genetic algorithm (GA) based associative classification rules (ACR) classification model for discovering trading rules from technical indicators which generate buy or sell signals. Weng *et al.* [67] employed both FI and TI to provide inputs to the model. The reasoning of why the combination of the two types of input features is so scarce is because the fundamental analysis is more appropriate for low-frequency financial forecasting as fundamental data is available quarterly and highly dependent on companies' financial report releases. Instead, the technical analysis relies on data with higher frequency, *e.g.* daily up to minute frequency.

The presence of noise and outliers may result in poor prediction accuracy of forecasting models. The data must be prepared such that it covers the range of inputs for which the network is going to be used [68]. In the literature, authors have used various kinds of normalization techniques for transforming data from one scale to another. In [69, 70] use data preprocessing techniques such as MinMax scaling in order to minimize the impact of noise in stock market data and to increase the accuracy. The work in [71] used Principal Component Analysis to reduce the dimensionality of the data, Discrete Wavelet Transform to reduce noise, and an optimized Extreme Gradient Boosting to trade in financial markets. In [72] the authors presented an integrated independent component analysis (ICA) to remove noise contained in the dataset. The output of ICA containing less noise is used as input of back-propagation neural network (BPN) to create a forecasting model. In [73, 74], the authors applied ARIMA to a preprocessed time series data in order to predict prices. In [75] proposed a hybrid approach, based on Deep Recurrent Neural Networks and ARIMA in a two-step forecasting technique to predict and smooth the predicted prices.

The solutions of ML in finance have a wide variety of algorithms *e.g.* *Naive Bayes*, *Decision Trees*, *Support Vector Machines*, *Gradient Boosting* etc. In [76], the authors investigated how to predict stock indices by using SVR to learn the relationship among several technical indicators and the index price. The grid search method was used to optimize the SVR model parameters. Patel *et al.* in [77] evaluated the efficiency of the daily closing

price predictions performed by SVM, RF and neural networks.

Besides individual ML approaches, in the literature, we can find several hybrid approaches. That is because each of these methodologies has its own set of flaws, such as local optima, overfitting, and the difficulty in choosing multiple parameters, among others, all of which have a direct impact on forecasting accuracy [78]. The hybrid approaches usually translate into a cascade of intelligent methods of feature extraction and forecasting, or simply parametric and non-parametric forecasting models. For example, in [79] propose a cascade of parametric and non-parametric methods such as linear neural networks, Multilayer perceptrons, and SVR to forecast option prices. The work in [80] validated an extension of SVR, called *Twin Support Vector Regression*, for financial time series forecasting. Similarly, in [81, 77], a two-stage fusion approach is proposed for predicting CNX Nifty and S&P Bombay Stock Exchange Sensex from Indian stock markets. Specifically, the first stage uses an SVR, whereas the second one exploits, in turn, Artificial Neural Networks, RF and SVR. Ten indicators have been selected as input to the prediction models. In [82], the authors combined economic theories with ML models. Specifically, they combined parametric option pricing models such as Black–Scholes option pricing model, Monte Carlo option pricing model, and finite difference method with SVR and extreme learning machine-based regression models.

To mitigate the noise problem, a successful approach has proven to be the use of ensembles. They have demonstrated superior predictive performance compared to individual forecasting models, hence their notable success in different domains such as credit scoring [83], sentiment analysis [84], power systems control [85] or natural calamities forecasting [86]. The literature also reports many approaches that exploit a set of different classification or regression algorithms [87, 88] whose results are combined according to a certain criterion *e.g.*, *full agreement*, *majority voting*, *weighted voting*, among others). An ensemble process can work in two ways: by adopting a *dependent framework* (*i.e.*,, in this case, the result of each approach depends on the output of the previous one), or by adopting an *independent framework* (*i.e.*,, in this case, the result of each approach is independent) [89]. In this regard, the work in [90] proposed a novel multiscale nonlinear ensemble learning paradigm, incorporating Empirical Mode Decomposition and Least Square Support Vector Machine with kernel function for price forecasting. The work in [91] fits the same SVM classifier multiple times on different sets of training data, increasing its performance to predict new data. Authors of [92] combined results of bivariate empirical mode decomposition, interval Multilayer Perceptrons, and interval exponential smoothing method to predict crude oil prices. Other interesting approaches using ensembles are the use of multiple feed-forward neural networks [93], multiple artificial neural networks with model selection [94], among others. In [67] the authors developed an expert system based on an ensemble of machine learning methods such as artificial neural network,

SVR, GB, and RF regression that utilizes the features from multiple online sources to predict short term stock prices. The key idea of this work is to incorporate data from various sources such as historical stock prices, technical indicators, articles about given stocks published in news, Google search about given stocks, and the number of visited Wikipedia pages for related stocks. After data collection and preprocessing, the authors have applied principal component analysis to extract the effective features which are used as input to the forecasting model. Results showed that the use of features from online sources improved the accuracy of ensemble methods.

Several other interesting studies have been carried out in the literature, such as a comparison of deep-learning technologies to price prediction [95], the use of deep learning and statistical approaches to forecasting crisis in the stock market [96], the use of reward-based classifiers such as Deep Reinforcement Learning [97], among others.

Also, in [98], the editors bring together a collection of works discussing innovative approaches of data-science and ML for economic and financial applications.

3.2 Machine Learning and Statistical Arbitrage Applications

As pointed out in Chapter 2 Section Section 2.4 the precursor of StatArb is pair trading. Moreover, one of the key ingredients of StatArb is identifying pairs of stocks that *move together*, *i.e.*, exhibit similar behavior in time. For this reason, over the years, a plethora of statistical and econometric techniques have been developed to analyze financial data. For example, the pivotal work of Gatev *et al.* [53] is distance-based, models based on co-integration [99] or models based on stochastic spread [100], to name a few. However, the results obtained by Machine Learning approaches have proved very promising. Works such as [101, 102, 103] explored the application of Artificial Neural Networks to forecast the spread change for three famous spreads. The authors of [104] explored the potential of applying Deep Reinforcement Learning in a Pairs Trading setup and obtained satisfactory results in comparison with more traditional methods.

Extending the vanilla pairs trading strategy to StatArb we find the work of Avellaneda and Lee [49]. Over time, in the context of StatArb there we find several approaches such as sophisticated, nonlinear models from the various domains such as physics, mathematics [105]. In [54] presents a comprehensive study of StatArb strategies and future directions. Among them, we find ML approaches that show promising results. Figure 3.1 presents the most influential works using ML models in the StatArb domain. The figure depicts

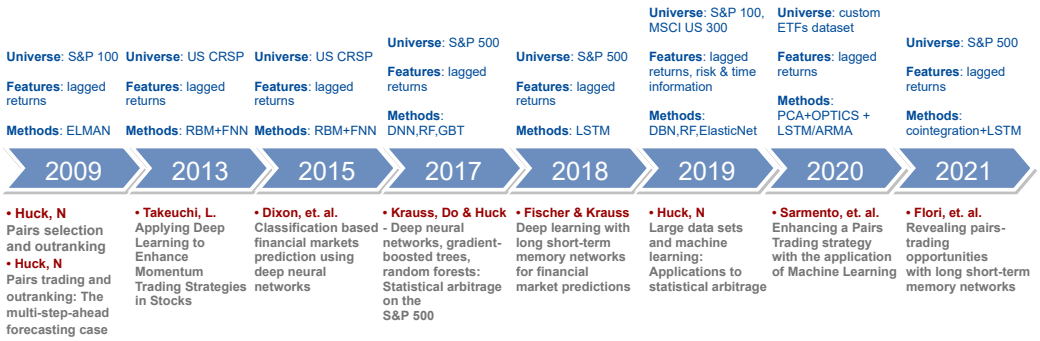


Figure 3.1: A road map of the forecasting methods and asset universes in StatArb trading along the years (ELMAN - ELMAN Recurrent Neural Network, RBM - Restricted Boltzman Machines, FNN - Feedforward Network, DNN - Deep Neural Network, Random Forest - RF, GBT - Gradient Boosting Trees, LSTM - Long Short Term Memory, DBN - Deep Belief Network, PCA - Principal Component Analysis, OPTICS - clustering algorithm).

the roadmap for the models behind StatArb, corresponding input features, and the considered universe (set of assets).

Starting chronologically, [106] is the earliest proposal of experimental statistical arbitrage system based on Neural Network Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models for modeling the mispricing-correction mechanism between relative prices composing a pair. Another pioneering work that considers coupling machine learning with StatArb is [107]. Here, the author proposes a StatArb system that entails three phases: forecasting, ranking, and trading. For the forecasting phase, the author uses an Elman recurrent neural network to perform weekly predictions and anticipate return spreads between any two securities in the portfolio. Next, a multi-criteria decision-making method is considered to outrank stocks based on their weekly predictions. Lastly, trading signals are generated for top k and bottom k stocks, considering a variable k. This work is later extended in [108], where the approach is evolved by introducing a multi-step-ahead forecast. Both studies consider constituents of the S&P100 Index on a period spanning from 1992 to 2006. Although these approaches also consider regression, they lack scalability as their application is limited to 100 stocks, and in the case of broader indexes such as S&P500 or Russell 1000, would become computationally intractable.

In [109], the authors use deep neural networks for classification and as features cumulative returns. The approach computes the probability that one stock outperforms the cross-sectional median return of all stocks in the holding month. Next, all stocks are ranked according to the forecasted prob-

ability, and then, the trading signals are constructed based on the top decile of predictions, which is bought long, and flop decile, which is sold short. The stock universe used is the U.S. CRSP and the study period spans from 1965 until 2009. The works in [110, 111] adopt a similar strategy. The former, in a high-frequency setting with five-minutes binned return data, whereas, in the latter, the authors construct a similar classification problem using cumulative returns as input features and employ models like deep neural networks, RF, gradient boosted trees, and three of their ensembles. They validate their study using $S\&P500$ Index constituents on a period ranging from 1992 to 2015, with a trading frequency of one day. In a few words, the approach computes the probability that one stock outperforms the cross-sectional median return of all stocks in the holding period. The authors conducted daily trading first, by ranking the stocks according to the forecasted probability, and then, by generating the daily trading signals on the top-decile of stocks that were bought long, and the flop-decile stocks that were sold short. Later, the authors extend their work in [112] by using a Long Short-Term Memory network for the same prediction task. This enhanced approach outperforms memory-free classification methods.

A different approach in terms of features, in the context of StatArb, is presented in [113]. The author used 4 types of one-hot encoded features on a period from January of 1993 to June 2015, with two forecasting/holding periods (*e.g.* one and five days). The used machine learning algorithms are RF, Elastic Net, and Deep belief networks in a classification setup. The most salient finding in this work is that increasing the number of features does not translate into increased performance.

In [114, 115], the authors took a different approach for predicting returns of S&P500, where the used features are stock tweets information. The aim is to unveil how the textual data influence the stocks' future returns. Knoll *et al.* used factorization machines and SVM. The proposed system performs prediction in a 20 minutes frequency over two years: from January 2014 to December 2015. The selection of flop and top stocks is made at the formation period based on the algorithm's performance evaluation (*i.e.*, lowest root relative squared error) and trading signals are generated based on Bollinger bands. In [115] engineer domain-specific features along with three categories, i.e., directional indicators, relevance indicators, and meta-features and use RF to predict "tweet returns" probability of being above or below the median return of all tweets in a day. Where the authors define tweet return as the stock's return between the tweet release and 120 minutes after the release.

The work in [116] examined an approach of stocks clustering to complement investor practices for the identification of pairs-trading opportunities among co-integrated stocks. The authors proposed to predict the trend of price or return gaps between each stock and its peers.

3.3 Explainable Machine Learning for Financial Applications

Chronologically covering prior works on XAI techniques applied to financial forecasting, the work by [117] represents the starting point. The authors used a hybrid feature selection method (filter and wrapper) for a classification task on stock trends. To prove its performance, they applied the method on 17 different stocks obtaining an enhanced prediction performance, i.e., 87.3% accuracy on average. The proposed method was a hybrid one, since it used the F-Score to initially filter the features, and then it used a Supported Sequential Forward Search, that is a technique created for SVM to sequentially perform a forward search for the best features to add to the final feature set. Moreover, in the mentioned work, the method selected 17 features, out of a total of 30 features, as the best ones that significantly improved the overall classification accuracy.

In [118], the authors aimed at forecasting the DAX index closing price and devised a pipeline with four layers: data preprocessing, feature selection, modeling, and finally a reporting and scenario planning. The authors used cross-correlation as a feature selection technique, and out of 20 features, they chose 13 as appropriate for predicting the stock price.

The work in [119] introduces a feature selection approach based on trading rules generated from technical indicators and genetic programming. The authors defined the pipeline as an automated investment system envisaged to identify the right moments for executing buying and selling orders. The authors tested its performance by applying it to six shares on two study periods: February 2013 to February 2015 and July 2015 to July 2016. They identified that the feature selection strategy had a positive impact on 5 out of 6 shares.

In [120] the authors proposed to use Variational Autoencoders as an unsupervised feature reduction mechanism, followed by a recursive feature elimination technique, to further decrease the number of features in the input feature set. The selected features constituted the input for three different classifiers: Gradient Boosting, SVM, and two variants of Long short-term memory (LSTM) networks, with and without attention. The proposed ensemble approach was used to forecast the hourly direction of eight different stocks of the Borsa of Istanbul on a study period from 2011 to 2015.

In [121] the authors proposed an "instability index" strategy for feature selection based on the features ranks variance. Their investigation focused on identifying a convergence point for feature importance stability. To this end, for the same time series, the authors trained various RF models having different configurations, and computed the feature importance for each of these models to infer the "instability" index. Continuing in the realm of XAI

techniques applied to financial forecasting, [122] proposed an approach to indicate any links between news and stock behavior. Towards this objective, the authors employed an explainable ML model based on decision trees to discover keywords relevant to stock return trends.

When it comes to explainable methods applied to StatArb applications,the work in [111] should necessarily mentioned. The authors used an ensemble of classifiers composed of RF, Gradient Boosted Trees, and Deep Neural Networks to predict stock return daily trends, for the stocks of the S&P500 index. To understand the key features (lagged stock returns) of the long-short strategy, and how they contributed to the trading strategy results, the authors presented the features' importance as given by RF and Gradient Boosted Trees.

In [112], Fischer *et al.* used an LSTM network for the same prediction task. This enhanced approach outperformed memory-free classification methods such as RF. To explain the results of the StatArb strategy, the authors used the coefficients of a Carhart regression [123] applied to the returns and first and second-order time-series statistics and, in doing so, they untangled the dependence between the market regime, stock behavior, and the results obtained by the long-short trading strategy.

N. Huck proposed in [113] proposed the use of four predefined feature sets and constructed the target similar to the previous works, i.e., by computing the probability that stocks outperform their peers in the S&P100 or S&P300 indexes. The author used various ML models such as Deep Belief Networks, Elastic Net, and RF. Concerning the XAI techniques, the author also inferred the feature importance by assessing the performance of the algorithmic trading strategy given the four groups of features. Also, the author reported the feature importance per feature sub-groups according to RF and permutation importance and noted that increasing the number of features does not translate into enhanced financial performance.

In [115], the authors used MDI to analyze feature importance and establish which of the handcrafted feature tweet-related features have a "heavier" weight than others. They found out that directional features based on financial dictionaries have the highest importance.

Chapter 4

Statistical Arbitrage with Ensembling and Dynamic Asset Selection

"The test of a first-rate intelligence is the ability to hold two opposed ideas in mind at the same time and still retain the ability to function."

F. Scott Fitzgerald

Outline

This chapter presents a general approach for risk-controlled trading based on ML and StatArb. Section 4.1 formulates the problem, Section 4.2 provides an overview of the architecture for the proposed approach, while Section 4.3 provides further details of the methodology and describes a specialization of the general framework presented here. Section 4.4 discusses the experiments carried out and their results. Section 4.6 presents some concluding remarks.

StatArb relies on identifying pairs of stocks whose prices have historically moved together (*i.e.*, their prices are correlated) over a specific period. Once having identified the stocks that behaved in a "similar" way in the past, traders short the outperforming stock (*i.e.*, betting on a value of a security to fall) and buy long the under-performing one *assuming* an equilibrium relationship between the two securities (*i.e.*, stocks) and a reversion occurring in price spread.

Therefore, one can readily identify two challenges when implementing a StatArb trading strategy. First, correctly identifying pairs of assets that exhibit similar behavior and determining the point in time when the prices start moving away from each other and open trading positions. This insufficiency, coupled with the large number of assets involved in trading, has led to the

need of introducing a forecasting component and, consequently, the rise of the ML models [11, 107]. Furthermore, since investors often exhibit irrational behavior assets' price changes differently. This leads to financial data containing a large amount of noise, jump, and movement. As a consequence, such time series are highly non-stationary in time and notoriously unpredictable [25]. To mitigate the noise problem, a successful approach has proven to be the use of ensembles. In the literature, one can find several implementations of StatArb that apply classification to construct the trading portfolio [111, 109]. In this regard, this work applies predicted return in building the portfolio by buying long assets with the highest expected return and short selling assets with the lowest expected return. Although regression in the context of financial predictions poses more challenges [124, 125], it allows for a more *granular ranking*, without reference to any balance point.

This chapter proposes and discusses a general approach for risk-controlled trading based on ML and StatArb. The approach employs an ensemble of regressors for which provides three levels of heterogeneity: (i) *forecasting algorithms*, as the approach can be implemented with any state-of-the-art forecasting algorithms; (ii) *data level*, as the models in this work are trained with information of constituents of financial time series with a diversified feature set, considering not only lagged daily prices return but also a series of technical indicators; and (iii) *diversified models*, as the models are trained using either data from individual assets or aggregated assets' data of the same industry. Finally, in the approach of this dissertation, after ranking the assets in descending order, a *dynamic asset selection* is proposed. The mechanism looks at the past and influences the ranking by removing assets with "bad" past behavior. Then, the strategy buys (performing long operations) the flop k assets and sells (performing short operations) the top k assets. Also, it is proposed one possible instance of trading strategy has been configured and it considers intra-day operations and trades the constituents of the well-known S&P500 Index. The regressors employed for such an instance are the state-of-the-art ML algorithms presented in Chapter 2: RF, LGB, SVR, and the widely known statistical model, ARIMA. To validate the chosen configuration, the performance of the proposed approach is evaluate from both return and risk performance perspectives. The comparisons against the baselines clearly illustrate the superiority of the proposed methodology in performing the forecast.

4.1 Problem statement

Using StatArb as a trading strategy, the proposed general approach aims at solving a risk-controlled trading problem. In this regard, it leverages ML to identify possible sources of profit and balance risk at the same time. At a very

high level, the proposed approach takes various assets' time series as an input and outputs the subset of assets to invest in, together with the appropriate trading signals, *i.e.*, long or short. The proposed trading strategy follows these three steps:

i *forecasting* - the StatArb is tackled as a regression problem, investigating the potential of forecasting price returns for each of the assets in a selected assets collection, on a target trading day.

ii *ranking* - based on the anticipated price returns, the assets are ranked in descending order. Next, to *balance the risk* due to inaccurate predictions, the "bad" assets are pruned based on their past behavior, thus the *dynamical asset selection* yields a reorganized ranking of the assets.

iii *trading* - the last step issues trading signals for the top k and flop k assets from the ranked set of assets obtained in the previous step.

With respect to the Task, Experience, and Performance measure discussion (Section 2.2.2), in the proposed StatArb trading workflow, the ML algorithms perform a regression task in supervised manner investigating the potential of forecasting intra-day stock price returns. The rationale behind choosing a regression task lies in the fact that it allows refined trading, as opposed to the used classification methods as previously mentioned in Chapter 3. Moreover, by clearly separating the forecasting phase from the ranking and trading signal generation, the system can be thought as agnostic of any alterations that can happen over time to the set of stocks and, hence, tackle the survivorship[1] and data snooping biases[2]. Additionally, in this study, the ensemble of forecasters can be viewed as one with a high degree of diversity provided by the input features and dissimilar prediction models, each of them capturing different structures in the financial market data.

4.2 Overview

Figure 4.1 presents the architecture for such a risk-controlled trading approach. Let s_i be an individual asset in the asset collection S. It starts by collecting historical raw financial information for each asset s_i in the selected asset collection S. To note that *only* information available before the trading day d is used. Using the historical data, the *diversified feature set* is generated. The feature set is denoted by $\mathcal{F}_d^{s_i}$, that is used as input to each regressor m in the regressors pool $\mathcal{M}$.

[1]Survivorship bias refers to the tendency to exclude from performance studies the failed companies because they no longer exist [126].

[2]Data snooping refers to the use of data mining to uncover misleading relationships in data [127].

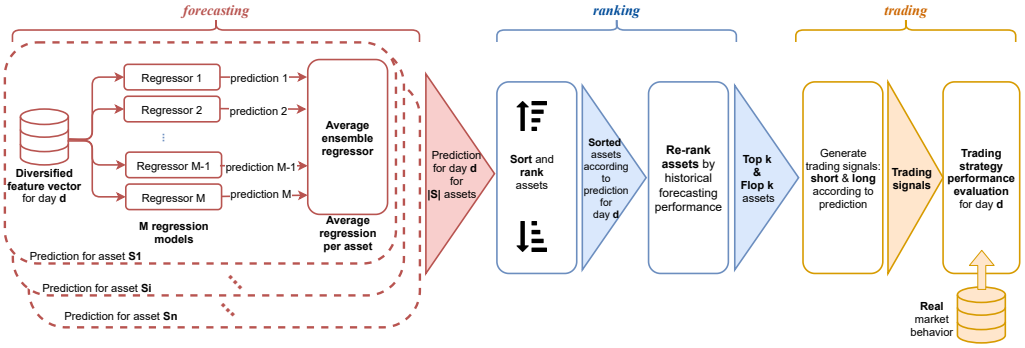


Figure 4.1: Architecture of the proposed general approach for risk controlled trading

In the *forecasting* step, each trained model m makes its prediction, $o_d^{s_i,m}$ for day d and asset s_i. Then, their results are averaged by a given ensemble method, to obtain a final prediction output denoted by $o_d^{s_i,ENS}$.

In the *ranking* step, the *assets are sorted* in descending order based on the forecast output $o_d^{s_i,ENS}$. This means that the top reprents assets whose prices are expected to increase, and at the bottom assets whose prices are expected to drop. Assets at the top and at the bottom of the sorted list represent the most suitable candidates for trading. After the ranking is performed, the *dynamic asset selection* step is introduced: from this pool of assets, the ones that do not satisfy a prediction accuracy higher than a given threshold ε in a past trading period are discarded, therefore the ranking is rearranging accordingly.

The *trading* step consists of selecting the top k (winners) and flop k (losers) assets and issue the corresponding trading signals: k long signals for the top k stocks and k short signals for the bottom k stocks. These selections are repeated for every day d in the trading period.

Finally, the performance of the proposed architecture is evaluated by using a back-testing strategy [49].

4.3 Methodology

4.3.1 Data and Feature Engineering

As previously mentioned, in this chapter it is instantiated one example out of the proposed general framework by using the S&P500 index as a reference dataset. By reference to Figure 3.1, only three studies [112, 111, 116] focus on such a large pool of companies. As stated by the authors in [111], the S&P500 constituents represent the leading 500 companies in the U.S. stock market, and at the same time, the large-capitalization segment. These characteristics

make these companies highly attractive to investors, and as a consequence, they are very challenging for any trading strategy.

For each stock, daily raw financial information is collected, such as *Open Price*, *High Price* in the day, *Close Price*, *Low Price* in the day, and *Volume* of stocks traded during the day (OHCLV variables). Based on this information, two different types of features are created:

Lagged intra-day price returns (LR) - for a stock s_i and a trading day d, we compute the intra-day $LR^{s_i}_{d-j}$ expressed as:

$$LR^{s_i}_{d-j} = \frac{closePrice^{s_i}_{d-j} - openPrice^{s_i}_{d-j}}{openPrice^{s_i}_{d-j}}, \tag{4.1}$$

where j is a given time-lag. We considered $j \in \{1, \ldots, 10\}$, thus yielding for each trading day d, a series of 10 historical price returns as it follows:

$$\mathbf{LR}^{s_i}_d = [LR_{10}, LR_9, LR_8, LR_7, LR_6, LR_5, LR_4, LR_3, LR_2, LR_1] \tag{4.2}$$

For brevity, the subscript d indicating the day d and the superscript s_i indicating the stock are omitted. In this study, the lagged returns in the sense of Box-Jenkins models [31] are used. They consider the nearest past (j is maximum 10), thus, each of the prediction models is fed with the information about asset behavior in the last days. Glancing at Figure 3.1, historical price returns are the preferred choice in financial studies. With respect to previous works, in this dissertation it proposed the use of the intra-day price returns. This decision was motivated by recent studies [128] where the authors decompose the returns in overnight returns and intra-day price returns. They point out that after-hours trading happens much more seldom than trading while markets are open. Moreover, the pre-open auctions on the U.S. stock exchanges only average one to four percent of median daily volume, as pointed out by the same authors. Also, trading in the first half-hour of the day (the interval in which the open price is measured) is significantly less than the volume one observes intra-day, particularly near or at the close.

Technical Indicators (TI): Throughout this dissertation a set of technical indicators is used as summarized in Table 2.1. For this second type of feature the following vector is constructed:

$$\mathbf{TI}^{s_i}_d = [EMA(10), \%K, ROC, RSI, ADO, MACD, \%R, D(5), D(10)] \tag{4.3}$$

Technical analysis and by extension technical indicators are valuable statistical tools through which investors extensively use to make their investment decisions. This fact partially motivated the reasoning behind our choice for this second type of features. Moreover, the center of attention is predicting the price movement range and also its direction and each of the technical indicators has its own inherent opinion about the asset price movement. When

feeding the learning models with such data, the forecasting models receive trend information as perceived by each of the individual technical indicators. Several other financial studies use technical indicators in conjunction with ML algorithms to predict individual stock direction movement [129, 77] or closing price [81].

Given that the problem defined in this work entails an ML supervised setup, for regressors' training, the associated *target value* (label) is generated as well, *i.e.*, $y_d^{s_i}$ which is the intra-day price return for the current day.

$$y_d^{s_i} = \frac{closePrice_d - openPrice_d}{openPrice_d}. \tag{4.4}$$

4.3.2 Forecasting Models

In the proposed instance of general trading approach, three different state-of-the-art machine learning models are used as well as the widely known statistical model, ARIMA. The rationale of this choice relies on the following criteria:

1. robustness to noisy data and over-fitting;
2. diversity amongst models in the final ensemble;
3. computational efficiency;
4. adoption of such models in the scientific community for similar tasks.

LGB is considered to be more prone to overfitting and sensitive to noise [130]. This fact is attributed to how the trees are built, *i.e.*, sequentially and dependent on the residuals of the previously built tree. On the other hand, boosting reduces bias by adding each new tree in a sequence such that the errors missed by the preceding tree are also captured. It also reduces variance by combining many models. The specific implementation employed in this dissertation uses a series of unique features like observations and features sampling, which bring great improvement in terms of training time. Its adoption in the scientific community for financial forecasting is scarce [131], but by contrast, one can find this algorithm in the leading positions in machine learning competitions platforms[3].

RF belong to a category of ensemble learning algorithms, as pointed out in Chapter 2. This learning method is the extension of traditional DT techniques where random forests are composed of many de-correlated decision trees. Such a de-correlation is achieved by bagging and by

[3]https://www.kaggle.com/

random feature selection. These two techniques make the RF algorithm robust to noise and outliers. In the academic literature corpus, one can find that RF has been extensively used in financial forecasting [81] and StatArb applications [107, 111].

SVR is considered to be as efficient as ANN, with a substantial body of literature supporting this idea [132, 133, 134]. Moreover, through its regularisation parameter, it is less susceptible to over-fitting. Also, by using the *kernel trick* it allows building-in expert knowledge about the problem via engineering the kernel. SVR is defined by a convex optimization problem (no local minima) for which there are efficient computation methods. For these reasons, SVM has a high adoption in the financial forecasting domain. Authors in [77, 81] successfully used SVR in financial forecasting tasks, to name a few.

ARIMA is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables and predicts future values by examining the differences between values in the series instead of the actual values. Through the years, in the literature corpus, one can find several applications of ARIMA for financial forecasting since the market's behavior is considered to exhibit repetitive patterns [47] and ARIMA can be considered a memory-enabled model.

4.3.3 Model Training

In standard statistics, fitting a simple model on data can be done without the sensitive choice of a hyper-parameter and the model's parameters are estimated from the data, *e.g.* using the maximum-likelihood criterion. However, in high-dimensional settings, some form of regularization is needed. Choosing the amount of regularization is a typical bias-variance problem. But, in general, the best trade-off is a data-specific choice, governed by the statistical power of the prediction task, or differently put, it is governed by the amount of data and the signal-to-noise ratio. In this context, given that each of the assets exhibits different behavior in time and incurs a different amount of noise, the same set of hyper-parameters do not apply to all the stocks. As such, a hyper-parameter tuning is embedded in the model training phase.

In addition to that, for each of the machine learning models, two variants of *data input* are proposed:

1. Data specific to each stock s_i in our stocks pool S, resulting in a model for each stock.

2. Data resulting after aggregating information from stocks pertaining to the same industry sector as given by the Global Industry Classification

Standard (GICS), thus resulting in a model for each industry. To be noted that industries contain a different number of stocks, but their content does not overlap. That is, one stock cannot be part of multiple industries at the same time.

This last variant of the model was encouraged by previous work [53], where some portfolios were restricted to only include stocks from the same industry. In the spirit of the aphorism "a rising tide lifts all boats", companies in the same industry tend to have similar behavior, and the returns of stocks tend to follow each other [49].

As such, one is faced with the problem of selecting for each of the stocks and each type of algorithm (*i.e.*, LGB, RF, and SVM) the best combination of hyperparameters as well as input data (*i.e.*, per stock or per industry) or feature types (*i.e.*, LR or TI) in order to obtain the model with the best predictive power for that specific stock. Standard machine learning practice advocates to measure predictive power using cross-validation [36]. This means that the available data is split into a training set, used to train the model, and a validation set, unseen by the model during training and used to compute a prediction error. By using such a scheme, one expects to obtain a better estimate of the model's out-of-sample predictive performance. To properly account for the time dependence of observations, empirical research based on traditional time series models typically reverts to validation schemes that keep the temporal order of observations between training and validation sets. However, our goal to select the best hyperparameters and best features in the same step is slightly different from standard practice. As such, the machine learning algorithms selection are modified as presented in Algorithm 4.

Our algorithm receives as input: (i) the set of stocks, S_I, with their associated financial historical data, (ii) the chosen machine learning algorithm a to be trained, (LGB, RF, or SVM) and its corresponding set of hyperparameters. The output is represented by a set of trained regressors, $\mathcal{R}$, each corresponding to each stock in S_I. The algorithm starts by acquiring historical raw financial data (*ohclv*) for each of the stocks, for the entire training period (Lines 2-6). Financial information will be timestamped, thus the temporal order of observations is kept as well as any operation on data is performed using the timestamps. The historical dataset is split into two portions: training and validation sets (Line 7), where the former is used for model training and the latter for model selection. Then, it proceeds with the stock level model training Line 8. For each type of feature, the optimal hyperparameter constellation is selected by employing an inner-cross-validation. The same process is applied at the industry level (Line 13). Therefore, to forecast the return of each stock, 4 models are created: 2 models (per asset) using TI and LR, that use data of a single stock; and 2 models (per industry) using TI or LR as features, that use data of all assets associated to that indus-

Algorithm 4: Model selection

Input:
S_I, set of stocks in industry I
$\mathcal{D}$, training dates (calendar days)
a,$\mathcal{P}_{set}$, set of hyperparameters for the machine learning algorithm a
Output: Pool of regressors $\mathcal{R} = \{r_1 \dots r_{|S_I|}\}$

begin
  for s_i ***in*** S_I **do**
    $\mathbf{LR}^{s_i} \leftarrow$ COMPUTELR($ohclv$) using $ohclv \in \mathcal{D}$
    $\mathbf{TI}^{s_i} \leftarrow$ COMPUTETI($ohclv$) using $ohclv \in \mathcal{D}$
    $\mathbf{LR}^{I} \leftarrow$ APPEND($\mathbf{LR}^{s_i}$)
    $\mathbf{TI}^{I} \leftarrow$ APPEND($\mathbf{TI}^{s_i}$)
  Split $\mathcal{D}$ into $\mathcal{D}^{train}$,$\mathcal{D}^{val}$
  for s_i ***in*** S_I **do**
    for f ***in*** $\{\mathbf{LR}^{s_i}, \mathbf{TI}^{s_i}\}$ *using data in* $\mathcal{D}^{train}$ **do**
      `/* using inner cross-validation */`
      $p^* \leftarrow$ HYPERPARAMETEROPTIMIZATION($\mathcal{P}_{set}, f$)
      $m_f^{s_i} \leftarrow a.train(p^*, f)$
      Save $m_f^{s_i}$
  for f ***in*** $\{\mathbf{LR}^{I}, \mathbf{TI}^{I}\}$ *using data in* $\mathcal{D}^{train}$ **do**
    `/* using inner cross-validation */`
    $p^* \leftarrow$ HYPERPARAMETEROPTIMIZATION($\mathcal{P}_{set}, f$)
    $m_f^{I} \leftarrow a.train(p^*, f)$
    Save m_f^{I}
  $\mathcal{R} \leftarrow [\,]$
  for s_i ***in*** S_I **do**
    foreach $m \in \{m_{LR}^{s_i}, m_{TI}^{s_i}, m_{LR}^{I}, m_{TI}^{I}\}$ **do**
      `/* Compute loss for model` m `*/`
      $\mathcal{L}_{s_i,m}^{val} \leftarrow$ COMPUTELOSS($\mathcal{D}^{val}$)
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\text{model with lowest } \mathcal{L}_{s_i,m}^{val}\}$
  return $\mathcal{R}$

try, and, in turn, forecast one asset at a time. The second phase consists of choosing among the 4 models trained in the previous step, *i.e.*, the one with the highest predictive power. Hence, using the validation set, for each stock and each model the loss $\mathcal{L}_{s_i,m}^{val}$ is computed, which, in this case, is the mean squared error (MSE) between the forecast and the ground truth. Then, the best model out of the four is chosen by taking into account the lowest MSE.

ARIMA is a class of statistical models that captures temporal structures in time series data. Its methodology of training differs from machine learning algorithms in the sense that ARIMA is not designed for multivariate input and as a consequence for ARIMA the series of lagged returns is used. Furthermore, ARIMA is designed for forecasting one-step out-of-sample forecast. Hence the methodology that was developed in this dissertation, performs a one-step out-of-sample forecast with re-estimation, *i.e.*,, each time the model is re-fitted to build the best estimation model.

In conclusion, for each stock, 3 types of machine learning algorithms are trained (LGB, RF, and SVM) each of them for 2 levels of data (stock level and industry level), and 2 types of features (*i.e.*, LR and TI). This yields: 3 types of algorithms $\times$ 2 data level $\times$ 2 types of features = 12 models for each stock. Algorithm 4 selects the best model per machine learning algorithm type, resulting in 3 models. To this pool of regressors, for each stock, the ARIMA model is added. To note that ARIMA is most suitably used with univariate time series (equivalent to stock level and lagged returns features).

4.3.4 Ensembling

The final ensemble entails for each stock s_i three machine learning models (*i.e.*, LGB, RF, SVR as given by our model selection/cross-validation phase) and a statistical model ARIMA. Their forecasted output is averaged. The benefit of this approach is that if the errors of each model are sufficiently independent, they average out: the average model performs better and displays much less variance [36, 135]. Several works in literature validated the power of averaging decisions. In [136, 137] it is revealed, by empirical evaluation, that averaging ensembles performs well in practice, especially when compared to ensembles that use more complex weighting strategies. Moreover, according to [138], simple ensembles of forecasters such as averaging decisions outperform sophisticated combination methods in empirical applications. Finally, in [111], the authors show that, for the same purpose, a simple weighted ensemble performs better when compared to performance-based or rank-based ensembles. The previous works justify our choice of model output averaging. As such, the outputs of each of the selected models, *i.e.*, LGB, RF, SVR, and ARIMA are averaged to the final output, $o_d^{s_i,ENS}$.

4.3.5 Ranking and Dynamic Asset Selection

The final step in our trading framework comprises ranking the assets by their predicted return to be able to issue the appropriate trading signals. Additionally, to mitigate the risk of bad forecasting performance, I propose a stock pruning mechanism by performing a *dynamic asset selection* strategy. For a stock $s_i \in S$, given its past forecastings $o_t^{s_i,ENS}$, and also its past real values $y_d^{s_i}$ in a predefined look-back period T, a modified version of the mean directional accuracy [139, 140] is computed as follows:

$$MDA_{s_i,T,d} = \frac{1}{T} \sum_{t=d-1}^{d-T-1} \mathbf{1}_{sgn(o_t^{s_i,ENS})==sgn(y_t^{s_i})}, \tag{4.5}$$

where d is the current trading day, T is the look-back length, and $\mathbf{1}_P$ is the indicator function that converts any logical proposition P into a number that is 1 if the proposition is satisfied, and 0 otherwise, $sgn(\cdot)$ is the sign function. In other words, $MDA_{s_i,T,d}$ compares the forecasted direction (upward or downward) with the realized direction of the return and yields the probability that the forecasting model can detect the correct direction of returns for a stock s_i on a given timespan T prior to the day d. Such a component is used on a limited look-back period as a consequence of proven studies [141] that stocks exhibit behavior with short periods of significant returns predictability ('pockets'). These periods are interspersed with long periods with little or no evidence of return predictability.

Such a component introduces a new step in the StatArb pipeline: after performing the forecast, the companies are ranked by their forecasted daily price returns. From this pool of stocks, the companies that do not satisfy a certain standard are discarded. The standard means a prediction accuracy higher than a given threshold ε in a past trading period. This process leads to rearranging the ranking accordingly.

4.4 Experimental Setup

In this section, for reproducibility purposes, the parameters of the proposed approach are detailed. The detailing follows along the main steps of the proposed trading framework and presents the backtesting methodology, model training and forecasting technical aspects, ranking and dynamic asset selection, trading execution, and finally we discuss the used baselines. The framework is back-tested by choosing as study period a timeline starting from March 2003 to January 2016. The back-testing experiments consist in running the signals through historical data together with the estimation of forecasting hyper-parameters, signal evaluations and portfolio re-balancing [49]. As presented in Chapter 2, a common approach for validating time-series data in

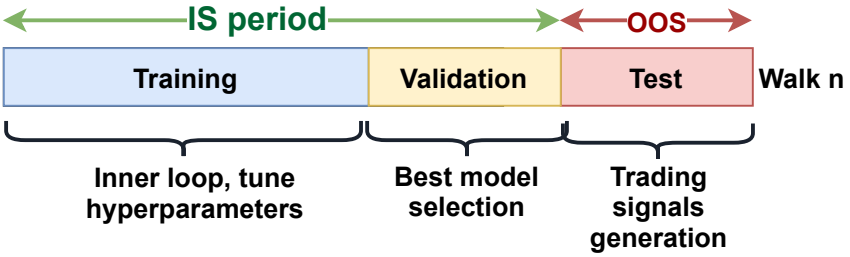


Figure 4.2: Example of a Walk's temporal splits and their functional role

finance is used, namely the *walk-forward* validation, which consists of splitting the study period into overlapping IS periods and non-overlapping test (trading) periods (for a more detailed discussion see Section 2.5).

In this experimental setup, four years of training and a year of test are considered. The IS period is further split into three years of training for individual regressors training and one year of validation. With respect to the walk forward setup presented in Chapter 2, the validation period is used for the best model selection for each of the stocks (procedure presented in Section 4.3.3), as shown in Figure 4.2. Running the experiments having as study period March 2003 to January 2016, yields 9 walks.

4.4.1 Model Training

For model training and parameter tuning, a 10-fold *TimeSeriesSplit* is used, in combination with a *GridSearch*, both using scikit-learn implementations [142]. In this experimental scenario, the temporal linking between the observation at day $d-1$ and d is taken into account to compose the same bunch of training and validation sets.

Table 4.1: Hyperparameters of the forecasting models.

Model Type	Hyperparameters
LGB	n_estimators=100, num_leaves= $\{70, 80, 100\}$, max_depth=8, colsample_by_tree=0.8, learning_rate=0.1
RF	n_estimators=range(50,500,25), min_samples_leaf=3, learning_rate=0.01
SVR	$C \in \{8, 10, 12\}$, tol=1e-5, epsilon=0.1, gamma=$\{0.01, 0.5\}$
ARIMA	$p \in \{1, \ldots, 5\}$, $d \in \{1, \ldots, 5\}$, $q \in \{0, \ldots, 5\}$

Table 4.1 shows the forecasting models hyperparameters and in the following paragraphs, the reasoning behind the choices made are discussed.

LGB Compared to the traditional gradient boosting techniques, LGB would grow the tree vertically (*i.e.*, leaf-wise tree-growth until the maximum depth is reached), whereas other alternative algorithms extend their structures horizontally (*i.e.*, level-wise tree-growth), which makes an effective method for LGB in processing large-scale and high-dimensional data, on the one hand, but more prone to over-fitting, on the other hand. To control this behavior I defined the maximum depth levels of the tree, *max_depth*, to 8. I chose to vary the *num_leaves* parameter in the set $\{70, 80, 100\}$, achieving a balance between a conservative model and a good generalization. The feature selection is restricted by a parameter *colsample_by_tree* set at 0.8 of the total number of features, which can be thought of as a regularization parameter. The work in [36] suggests a learning rate lower than 0.1, so I set it to 0.01 to account for a better generalization over the data set.

RF When using RF the larger the size of the forest (the number of trees), the better the convergence of the generalization error. But, a higher number of trees or a higher depth of each tree induces computations costs, therefore a trade-off must be made between the number of trees in the forest and the improvement in learning after each tree is added to the forest. I opt to vary the number of trees by ranging *n_estimators* from 50 to 500 with a 25 increment, similarly to [113]. Random feature selection operations substantially reduce trees' bias, thus I set *min_samples_leaf* to 3 of the total number of features in a leaf. The learning rate is set to 0.01.

SVR introduced in Chaper Section 7 solves an optimization problem that involves two parameters: the regularization parameter, C, and the error sensitivity parameter ϵ. C controls the trade-off between model complexity and the number of non-separable samples. A lower C will encourage a larger margin, whereas higher C values lead to a hard margin [38]. Thus, I set the search space in $\{8, 10, 12\}$. Parameter ϵ controls the width of the ϵ-insensitive zone and is used to fit the training data. A too high value leads to flat estimates, whereas a too-small value is not appropriate for large or noisy data-sets. Therefore, it is set to 0.1. The work in [143] suggests that the γ value of the kernel function should vary together with C, and higher values of C require higher values for γ too. Therefore, a smaller search space in $\{0.01, 0.5\}$ is set.

ARIMA The parameters of the ARIMA model are defined as follows: p - he number of lag observations included in the model, also called the lag order, d - the number of times that the raw observations are differenced, also called the degree of differencing, and finally q - he size of the moving average window, also called the order of moving average.

The set ARIMA parameters are: lag order $p \in \{1, \ldots, 5\}$, the degree of differencing $d \in \{1, \ldots, 5\}$, the size of the moving average window $q \in \{0, \ldots, 5\}$.

4.4.2 Ranking and Dynamic Asset Selection

As pointed out in Section 4.3.5, the dynamic asset selection requires a series of parameters: the accuracy threshold ε, and rolling window length, T. The threshold value is set to $\varepsilon = 0.5$ and for the rolling window, different lengths are considered, $T = \{30, 40, 60\}$. These choices are based on findings in [144] where the authors noticed that MDA can efficiently capture the interdependence between asset returns and their volatility (hence forecast-ability) when using intermediate return horizons, e.g. two months. The threshold value has been set to $\varepsilon = 0.5$ as advised in [107] for a similar scenario.

4.4.3 Trading Execution and Portfolio Construction

As stated throughout this Chapter, each day, are perform $2 \times k$ operations, k long and k short operations. The number of pairs to be traded is fixed to $k = 5$, based on the findings in similar works [112, 111] where higher k values lead to a decrease in portfolio performance both in terms of returns and risks. The trading session is set as intra-day, meaning that the positions are opened at the beginning of the training day and close them at the end of the day. In other words, the portfolio is rebalanced daily. As the authors in [114], I assume transaction costs of 0.4% daily.

4.4.4 Baselines

To assess the value added by both our model selection strategy (**ENS**) and dynamic asset selection strategy (**ENS-DS**), they are benchmarked against two statistical arbitrage trading baselines: one based on cumulative five day return (**5-DAY**), and S&P500 buy-and-hold strategy (**Buy-and-hold**). These last two methods are well-established quantitative strategies, and largely used as baselines to evaluate the profitability of other investment approaches. They are described in the following:

1. **5-DAY** - Daily, we sort the set of stocks according to the 5-day cumulative return prior to the trading day, in ascending order. At the top we would find the stocks with the most negative cumulative returns and at the bottom stocks with the most positive 5-day cumulative return. We go long for top k and short for flop k stocks in the sorted list, building an equal weight portfolio. This approach has been considered in prior works such as [112]. Data, trading execution, and portfolio construction

are the same as the other strategies (*i.e.* we open the trading position for $k = 5$ at the beginning of the day and close them at the end of the day and the portfolio is rebalanced daily).

2. **Buy&Hold** This strategy buys in 2007 and holds the S&P500 exchange-traded fund (SPY security) during the whole backtesting period, *i.e.* until January 2016. This passive strategy runs without any trading signals. Such a baseline is widely recognized in literature as a valuable benchmark (e.g. [114, 145]).

4.4.5 Implementation details

With respect to the computational cost of our approach, one may notice that the most intensive step is represented by the training stage. Indeed, in terms of *Big O* notation, this step entails an $O(s \times h)$ cost for each sequential model considered, thus proportional to the number of stocks s and model hyperparameters h.

4.5 Results

The results are presented from three perspectives which include: (i) predictive results of the individual regressors and their resulting ensemble under different training setups; (ii) return evaluation before and after transaction costs, (iii) exposure to common risk factors by analyzing risk metrics, and (iv) comparisons against state-of-the-art trading strategies.

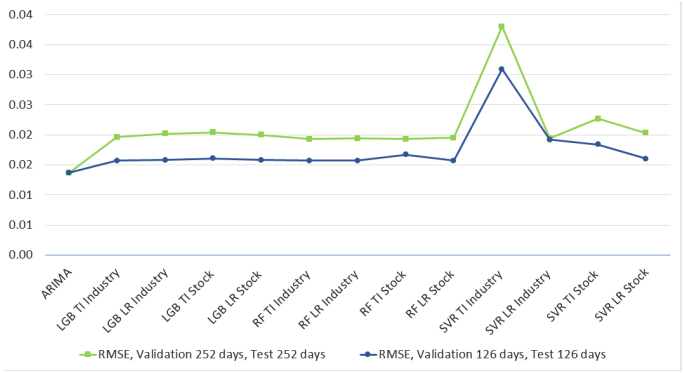
4.5.1 Model Selection Performance

The section contains a comparative study of the performances obtained by the models when using different walk-forward setups (for reference, see Figure 2.3). Specifically, the development period is fixed to 3 years and for validation and test two different lengths (in days) are chosen. For the first setup, the period of study is set to range from January 2003 to January 2016, with 3 years of training, 1 year (252 days) of validation, and 1 year of test, which amounts to 9 walks. For the second setup, the study period ranges from July 2003 to January 2016, with 3 years of training and a reduced validation and test periods to 6 months (126 days), thus resulting in 18 walks. The models corresponding to the two setups will be assessed in an overlapping trading/test period, *i.e.,* March 2007 to January 2016.

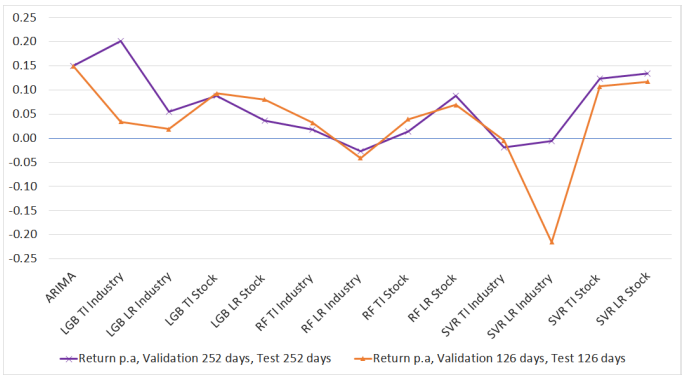
The reported models' performances entail the goodness on fit in terms of root mean square error (RMSE) between the realized return and forecasted return as average across companies and each walk. Also, for each of the models, the portfolio performance is shown in terms of annual returns (Return

Figure 4.3: Model performances in terms of RMSE, Return p.a., and MDD, developed from 2007 to 2016 when varying the validation and test period lengths

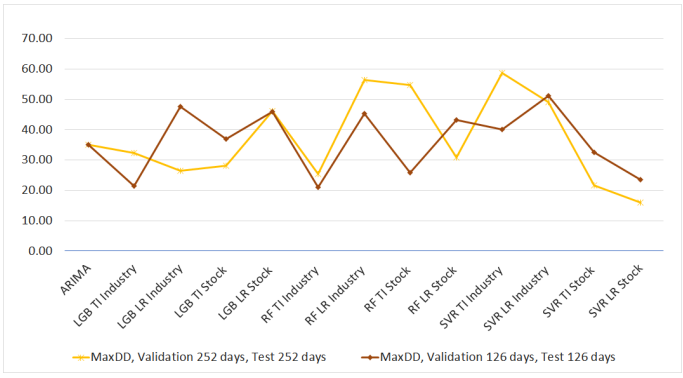
(a) RMSE



(b) Return p.a.



(c) MDD expressed in percentages



p.a.) and risk metrics (MDD), when using a strategy that trades $k = 5$ pair. The maximum drawdown (MDD) quantifies the maximum amount of wealth reduction that a cumulative return has produced from its maximum value

over time. Figure 4.3 depicts the obtained results and Table 4.2 summarizes the performances of the models resulting after the model selection phase (for each stock) and their corresponding ensemble.

Table 4.2: Models performances resulting after model selection phase and their corresponding ensemble (ENS). Each model is a combination of type of features and data level for each stock. The trading period is March 2007 and January 2016.

Method	Validation length and Test length	RMSE	Return p.a.	MDD[%]
ARIMA	-	0.0137	0.1493	35.04
RF	Validation 252 days, Test 252 days	0.0193	0.0761	23.23
LGB	Validation 252 days, Test 252 days	0.0195	0.1796	33.29
SVR	Validation 252 days, Test 252 days	0.0197	0.1835	28.90
ENS	Validation 252 days, Test 252 days	0.0190	0.3126	13.76
RF	Validation 126 days, Test 126 days	0.0157	0.0192	40.85
LGB	Validation 126 days, Test 126 days	0.0157	0.1109	28.55
SVR	Validation 126 days, Test 126 days	0.0159	0.1165	33.45
ENS	Validation 126 days, Test 126 days	0.0155	0.0877	31.78

In terms of RMSE, Figure 4.3a shows a consistent decrease when the length of validation and test decreases. This can be attributed to the decreasing number of samples. Model SVR with data at industry level data and TI as features exhibits the worst performance in both setups. At the opposite pole, one finds ARIMA with the lowest RMSE. In terms of returns (Figure 4.3b) the best performing model is the LGB with data at industry level and TI as features in a year of validation and another year of test setup. The second best performing is ARIMA with annual returns close to 1.5%. The riskiest strategy (Figure 4.3c) is the SVR with data at industry level data and TI as features. Table 4.2 reveals for each forecasting algorithm the predictive performances and risk-return characteristics, respectively. With the reduction of validation length a deterioration in performance can be observed here as well, both in terms of return and MDD. The most surprising result it is represented by the small returns (0.0877) of the ensemble obtained by averaging models with a 6 month validation period. The ensemble is outperformed by most of the models. As a concluding remark, according to Table 4.2, the following statement can be made: the ENS with 1 year of validation and 1 year of test setup is the best performing model.

4.5.2 Return Characteristics Assessment

As presented in Section 4.5.1 the best model performance is achieved when using the following setup: 3 years of development, 1 year for validation, period that serves the purpose for model selection and 1 year of test where the result are recorded and the proposed approach performance is assessed. Given these considerations, the subsequent analyses are made under this setup.

Table 4.3: Return characteristics of StatArb strategies compared to the baselines (5-DAY and Buy&Hold) before transaction costs over a period between March 2007 to January 2016. Portfolio was constructed using $k = 5$ pairs. The best return performances are highlighted in bold.

	ARIMA	RF	LGB	SVR	ENS	ENS-DS $T = 30$	ENS-DS $T = 40$	ENS-DS $T = 60$	5-DAY	Buy & Hold
Return p.a	0.1493	0.0761	0.1796	0.1835	0.3126	0.3192	**0.3330**	0.3147	0.2155	0.0755
Excess return p.a.	0.1422	0.0695	0.1723	0.1762	0.3045	0.3111	**0.3247**	0.3066	0.2068	0.0680
Standard dev p.a.	0.2188	0.1776	0.1648	0.1380	0.1624	**0.1606**	0.1606	0.1606	0.2522	0.2227
Mean	0.0006	0.0004	0.0007	0.0007	0.0011	0.0012	**0.0012**	0.0011	0.0009	0.0004
Min	-0.0934	-0.0816	-0.0846	-0.0912	**-0.0754**	**-0.0754**	**-0.0754**	**-0.0754**	-0.1256	-0.0984
Q1	-0.0047	-0.0041	-0.0034	-0.0033	-0.0036	-0.0036	**-0.0034**	-0.0036	-0.0049	-0.0048
Median	0.0004	-0.0002	0.0005	0.0004	0.0006	0.0007	0.0007	0.0006	0.0003	**0.0008**
Q3	0.0057	0.0042	0.0046	0.0044	0.0050	0.0050	0.0049	0.0050	**0.0057**	0.0062
Max	0.1126	0.1387	0.1045	0.0577	0.1006	0.1006	0.1006	0.1006	**0.1830**	0.1452
Skewness	0.3613	1.4405	0.4219	-0.1524	0.8356	1.0659	1.1033	1.0121	**1.7274**	0.2227
Kurtoisis	9.4476	23.6159	17.1415	13.2800	15.1069	15.0316	14.9208	14.8477	**25.9812**	13.3642
Standard error	0.0003	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0003	0.0001
t-statistic	2.2093	1.4863	3.2157	3.8264	5.1979	5.3566	5.3385	5.2898	2.6648	1.9690

Tables 4.3 and 4.4 report daily return characteristics from March 2007 to January 2016, before and after the transaction costs, respectively. As expected, the equal weighted ensemble, ENS, and the dynamic asset selection, ENS-DS, outperform both baselines and the underlying individual regressors. The annual returns are almost four times the level of the Buy&Hold and roughly two times the return of individual regressors, (*e.g.*, LGB). The same ratios are maintained even after the transactions costs. Furthermore, one can observe a mean daily return of 0.11% for the ENS and 0.12% for ENS-DS ($T = 30, T = 40$) before the transaction costs, that is trice the market. After the transaction costs are applied, the returns deteriorate but are still considerably higher than the baselines. Analyzing the statistical moments,

Table 4.4: Return characteristics of the StatArb strategies compared to the baselines (5-DAY and Buy&Hold) after transaction costs over a period between March 2007 to January 2016. Portfolio was constructed using $k = 5$ pairs. The best return performances are highlighted in bold.

	ARIMA	RF	LGB	SVR	ENS	ENS-DS $T = 30$	ENS-DS $T = 40$	ENS-DS $T = 60$	5-DAY	Buy & Hold
Return p.a	0.0569	0.0410	0.1032	0.0924	0.2198	0.2268	**0.2278**	0.2228	0.1579	0.0755
Excess return p.a.	0.0503	0.0346	0.0964	0.0856	0.2103	0.2192	**0.2202**	0.2153	0.1507	0.0681
Standard dev p.a.	0.1931	0.1775	0.1647	0.1381	0.1624	0.1606	**0.1606**	0.1607	0.2519	0.2227
Mean	0.0003	0.0002	0.0004	0.0004	0.0008	**0.0009**	**0.0009**	0.0008	0.0007	0.0004
Min	-0.0936	-0.0817	-0.0848	-0.0916	**-0.0756**	**-0.0756**	**-0.0756**	**-0.0756**	-0.1257	-0.0984
Q1	-0.0047	-0.0042	-0.0037	-0.0037	**-0.0038**	**-0.0038**	**-0.0038**	**-0.0038**	-0.0051	-0.0048
Median	0.0001	-0.0003	0.0002	0.0000	0.0005	0.0006	0.0005	0.0005	0.0001	**0.0008**
Q3	0.0052	0.0040	0.0044	0.0041	0.0049	0.0048	0.0049	0.0049	0.0055	**0.0062**
Max	0.0830	0.1385	0.1043	0.0575	0.1005	0.1005	0.1005	0.1005	**0.1827**	0.1452
Skewness	-0.0863	1.4347	0.4205	-0.1378	0.8461	1.0756	1.1136	1.0221	**1.7294**	0.2227
Kurtoisis	9.7357	23.6643	17.1581	13.2533	15.1040	15.0343	**15.5535**	14.8518	**25.5818**	13.3642
Standard error	0.0003	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0003	0.0001
t-statistic	1.1363	0.9319	2.0127	2.1021	4.5401	4.6844	4.6711	4.6199	3.2354	1.9690

one can observe that apart from SVR all the strategies expose a positive skewness, which indicates a longer tail for gains.

Last but not least, the statistical significance of returns is assessed by performing a Newey-West t-statistics [146] with the null hypothesis that the mean return is equal to zero (the critical value at 5% significance level is 1.9600). The test resulted in the returns being statistically significant before and after transaction costs.

4.5.3 Risk Exposure Assessment

Tables 4.5 and 4.6 offer an in depth analysis of risk incurred by the trading strategies, before and after transaction costs, respectively.

Firstly the tail risk is analyzed. According to Value at Risk (Var 1%) before transaction costs (Table 4.5), ARIMA is the riskiest strategy of all, with -4.1% exceeding both baselines. At the opposite pole, one finds the ensemble strategies, with -2.6% for ENS, and approximately -2.3% for ENS-DS. After the transaction costs (Var 1%, Table 4.6) the same picture is maintained, but now, ARIMA (-3.80%) is almost on par with 5-DAY baseline (-3.78%). Compared to [111], that reports values between -5.9 and -6.9 percent af-

Table 4.5: Risk assessment of the trading strategies before transaction costs compared to the baselines (5-DAY and Buy&Hold) over a period between March 2007 and January 2016. Portfolio was constructed using $k = 5$ pairs. The best performances are highlighted in bold.

	ARIMA	RF	LGB	SVR	ENS	ENS-DS $T = 30$	ENS-DS $T = 40$	ENS-DS $T = 60$	5-DAY	Buy & Hold
Var 1%	-0.0418	-0.0297	-0.0306	-0.0225	-0.0262	**-0.0220**	-0.0229	-0.0228	-0.0377	-0.0354
Var 5%	-0.0181	-0.0129	-0.0126	-0.0105	-0.0111	-0.0110	**-0.0106**	-0.0111	-0.0179	-0.0171
CVar 1%	-0.0569	-0.0458	-0.0444	-0.0457	-0.0393	**-0.0370**	-0.0403	-0.0372	-0.0624	-0.0512
CVar 5%	-0.0327	-0.0241	-0.0236	-0.0208	-0.0206	**-0.0199**	-0.0202	-0.0201	-0.0318	-0.0290
Downside dev. p.a.	0.1460	0.1129	0.1090	0.1094	0.0987	0.0960	**0.0954**	0.0960	0.1514	0.1347
Sharpe Ratio	0.7167	0.4663	1.0468	1.2454	1.7185	1.7677	**1.8575**	1.7452	0.8539	0.3152
Sortino Ratio	1.1171	0.7887	1.6403	1.9690	2.8908	3.0303	**3.0404**	2.9850	1.4922	0.7812
MDD	35.04%	23.23%	33.29%	28.90%	13.76%	11.63%	**11.46%**	13.69%	36.68%	55.19%
Calmar	0.4261	0.3278	0.5394	0.6350	2.2711	2.7445	**2.9048**	2.2982	0.5876	0.1646
Omega	1.1633	1.1131	1.2508	1.2888	1.4230	1.4220	**1.4636**	1.4348	1.2248	1.1139

Table 4.6: Risk assessment of the trading strategies after transaction costs compared to the baselines (5-DAY and Buy&Hold) over a period between March 2007 and January 2016. Portfolio was constructed using $k = 5$ pairs. The best performances are highlighted in bold.

	ARIMA	RF	LGB	SVR	ENS	ENS-DS $T = 30$	ENS-DS $T = 40$	ENS-DS $T = 60$	5-DAY	Buy & Hold
Var 1%	-0.0380	-0.0299	-0.0308	-0.0227	-0.0262	**-0.0223**	-0.0231	-0.0231	-0.0378	-0.0354
Var 5%	-0.0154	-0.0130	-0.0129	-0.0107	**-0.0112**	**-0.0112**	**-0.0112**	**-0.0112**	-0.0183	-0.0171
CVar 1%	-0.0535	-0.0459	-0.0446	-0.0359	-0.0394	**-0.0372**	-0.0375	-0.0375	-0.0625	-0.0512
CVar 5%	-0.0290	-0.0242	-0.0238	-0.0191	-0.0208	**-0.0201**	-0.0203	-0.0203	-0.0324	-0.0290
Downside dev. p.a.	0.1341	0.1138	0.1107	0.0926	0.1006	0.0980	**0.0974**	0.0980	0.1539	0.1347
Sharpe Ratio	0.3511	0.2793	0.6411	0.6640	1.2711	1.3118	**1.3145**	1.2936	0.6812	0.3152
Sortino Ratio	0.5515	0.4901	1.0100	1.0576	2.1142	2.2279	**2.2319**	2.1852	1.1560	0.7812
MDD	35.92%	30.09%	36.53%	50.56%	28.42%	27.09%	**26.06%**	27.10%	37.15%	55.19%
Calmar	0.1584	0.1362	0.2826	0.1827	0.7769	0.8407	**0.8730**	0.8225	0.4250	0.1646
Omega	1.0794	1.0694	1.1503	1.1492	1.3003	1.3104	**1.3106**	1.3043	1.1727	1.1139

ter transaction costs, our strategies have significantly lower values. Under the conditional value at risk (CVar 1%, Table 4.5), positions have slightly changed, in the sense that ARIMA (-5.7%) incurs the highest risk out of

regression models, but lower than 5-DAY strategy (-6.2%).

Sharpe ratio is a risk metric, defined as the excess return per unit of risk measured in standard deviations [15]. A Sharpe ratio greater than one usually signifies a portfolio of superior performance as opposed to a portfolio with a Sharpe ratio less than one. In general, a portfolio with a larger Sharpe ratio will outperform one with a smaller ratio. In Table 4.5, it can be noticed that Sharpe ratio started from 1.72 for the simple ensemble and turned into 1.85 for the proposed ENS-DS ($T = 40$), before transaction costs.

By the same token, another metric that measures the reward-to-risk ratio is Sortino Ratio which considers the risk expressed as downside deviations. By checking results in Tables 4.5 and 4.6, one can realize that downside deviations are less expressed for the proposed strategies. This naturally leads to a more favorable Sortino ratio: for ENS 2.89 before transaction costs and 2.11 after transaction costs, and for for ENS-DS strategies approximately 3 before transaction costs and approximately 2.2 after transaction costs. This amounts to twice the values for 5-DAY baseline (1.49 before transaction costs and 1.16 after transaction costs). MDD offers an outlook on how sustainable an investment loss can be, where lower is better. Also, for this metric, it must be noted the better performance of ENS-DS strategies compared to the Buy&Hold and 5-DAY baselines. ENS produces 13.76% value decreasing to 11.46% for ENS-DS, T=40 (Table 4.5), that is less than one fourth of the Buy-and-Hold (55%) and one third compared to the 5-DAY. After transaction costs (Table 4.6), as expected, the values of MDD have increased for both ENS (28.42%) and ENS-DS, T=40 (26.06%), nevertheless lower than the baselines. With respect to Calmar Ratio, before applying the transaction costs (Table 4.5), its value has increased from 2.27 for ENS to 2.90 for ENS-DS, T=40. After transaction costs (Table 4.6), the ENS strategy registers a value of 0.77, whereas for the ENS-DS, $T = 40$, the value is 0.87, compared to 0.16 of the Buy & Hold strategy. Calmar Ratio scales annualized returns by the value of MDD and determines how many average annual returns are needed to recover from a maximum drawdown. Thus, the ENS-DS strategy needs 1.15 years to recover from the maximum drawdown, whereas the Buy & Hold would need approximately 6 years. The Omega metric, introduced by [147], divides expected returns into two parts: gains and losses or returns above the expected rate (upside) and those below it (downside). In simple terms, Omega can be considered as the ratio of upside (good) returns relative to downside (bad) returns. Before transaction costs (Table 4.5), this ratio has increased also from 1.42 (ENS) to 1.46 in our proposed approach (ENS-DS, T=40). After transaction costs (Table 4.6), results show the same positive trend, *i.e.*, from 1.3003 for simple ensemble, ENS, to 1.3106 for ENS-DS, T=40. This translates into higher chances of achieving daily positive returns.

Based on the aforementioned results, the following conclusions can be drawn. In terms of risk, the dynamical asset selection strategy, ENS-DS,

improves the results of simple ensemble, ENS, irrespective of the look-back period, T. In particular, when fixing $T = 40$ the best improvement can be noticed.

4.5.4 Comparison with State of the Art Trading Strategies

To finally assess the performances of the proposed StatArb approach in a real-world trading scenario, a comparison with a set of state-of-the-art portfolio strategies is done. The portfolio strategies are well-established in the literature [148]. Also, to be noted that for the sake of fair comparison the same validation process as in [114] was followed. Specifically, the considered list of baselines is shown below (for more in-depth details on these algorithms the reader is referred to [148]):

BHP - Buy&Hold-based Portfolio, *i.e.*, a portfolio implementation of the standard Buy&Hold strategy described in Section 4.4.4, where rather than buying a single asset (e.g., an *ETF* or a *stock*), the investor buys shares of all the index companies proportionally to their prices;

CRP - Constant Rebalanced Portfolio, *i.e.*, a variation of the BHP strategy, where the portfolio weights are periodically rebalanced, according to the price changes of the underlying assets;

UP - Universal Portfolio, which is a parameterized CRP strategy over the whole simplex domain. The algorithm learns adaptively from historical data and maximizes the log-optimal growth rate in the long run;

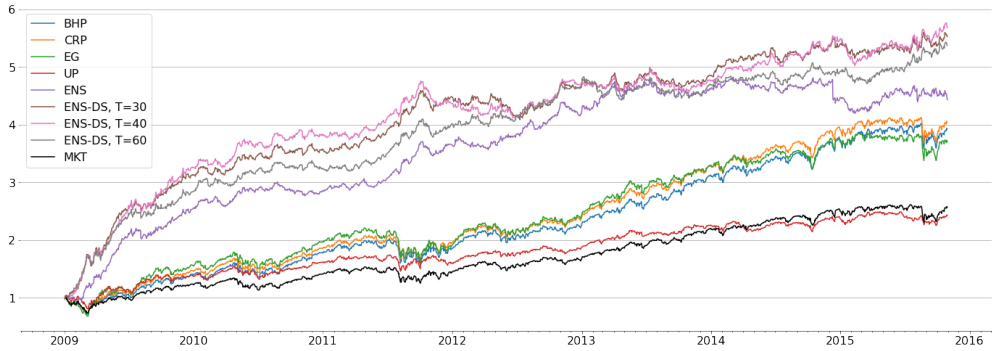
EG - Exponential Gradient, *i.e.*, a momentum strategy that focuses on the best performing asset of the index, in the last time period.

Table 4.7: Risk assessment of the trading strategies before transaction costs compared to the BHP, CRP, EG, UP, and MKT from a period from January 2009 to November 2015. Best values are highlighted in bold.

	BHP	CRP	EG	UP	ENS	ENS-DS $T = 30$	ENS-DS $T = 40$	ENS-DS $T = 60$	MKT
MDD[%]	28.6751	29.7377	29.5396	22.4178	13.6908	13.3987	**11.4293**	11.7018	27.131
Sharpe Ratio	1.0794	1.1037	1.0994	0.8644	1.6701	1.9902	**2.0045**	1.9218	0.8687
Sortino Ratio	1.5591	1.6085	1.5989	1.2387	2.5439	3.1163	**3.1176**	2.9901	1.2377

For this analysis, it has been taken into account a long period spanning between 2009 and 2015. The period encompasses both bull and bear markets

Figure 4.4: Equity curves for the proposed strategies compared to BHP, CRP, EG, UP, and MKT within a period from January 2009 to November 2015, before transaction costs.



regimes, *i.e.*, the market recovery post-global financial crisis (2009-2013), as well as low volatility periods (2013-2015), where statistical arbitrage strategies do not fare particularly well, as noticed by several works in the literature [149]. Figure 4.4 presents the cumulative returns (or *equity* curves) of the proposed methods (ENS and ENS-DS), against the aforementioned competitors and the behaviour of the general market (MKT). Table 4.7 presents the risk characteristics (in terms of Maximum Drawdown, Sharpe Ratio, and Sortino Ratio) for all these strategies. By analyzing Figure 4.4, one can notice that the proposed methods clearly outperform the baselines and the market, with our ENS-DS ($T = 40$) providing the best overall performance. Such a perception is then confirmed by looking at the results in Table 4.7, where the proposed approaches show significantly lower values of risk exposure (with a MDD ranging between 11.43% to 13.69%, against the 22.41% of the best alternative), and better return-over-risk metrics (e.g., by reaching a considerable Sharpe Ratio of 2.00).

Overall, our results also compare favorably to other statistical arbitrage strategies, such as classical pairs trading results in [53], where Sharpe ratio has a value of 0.59 for the top 20-pairs from 1962 to 2002. In [49], for a generalized pairs trading, the authors report a Sharpe ratio of 1.44 from 1997 to 2007. In [107], the author proposes a method that uses Elman neural networks and ELECTRE III with a Sharpe ratio of approximately 1.5 within a time-span from 1992 to 2006. Similarly, researchers from the Union Bank of Switzerland (UBS), in their work [150], leverage on RF models to form monthly portfolios. The trading signals are constructed based on the top quantile of the forecasted monthly returns, for which stocks are bought long, and the flop quantile for which the corresponding stocks are sold short. The authors test the methodology on constituents of Asia and Pacific indexes (including the Australia SP300 index), on a period from 1997 to 2015. Com-

paring the cumulative return in time (ours - ENS-DS, $T = 40$ versus theirs - MSCI AC Asia Pacific ex-Japan), one can observe a similar behavior during the financial crisis of 2007-2009, where both approaches recorded their best performance. Afterward, they report a declining performance that reaches a plateau in 2011-2015. As per Figure 4.4, our proposed method developed its highest earnings in early 2009, a period of high turmoil. After that period, though the method registers moderate returns, it exhibits an increasing trend. Finally, in terms of risk, they report a MDD slightly lower (approximately 10% compared to our MDD values of 11% before transaction costs and 26% after transaction costs) and quite surprisingly registered after the global financial crisis, in 2012.

4.6 Conclusions

This chapter proposes a general approach for risk-controlled trading algorithmic trading based on machine learning and statistical arbitrage. In this regard, the well-established steps of a statistical arbitrage trading system are followed. That is forecasting the price returns, then ranking followed by trading of a number of pairs. The forecasting is performed by a heterogeneous ensemble and, subsequently, to mitigate the risk, a dynamic asset selection strategy was proposed. Such a strategy prunes assets if they had a decreasing performance in the past period. The proposed approach can be regarded as general and applicable to a wide variety of assets, as well as all of its underlying components.

To test the hypothesis, the proposed approach was instantiated, and as such the study focuses on the S&P500 Index, using its constituents as tradeable assets and statistical arbitrage as a trading strategy. For forecasting, the approach uses three machine learning algorithms, that is Light Gradient Boosting, Random Forests, and Support Vector Machines to which it was appended a statistical tool, Auto-regressive moving average, widely used for time-series forecasting. Subsequently, their output is ensembled. To create a heterogeneous ensemble, the proposed approach uses a set of heterogeneous features as input for the models. By performing a walk-forward procedure, for each stock and walk, all the combinations of features and internal parameters of each regressor are tested, and finally the best model out of each of the possible combinations is selected. Based on this diverse pool of forecasting methods, the experiments show that the ensemble strategy reaches significant returns of 0.113% per day or 31% per year. When enhancing the trading process with the dynamical asset selection strategy, the returns show an increase from 31% of the common ensemble to 33%. As many strategies can provide very high returns, they often are also very volatile. The relation of profit and the amount of risk to achieve this result have to be in a reasonable propor-

tion. Under this consideration, the strategies are assessed using various risk metrics. To this end, the most remarkable result is that the risk hedged to 14% maximum draw-down for the ensemble to 11% for the proposed dynamic asset selection method, compared to 55% of the market.

Chapter 5

Algorithmic Trading Powered by eXplainable AI

"On two occasions I have been asked, 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question."

Charles Babbage

Outline

In this chapter it is proposed and investigated the use of machine learning explainability techniques in a financial context. Section 5.1 formulates the problem, Section 5.2 describes in depth the XAI techniques specifically tailored for financial forecasting. Section 5.3 presents a first series of experiments and discusses the improvements brought by the XAI techniques to the forecasting performances, while Section 5.4 details the use of the proposed XAI methods in a financial trading framework. Section 5.5 presents some concluding remarks.

Despite the increased adoption of ML models in several domains including the financial one, relying only on sophisticated ML models introduces the risk of creating and using decision systems that work as "black boxes" that nobody can truly understand. This fact has a direct impact on ethics, accountability, safety, and industrial liability [151, 152]. Therefore, it is essential to have frameworks that understand how ML methods produce their findings and suggestions. As a consequence, the field of XAI has emerged, that is, techniques developed to "X-ray the black-box" and provide insights about *which input features* are more important and how they affect the predictions [42].

Feature selection is the process of identifying a representative subset of features from a larger cohort. In the context of ML, the advantages gained from feature selection are numerous. For example, finding the most descriptive features reduces model complexity and accelerates computation, *i.e.*, model training and prediction. Additionally, it improves model interpretation and diagnosis [153], as understanding what the representative features mean leads to gain a deeper insight of the problem at hand. Besides the advantages enumerated above, the most important one lies in improving prediction performances as, when performing the forecasting, it is well-known that using too many features can lead to over-fitting, which can significantly hinder the performance of the predictive models [16].

One can either choose to manually select the features or to apply an automated method. The challenge in the manual selection of the features is that this process requires expert knowledge about the data at hand. Besides, due to unclear dependencies within financial data, identifying significant information from irrelevant information is a challenging task that can be better tackled in an automated fashion. This constitutes the goal of this chapter.

Specifically, in this chapter it is investigated the ability to select relevant features in an applied setting. In order to do so, for a large set of stocks, a RF model is trained using lagged returns as features, and the aim at forecasting the next-day return. To be noted that the problem is formulated so that a model is built for each of the stocks. That is because in many cases stocks behavior is highly heterogeneous, and the relevant features may differ between stocks, across market regimes, or geographies [16]. Naturally, by performing feature selection at a stock level, the features that are the most relevant to each stock can be selected. Conversely, globally selecting the features may perform well, on average, across all stocks, but choosing the input features at an individualized level outperforms the former.

Secondly, the set aim of this chapter is combining these three dimensions, *i.e.*, machine learning, explainable AI, and trading/statistical arbitrage. Although these points are discussed in the literature on an individual basis, there is a lack of empirical work dealing with the use of XAI techniques in a StatArb trading context.

5.1 Problem Statement

Considering that each stock has a corresponding dataset $\mathcal{D} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$ and the input features given by the matrix $\boldsymbol{X}$ and the target by $\boldsymbol{Y}$, our forecasting Task (as described in Section 2.2) is to predict the target $\boldsymbol{Y}$ based on the input features $\boldsymbol{X}$ using a model function $f(\cdot)$. While doing so, the particular interest lies on identifying the parts of the input X that are the most relevant for f to produce the prediction Y. More specifically, the proposed challenge consists

of selecting *automatically* from the input features set the features which are relevant and discarding the noisy ones and, as a consequence, increasing the predictive power of our model f with respect to some performance measure. A feature X_j is a noisy feature if X_j and $\boldsymbol{Y}$ are independent and a relevant feature, otherwise. Note that this definition of noisy features has also been used on multiple occasions by several authors [45, 154]. Moreover, for a given set of stocks the proposed approach aims at identifying which of the features are considered irrelevant across the whole set of stocks and discard them so that the overall performance is increased.

5.2 Methodology

5.2.1 Overview

At a high level, first a feature vector is constructed. The vector contains daily financial information that is fed into a ML model (base regressor), used in turn to obtain the predictions for the stock returns. Next, feature importance scores are computed and, based on them, features are divided into two categories: important features and unimportant ones. The unimportant features are then discarded, and a new model is trained. The new model receives as input only the features that are considered to be important and makes consequent predictions. The loss of the newly trained model is subtracted from the loss of the base regressor. The loss difference is used to compute an optimal feature score threshold, below which features are indeed set as unimportant – our intuition is that, if removed, these features do not worsen the overall prediction performance. The described process constitutes the learning phase. In the trading phase for each stock, only the model trained with the optimal features set is used.

5.2.2 Feature selection

With respect to Figure 5.1, the second block of the forecasting phase aims to assign feature importance scores with the goal of identifying uninformative features for the prediction task, and proposes them for removal. In the following paragraphs, it is detailed the methodology for assigning the feature importance scores and the subsequent feature removal strategies. Note that the feature selection importance score computation, the optimal threshold computation, and the feature selection process, are computed using the validation dataset only, thus ensuring the absence of data-leakage.

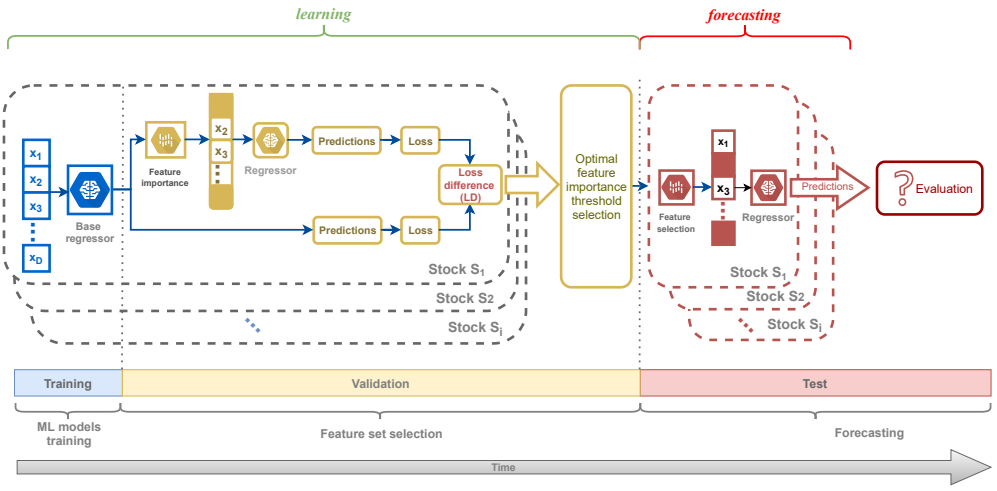


Figure 5.1: Block diagram of the XAI powered trading system

Feature Importance Score with Permutation Importance (PI)

The motivation in choosing the PI approach is threefold: (i) it is model agnostic, i.e., applicable to any model derived from any learning method; (ii) it computes the feature importance on the test set, i.e., out-of-bag samples (OOS), which makes it possible to highlight which feature contributes the most to the generalization power of the inspected model [155]; (iii) it has no tuning parameters, and it relies only on averages which makes it statistically very stable.

Regarding the computation of the feature importance and of Equation 2.3, there are variations of the equation that scale better the changes in loss due to permuting feature j values with the loss of the base regressor, such as the following:

$$VI_j = \frac{1}{N} \sum_{\pi} \frac{L(Y, f(X^{\pi,j})) - L(Y, f(X))}{\mathbf{L(Y,\ f(X))}}. \tag{5.1}$$

The feature importance expressed as above can be interpreted as the average change in prediction loss relative to the loss of the base regressor.

In any of the variants (Equation 2.3 or Equation 5.1), the feature importance score will have either (i) positive, (ii) close to zero, or (iii) negative values. As a consequence,the features can be classified as follows. *Important features* will have *positive* scores, as replacing the corresponding features values with other random values, increases the loss with respect to the original regressor. On the other hand, *unimportant features* will have *negative* feature importance scores, as their replacement with noise-like information improves the prediction performance, *i.e.*, the loss decreases with respect to the original predictions. In the middle, lie the *unimportant features*, *i.e.*, features whose feature importance score is close to 0, thereby they do not significantly

contribute to the prediction outcome. As a consequence, replacing them with noise does not introduce a substantial loss change. Out of these three categories of features, the best suited candidates for removal are the last two categories, as long as their feature importance is lower than 0.

Feature Removal Strategies

Given the possible values of the feature importance score and the implication on feature categorization, in this dissertation three strategies to remove the features are being proposed:

1. *PI best* - identifies the features which have feature importance lower than 0 and, at the same time, have the *highest* feature importance among them. We remove those features if their importance is lower than a certain threshold.

2. *PI worst* - identifies the features whose importance is lower than 0, and, simultaneously have the *lowest* feature importance among them. Similarly to *PI best*, if those features have their importance score below a certain threshold, we remove them.

3. *PI running* - identifies the features with the importance lower than a variable threshold, and proposes for removal the one(s) with the *highest* feature importance score.

The main difference between *PI running* and the other strategies is that *PI running* uses the threshold to identify the best features to remove. *PI best* and *PI worst* use, instead, the threshold to identify whether the best features (*i.e.*, the features with the highest importance below 0) or the worst feature (*i.e.*, the features with the lowest importance below 0) will be removed.

To better grasp the differences between the three proposed strategies, Figure 5.2 illustrates a simplified example. The figure presents information for three stocks: Alphabet, Inc. (GOOGL), International Business Machines (IBM), and Intel Corporation (INTC). The example uses as features lagged returns (LR) whose indices correspond to the window lag when computing the returns (see Section 2.1 for further details on features). The left-hand side panel, *i.e.*, (a) Feature importance, shows the stocks, the features, and their corresponding feature importance score computed by Equation 5.1. For each stock, the features are sorted in descending order by their feature importance. For sake of simplicity, for stocks *GOOGL* and *INTC* only the features with negative feature importance are shown. For completeness of the example, for the *IBM* stock, the least important feature is shown. Such a scenario can happen if the stock has no features whose importance is lower than 0, therefore all features contribute to the prediction outcome. The right-hand

Stock	Feature	Feature importance
GOOGL	LR_2	-0.1077
GOOGL	LR_{63}	-0.3906
GOOGL	LR_5	-0.5298
GOOGL	LR_3	-1.1571
IBM	LR_1	0.1060
INTC	LR_5	-0.2911
INTC	LR_{252}	-0.3211
INTC	LR_3	-0.3420
INTC	LR_{63}	-0.7658
INTC	LR_{126}	-1.1547

(a) Feature importance

	PI best		PI worst		PI running	
Stock	Feature	Feature importance	Feature	Feature importance	Feature	Feature importance
GOOGL	N/A		LR_3	-1.1571	LR_{63}	-0.3906
IBM	N/A		N/A		N/A	
INTC	LR_5	-0.2911	LR_{126}	-1.1547	LR_5	-0.2911

(b) Selected features

Figure 5.2: Illustration of the proposed feature selection strategies for three stocks: *GOOGL*, *IBM*, and *INTC*. Panel (a) shows, for each stock, the features and their importance. To note that, in this example for simplicity purposes only a limited number of features is presented. In a working scenario, the feature importance panel would include all the features associated to a stock. Moreover, the example illustrates the removal of only one feature, while the strategies allow for multiple features dropping. Panel (b) shows the selected features for each method for an assumed threshold of -0.15.

side panel, *i.e.*, (b) Selected features, shows the effect of the three feature selection strategies that use a threshold equal to -0.15.

Starting with *PI best*, for the *GOOGL* stock, it does not select for removal any feature, as the feature with the highest negative feature importance is LR_2. However, its feature importance is not below the threshold -0.15. For the *IBM* stock, for obvious reasons no features are removed. In the case of the *INTC* stock, the feature with the highest negative feature importance, and that also falls below the threshold -0.15, is LR_5; thus, *PI best* will propose it for removal.

Switching to *PI worst*, in the case of *GOOGL*, the strategy will remove LR_3, since it has the lowest value and is also below the threshold. For the *INTC* stock, the strategy will remove LR_{126} for the same reasons.

Finally, for the *PI running* strategy, with respect to the *GOOGL* stock, the strategy will remove LR_{63}, since it is the first feature whose importance value is below the threshold. For *INTC*, the strategy proposes to remove LR_5, the same feature chosen using the *PI best* strategy.

Optimal Feature Importance Threshold Selection

For each of the proposed feature selection strategies, *i.e.*, *PI best*, *PI worst*, and *PI running*, an optimal feature importance threshold is computed. For each feature selection method, model and stock, the unimportant features according to the identified feature importance threshold are removed and a new model is trained. Then, for each threshold value, the mean loss difference between the base regressor (without any feature removed) and the corresponding newly trained model is computed.

Under this premise, for each feature selection method, the optimal threshold is computed. That is, the threshold that maximizes the mean loss difference above. The introduction of the optimal threshold enables the proposed approach to *learn* in cross-section the features that can be removed in order to increase the overall prediction performance, *i.e.*, also, to decrease the mean loss (across all the stocks). In other words, in the case of *PI best* and *PI worst*, the optimal threshold controls the sparsity of stocks and models that have features removed, whereas in the case of *PI running*, it controls the highest score below which the features can be considered unimportant, thus suitable candidates for removal. Note that, as highlighted in Figure 5.1, for the computation of the optimal feature threshold only the validation dataset is used in order to avoid any look-ahead bias.

Implementation details

In terms of *Big O* notation, our algorithm entails an $O(|S| \times m^{-})$ complexity proportional to the number of stocks $|S|$ and m^{-} the number of features to remove for each stock or group of stocks.

For data collection, in this study the publicly available data from Yahoo Finance is used https://finance.yahoo.com/. In the following Sections we provide two applications of the features selection strategies introduced thus far. Section 5.3 presents and discusses the performance of the feature selection strategies from a ML learning perspective, that is how the strategies fare when the the goal is to increase the predictive performance over the entire stock set. Section 5.4 presents the performance of the feature selection strategies from a financial perspective.

5.3 Explainable AI for Financial Forecasting

5.3.1 Experimental Setup

This section highlights the premises under which the experiments are carried out, *i.e.*, a feature selection approach applied to a financial dataset with the goal is to improve the prediction performance by discarding features for each

stock according to the methodology explained in Section 5.2 and use the remaining input features to predict the next day's returns.

Data and Input Features

The experiments are carried out using a set of 300 stocks given by the constituents of the S&P100 Index[1], CAC40[2], FTSE 100[3], S&P Asia 50 Index [4], and Dow Jones Global Titans 50.The choice is justified by the fact that the aim is to construct a highly heterogeneous stock set spread out among different continents, *i.e.*, Europe, the U.S., and Asia-Pacific.

For each stock, daily raw financial data is collected such as opening (*open*) and closing (*close*) prices. Based on these data, the following information are constructed:

Features - Lagged returns (computed with respect to day d), each expressed as

$$Return\_j = \frac{close_{d-1} - open_{d-j}}{open_{d-j}} \text{ for } j \in \{1, 2, 3, 4, 5, 21, 63, 126, 252\},$$

where j and denotes the length of the time-window for which the return was computed[5]. $Return\_j$ generates 9 features.

Target - Intra-day return

$$y_i = \frac{close_i - open_i}{open_i}.$$

Forecasting and Feature Selection

As prediction models, RF are used for two compelling reasons. First, it is a ML model that delivers competitive results within financial prediction tasks [111, 16]. Second, it is an explainable model [41]. Hence, For each stock, an individual model is trained using the following hyperparameters: *n_estimators=500* - the number of decision trees, *min_samples_leaf=5* - the depth of the trees is limited, and finally *max_features=1* - the maximum number of features in each split is limited.

Finally, for determining the feature importance, the feature values have been permuted $N = 100$ times (in Eq. 5.1). To determine the optimal feature importance threshold, for each feature removal strategy, the threshold values

[1] `https://en.wikipedia.org/wiki/S%26P_100`

[2] `https://en.wikipedia.org/wiki/CAC_40`

[3] `https://en.wikipedia.org/wiki/FTSE_100_Index`

[4] `https://en.wikipedia.org/wiki/S%26P_Asia_50`

[5] For example, $j = 1$ denotes the return in the previous day for each observation, whereas $j = 252$ denotes the return in the past year, considering that there are 252 trading days in a year.

takes values within the range of 0 to -0.025 with a step of 0.0001, as presented in Section 5.2.

Backtesting

To validate our assumption that the influential features may differ across different market regimes, *i.e.*, in time, the proposed strategies are backtested considering a study period of January 2007 to January 2018. As introduced in Section 2.5, the *walk-forward* validation has been employed and each walk consists of four years of training and a year of testing, that is seven walks.

5.3.2 Baselines

The proposed strategies are compared considering three baselines:

1. **Base regressor** - for each stock, we train a model including all the features in the feature set (we do not perform any removal);

2. **MDI** - for each stock, we compute the MDI as presented in Section 2.3 using scikit-learn implementation[6]. Specifically, MDI is the average feature importance of all decision trees (in our case 500) composing the forest. Moreover, using the RF hyperparameter *max_features=1* we overcome RF's inability to correctly estimate feature importance when dealing with features that are correlated with each other [156], as it often happens for financial time series. Then, given the obtained features' importance, we discard the feature with the lowest importance.

3. **LIME** - for each stock, given a trained base regressor, we explain each sample in the test period with a linear model, as detailed in Section 2.3. We proceed as such, as LIME is not designed to assign feature importance globally.Then, each observation's feature importance is equal to the absolute value of the regression coefficient of that feature. For each sample, we order the input features by their importance and rank them (where rank 1 denotes the highest feature importance). To obtain the global feature importance, *i.e.*, the feature importance for the whole test period, we average the ranks across all test observations and remove the least important feature, that is, the feature with the lowest average rank. For the LIME baseline, we use the python LIME package[7].

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor.feature_importances_`

[7] `https://github.com/marcotcr/lime/tree/master/lime`

Note that for the proposed strategies and MDI and LIME baselines, for a fair comparison, we perform the same procedure: we compute the feature importance for each model (*i.e.*, one per stock) and each input feature; then, we remove - from the feature set - the features selected by the strategies, or the least important one as indicated by MDI or LIME. Finally, we train a new model and evaluate the performance.

5.3.3 Results and Discussion

This section presents an assessment of the proposed strategies for each of the walks following testing methodology introduced in Section 5.3.1. Two different experiments are performed: (A) finding the optimal feature importance threshold in order to select features to be discarded, and (B) qualitatively evaluating the improvement over the baselines.

To assess the performances of the proposed feature selection strategies three performance metrics are considered. Point forecasting performance is determined by using the mean squared error (MSE), similar to [121]. Evaluating the performance of feature selection methods on real data is difficult since ground truth relevance is not known. Therefore, the focus is on the prediction performance. Specifically, the effectiveness of removing one feature is measured by assessing the number of improvements over the base regressor in terms of MSE and the ratio given by the number of improvements over the base regressor out of the number of models (stocks) which had a feature removed. Explicit expressions for the metrics are provided as follows:

- $MSE = \frac{1}{T}\sum_i (y_i - \hat{y}_i)^2$, where $\hat{y}_i$ represents the i^{th} predicted value, y_i the i^{th} target value, and T the number of observations in the test period;

- $no\ improvements = \sum_{i=1}^{|S|} I(MSE_{base\ regressor_i} - MSE_{M_i^-} > 0)$, where $i \in \{1, ..., |S|\}$, S is the set of stocks, $MSE_{base\ regressor_i}$ is the error of the base regressor of stock i (*i.e.*, no features have been removed), $MSE_{M_i^-}$ is the error of the regressor for which a feature has been removed, and finally, I is the indicator function. In other words, *no improvements* measures how many models have an MSE improvement (lower than the MSE of the base regressor) when removing a feature;

- $ratio = \frac{no\ improvements}{\mathcal{M}^-}$, where $\mathcal{M}^-$ is the number of models that had a feature removed, or, differently put, the number of models that have one feature whose importance is lower than a given threshold.

(a) (b) (c)

Figure 5.3: (a) Average MSE difference between the base regressor and the model when discarding unimportant features according to each method for different threshold values. To note that positive values indicate a better performance of our strategies. (b) The number of models having an MSE improvement when discarding unimportant features according to each method for different threshold values. (c) Percentage of models having an MSE improvement when discarding unimportant features according to each method for different threshold values.

Experiment A.

Figure 5.3 shows the performance of the proposed strategies under the three evaluation metrics previously introduced, for different thresholds and against the base regressor. Figure 5.3a shows the average MSE difference between the base regressor and the model with features removed (the feature whose importance is lower than the threshold). Figure 5.3b, analogously, shows the number of improved models by comparison to the corresponding base regressor. Finally, Figure 5.3c presents the ratio of improved models (*i.e.*, number of times the MSE decreases) out of the total number of models that had a feature removed. Each figure shows metrics values for thresholds ranging from 0 to -0.025 with a step of 0.0001. Furthermore, for a subset of threshold values $\{0, -0.005, -0.01, -0.015, -0.02, -0.025\}$ in each the graph there are shown the corresponding metric values,*i.e.*, *average MSE difference*, *no improvements*, and *ratio* obtained by each of the proposed methods.

Table 5.1 presents the optimal threshold per walk for each of the proposed methods and reports the evaluation metrics. With regards to finding the optimal threshold, it can identified as the one that maximizes the average MSE difference between the base regressor and the corresponding models trained with a feature removed, provided that the feature has the feature importance lower than the threshold. Average MSE differences with values higher than 0 are those showing a clear improvement of the proposed strategies.

Figure 5.3a shows the variation of the average MSE difference between the base regressor and the model trained without the feature identified as irrelevant by the proposed strategies. In this context, naturally, for the *average MSE difference*, values higher than 0 are sought, as they clearly indicate a better prediction performance of our strategies by comparison to the base regressor. Values close to 0 indicate that removing features does not bring any improvement to the prediction performance. However, having the same performance with a lower number of features is better for the computational time and simplicity of the feature engineering process. Conversely, values lower than 0 indicate a prediction performance degradation and that most of the features removed were, in fact, informative. In Figure 5.3a the "0" value is marked with a black, continuous line. In Figure 5.3a, the reader can notice that the *average MSE difference* shows different values across the presented walks. This fact may be due to data-shift that makes the identification of non-representative features difficult. It negatively stands out the walk between 2015-2016 when *PI best* has the lowest performance with values asymptotically close to 0 or below. The results show that *PI best* systematically misclassifies important features as unimportant. Similarly, for high threshold values, *PI running* suggests removing features that are important and, as such, worsens the prediction performance of the regressors. For the other walks, all the methods demonstrate their capability in identifying unim-

portant features, thus providing a prediction performance improvement. As a final remark, *PI worst* is the most stable strategy, and it outperforms the base regressor in all cases.

By analyzing the Figure 5.3b which shows the number of models that have an increase in predictive power (*i.e.*, the feature removal was effective), the reader can notice that the best performing method is *PI worst*, followed by *PI running* and *PI best*. The latter produces satisfactory results only for high threshold values. One can expect such behavior since *PI best* removes the uninformative features (with importance lower than 0), but the input features with the highest feature importance among them.

Figure 5.3c shows the ratio of the number of models whose predictive performance has increased (with the feature removal) by the total number of models that had a feature removed. One can see a similar behavior of *PI worst* and *PI running* for all the walks, but *PI worst* shows a higher ratio. Desirable values are close to 1. The *PI best* method is at the opposite pole by displaying discordant values: (i) below 0.5, meaning that less than half of the models have a significant improvement, (ii) asymptotically close to 1 which means that all feature removals resulted in a loss decrease. Although this might seem an optimal result, in Figure 5.3b one can notice a low number of feature removal actions. Therefore, the *ratio*, in this case, does not provide informative insights.

Table 5.1 shows the optimal thresholds for each of the strategies together with the corresponding evaluation metrics. Judging by the *average MSE difference*, *PI running* obtains the best results overall. Furthermore, the *no improvements* indicates that *PI worst* is the best performing method. Thus, when removing the feature with the lowest feature importance the highest number of improvements are obtained. *PI best* is the worst performing of the three strategies, accounting for a negative *average MSE difference* for the walk 2015-2016. Moreover, for the same walk, the *ratio* of improvements over the total number of feature removals performed is 0.5. Such a value means that half of the feature removals resulting in lower performance than the base regressor are worst than the other half, which has better performance. The different values of the optimal threshold across different walks confirm the prediction models' time-dependency.

Experiment B.

Figure 5.4 presents the average MSE across each walk for the models when removing the features according to the proposed strategies. Note, that if for *PI best* and *PI worst* no feature was removed the MSE reverts to the value of the base regressor. The obtained results are compared against the three presented baselines in Section 5.3.2.

Results presented in Figure 5.4 highlight that the proposed strategies out-

Table 5.1: Threshold across walks and the corresponding average MSE difference between the base regressor (all features are used) and the models when a feature was removed according to the proposed methods.

	PI best				*PI worst*				*PI running*			
walk	thresh.	average MSE difference	no improvements	ratio	thresh.	average MSE difference	no improvements	ratio	thresh.	average MSE difference	no improvements	ratio
2011-2012	-0.0142	5.05e-05	4.0	1.0000	-0.0198	1.3e-05	101.0	0.9018	-0.0249	2.39e-05	26.0	0.8965
2012-2013	-0.0076	2.65e-06	6.0	0.8571	-0.0079	3.0e-06	158.0	0.8316	-0.0079	0.7e-06	135.0	0.7219
2013-2014	-0.0084	5.04e-06	1.0	1.0000	-0.0188	0.5e-05	144.0	0.8521	-0.0249	2.70e-06	9.0	0.7500
2014-2015	-0.0095	9.58e-07	1.0	1.0000	-0.0001	0.9e-05	129.0	0.8600	-0.0003	7.04e-06	110.0	0.4247
2015-2016	-0.0027	-1.76e-07	30.0	0.5000	-0.0198	27.8e-05	197.0	0.7787	-0.0246	1.06e-03	9.0	0.6923
2016-2017	-0.0083	1.02e-05	2.0	1.0000	-0.0199	1.90e-05	151.0	0.8629	-0.0249	4.89e-05	7.0	0.7000
2017-2018	-0.0079	1.53e-05	1.0	1.0000	-0.0195	4.7e-05	169.0	0.8009	-0.0214	4.17e-05	28.0	0.8235

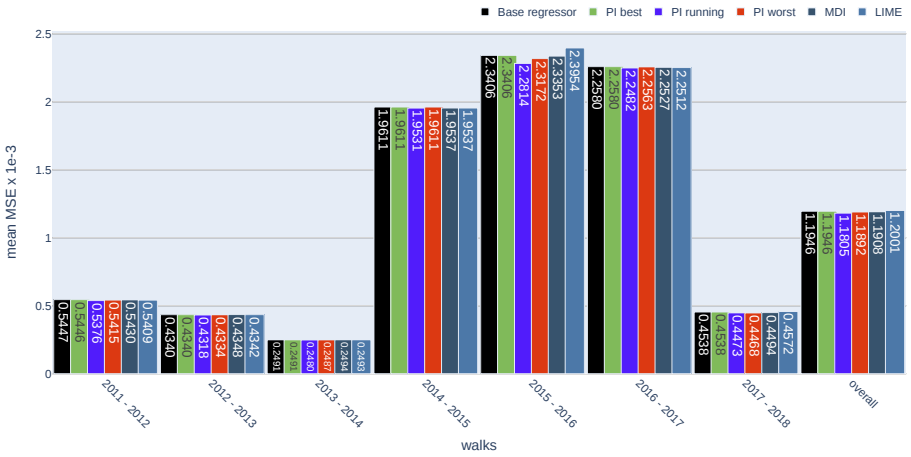


Figure 5.4: Average MSE across different walks for the proposed methods when, for each stock, the features are removed if the feature importance is lower than the optimal threshold. Comparison to the three baselines: all feature are used (base regressor), remove the feature with lowest feature importance as given by MDI (MDI), and remove the feature with lowest feature importance as given by LIME (LIME).

perform the three baselines. Specifically, *PI best* achieves similar average MSE values of the base regressor on most of the walks and overall. Such a result is encouraging as, although the prediction performance does not increase much, employing a smaller number of features leads to a lower training and prediction time.

Overall, *PI worst* has the second-best behavior, while *PI running* achieves

the best results out of the three proposed methods. Such a behavior meets our expectations, as the MDI is known to be biased, especially when it comes to time-series data-sets [45]. Consequently, MDI may over- or under-estimate feature importance, which might be a possible explanation for the close average MSE to the base regressors' (using all the features). As for LIME, the reader can notice that it has the worst performance amongst all methods. Therefore, a conclusion can be drawn, *i.e.*, LIME consistently underestimates feature importance and mistakenly assigns lower feature importance to otherwise important input features. A possible explanation of this behavior lies in the fact that LIME relies on perturbing the input observation around its neighborhood and observes the model's behaviors (predictions). Originally, LIME was designed with specific implementations for text and images but without focus on time-series data. Moreover, financial time series have a low signal-to-noise ratio and exhibit a random-walk behavior with a low degree of predictability, and opening the black-box models by "peaking" into the randomly generated neighborhood yields under-par results.

Also, to be noted that in the case of financial time-series forecasting, in the particular case of predicting the daily returns, both the input features and the target variable values are small with an order of magnitude of 1e-4. Consequently, the MSE has small values too, which constitutes a challenge for model training or when it comes to distinguishing a performing model from an under-performing one.

Statistical Significance.

To test the statistical relevance of our results, the Wilcoxon signed-rank test has been used, which is non-parametric test that determines whether two samples were selected from populations having the same distribution. Such a test is an alternative to the paired Student's t-test, when the distribution of the difference between two samples' means cannot be assumed to be normally distributed, as in our case. The null hypothesis $\mathcal{H}_0$ is asserting that the MSE of each feature selection method,*i.e.*, *PI best*, *PI worst*, and *PI running* has the same distribution to the MSE of the base regressors. The p-values obtained are: 0.0678, $7.3761e-21$, and 0.0179, respectively. In all tests, the confidence level is set to 95%, that is, if p-value is below 0.05, then $\mathcal{H}_0$ can be rejected. Under this assumption, $\mathcal{H}_0$ can be rejected for *PI worst* and *PI running*, but not for *PI best*. However, the latter outcome is expected and due to low number of feature removals, as shown in Table 5.1.

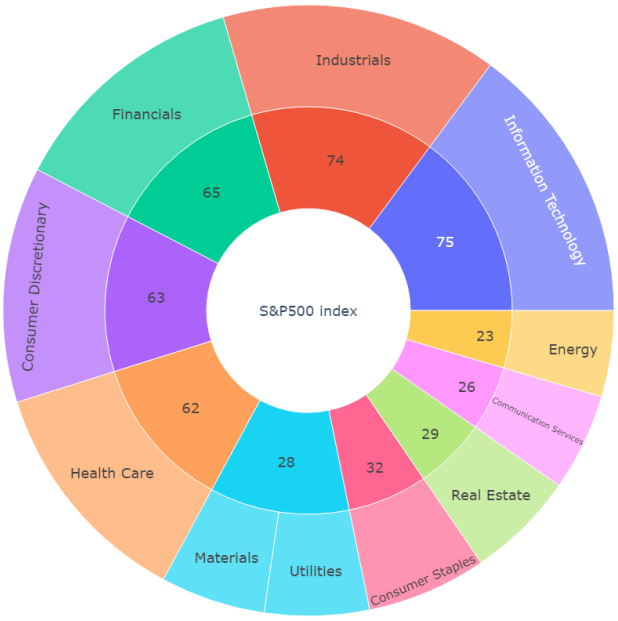


Figure 5.5: Number of constituents of the S&P500 index into sectors.

5.4 Statistical Arbitrage powered by XAI

5.4.1 Experimental setup

In this section, we describe the technical setup of the experiments that we carried out. We follow along with the main components of our trading strategy: data, model training, feature selection, ranking process, and trading execution.

Data and Input Features

We consider stocks composing the S&P500 index as a reference dataset. The S&P500 constituents represent the leading 500 companies in the U.S. stock market and, at the same time, a large-capitalization segment. These characteristics make these companies highly attractive to investors and, as a consequence, they are very challenging for any trading strategy, as various works in the literature have pointed out [112, 111].

For each stock, we construct our dataset $\mathcal{D}$ by collecting daily raw financial information such as *Open Price*, *High Price* in the day, *Close Price*, *Low Price* in the day, and *Volume* of stocks traded during the day (i.e., OHCLV variables). Based on this information, we create two different types of features:

Lagged returns (LR) - computed with respect to day t and expressed

as

$$LR_{t-j} = \frac{close_{t-1} - open_{t-j}}{open_{t-j}},$$

where $j \in \{1, 2, 3, 4, 5, 21, 63, 126, 252\}$ and denotes the length of the time-window for which the return is computed[8]. The feature vector $\mathbf{x_t}$ is therefore composed of 9 lagged returns for each day t. Similar types of input are used in previous works such as [157, 113, 3] and the lagged returns are meant to provide the ML algorithms with information within three different time horizons: (i) the recent past, i.e., $j \in \{1, 2, 3, 4, 5\}$, (ii) medium past of the last trading month, last three trading months, respectively, for $j \in \{21, 63\}$, and finally (iii) returns corresponding to the last six months and one year (i.e., for $j \in \{126, 252\}$).

Technical Indicators (TI) - represent statistical tools that investors extensively use to make their investment decisions. They are extensively used within the academic literature to either generate trading signals [119] or as input features for ML models [129, 77, 81]. To construct the feature vector $\mathbf{x_t}$, similarly to [1, 2], we use the same set of nine technical indicators: Exponential Moving Average (EMA(10)), Williams %R, Stochastic Oscillator (%K), Relative Strength Index (RSI), Rate of change (ROC), Accumulation/distribution indicator (ADO), Moving Average Convergence Divergence (MACD), and two disparity measuring indicators, respectively on the last 5 days and on the last 10 days (D(5), D(10)). To compute the feature vector of technical indicators for a day t, we use financial information over the past days to t (each indicator needs a different number of previous days to t, according to their formula).

Target variable is a continuous variable given by the intra-day price return defined as:

$$y_t = \frac{close_t - open_t}{open_t}.$$

The features are rescaled within the range $[-1, 1]$ by applying a MinMaxScaler [142] fit on the training set[9]. The MinMaxScaler scales and translates each feature individually such that it is in the given range on the training set, *i.e.*, $[-1, 1]$. The rescaling is a necessary step as it increases the convergence likelihood of the SVR model, as well as un-biased learning in the case of decision trees based models such as RF and LGB.

Similarly to [2], we build ML models containing data either for a single or multiple stocks belonging to the same sector in the S&P500 index. Herein we refer to this particular setup as the data level, also referred to as stock level,

[8]For example, $j = 1$ denotes the return in the previous day for each observation. At the opposite pole $j = 252$ denotes the return in the past year, provided that there are 252 trading days in a year.

[9]The transformation is learnt on the training set only in order to ensure that no data-leakage happens.

or sector level. In Figure 5.5 we show the distribution of the S&P500 index's stocks into their corresponding sectors.

Model training

As mentioned throughout the paper and reflected in Figure 5.1, the ML models are trained using a 3 years time-span for each rolling window. The hyperparameters of the models are listed in Table 5.2.

Table 5.2: Hyperparameters of the ML models.

Level	Feature Type	ML Model Type	Hyperparameters
Stock	LR	LGB	n_estimators=220, num_leaves=252, max_depth=7, learning_rate=4e-2, reg_alpha=9e-4, bagging_fraction=0.65
		RF	n_estimators=500, min_samples_leaf=5, max_features=1
		SVR	C=10, tol=1e-5, epsilon=1e-5, gamma=9e-3
	TI	LGB	n_estimators=100, num_leaves=21, max_depth=7, learning_rate=5e-2, colsample_bytree=9e-4, bagging_fraction=0.65
		RF	n_estimators=500, min_samples_leaf=5, max_features=1
		SVR	C=10, tol=1e-5, epsilon=1e-5, gamma=9e-3
Sector	LR	LGB	n_estimators=250, num_leaves=230, max_depth=12, colsample_bytree=.9,learning_rate=5e-3,
		RF	n_estimators=300, max_samples=0.5, max_features=1
		SVR	C=2, tol=1e-5, epsilon=1e-5, gamma='scale'
	TI	LGB	n_estimators=100, num_leaves=100, colsample_bytree=0.8, max_depth=15, learning_rate=0.03, subsample=0.8
		RF	n_estimators=300, max_depth=40, max_features=1
		SVR	C=2, tol=1e-5, epsilon=0.25e-2, gamma='scale'

LGB, as previously stated, is prone to over-fitting due to its manner of constructing the decision trees. To control this behavior, we defined the maximum depth levels of the tree, *max_depth*. Also, considering that LGB constructs the decisions trees sequentially, a large number implies high computation costs and also the possibility of over-fitting, therefore we chose for *n_estimators* conservative values lower than 250. Depending on the type of features and on the level of the model, we also considered different *num_leaves* values, aiming at achieving a balance between a conservative model and a good generalization capability. The percentage of features considered when doing a split is restricted by *colsample_by_tree*, which can be thought of as a regularization parameter. The work in [36] suggests in general to set a learning rate lower than 0.1; we therefore considered values lower than 0.01 in order to account for a better generalization over the dataset.

RF models are normally not subject to over-fitting when a large number of estimators (decision trees) is used. Therefore, we set *n_estimators* to high values up to 500. We limit the maximum number of features in each split by imposing *max_features* equal to 1, and limit at the same time the depth of the tree by setting either the *min_samples_leaf* or *max_depth* parameters.

SVR solves an optimization problem that involves two parameters: the regularization parameter, C, and the error sensitivity parameter, ϵ. The C parameter controls the trade-off between model complexity and number of non-separable samples. A lower C generally encourages a larger margin, whereas higher C values lead to a harder margin [38]. Instead, the ϵ parameter controls the width of the ϵ-insensitive zone and is used to fit the training data. An excessively high value leads to flat estimations, whereas a too small value is not appropriate for large or noisy datasets. The work in [143] suggests that the γ value of the kernel function should vary together with C, and high values of C normally require high values of γ too. As this is highly dependent on the data variance, we set it such that to be scaled accordingly with the data.

Feature selection

Next to model training, we compute, for each model and each feature, the feature importance using *PI* over $N = 100$ repeated permutations. We have chosen this value experimentally in order to achieve a good trade-off between computational running time and reliable estimations of the feature importance. Then, for each of the proposed feature selection strategies, *i.e.*, *PI best*, *PI worst*, and *PI running*, we perform a preliminary feature removal process by selecting the threshold values in the range $[0, -0.025]$ with a step of 0.0001. For each model we remove the indicated features $K \in \{1, 3, 5, 7\}$ at a time and retrain a new model. In our experiments we use a prespecified number of features to remove, where $K < no\,features$ (9 for TI, 9 for LR and 18 for TI+LR), and leave the dynamic feature selection as a future improvement of the proposed XAI methods. Finally, for each feature selection method, we compute the optimal threshold as the one that maximizes the mean loss difference between the base regressor (without any feature removed) and the corresponding newly trained model. In our experiments we consider two metrics to quantify the loss: the canonical `MSE` and the `mean return`[10]. When considering the `MSE`, the computation is straightforward and aligned with common practice. The `MSE` metric consists of computing the squared residual between the true observations (on the validation set) and the corresponding estimated values provided by the machine learning

[10]To distinguish between the MSE and mean return as an evaluation metric of the XAI trading strategies and the loss metric for the feature selection, we use the type-writer notation, *i.e.*, `MSE` and the `mean return`.

model. Instead, in the case of the `mean return` loss, we adopt the following approach. Considering a stock and its return forecast at time t, we "trade" the stock, *i.e.*,: long, if its return forecast is positive, or short, if its forecast is negative. Then, considering the actual stocks return, we can determine the daily stocks return when investing according to the forecast. The mean return is computed as the mean of the daily returns on the entire validation period. In this manner, we want to validate whether a financial metric, such as the mean return, can serve as a good proxy of the ML model performance. Or, differently, we try to answer the question whether the incorporation of a financial metric into the feature selection process can improve the financial performance of our overall XAI strategy.

Backtesting

We back-tested our trading strategy using the *walk-forward* validation approach, which represents a common practice in the related literature [116, 2]. The walk-forward validation consists of splitting the time interval into overlapping training and validation periods, and non-overlapping test (trading) periods, as shown in Figure 2.3. The example depicts values for the closing price of the AT&T stock over the whole study period, ranging from January 2003 to January 2016. Each triplet *(training, validation, test)* forms a walk. By sliding forward the triplet with the length of the test period, we construct the out-of-sample contiguous period from January 2007 to January 2016, *i.e.*, a total of nine years. Each walk entails three years for ML models training, one year of validation used for optimal threshold computation and feature selection, and one year considered as out-of-sample.

Trading execution and portfolio construction

The back-testing experiments consist in running the signals (*i.e.*, long and short) according to the ML models forecast and evaluating the performance against historical data. As stated already throughout the paper, StatArb consists of trading long and short the predicted k winners and k losers, respectively. We fix the number of pairs to be traded to $k = 5$, similarly to [2]. The trading session is set as intra-day, meaning that we are opening the positions at the beginning of the trading day and close them at the end of the day. In other words, we are rebalancing our portfolio daily.

5.4.2 Baselines

To assess the value added by the feature selection strategies, they are benchmarked against two statistical arbitrage trading baselines: the portfolio constructed using only the base regressors without having any features removed

(**BaseRegressors**), and the S&P500 Buy&hold strategy (**Buy&hold** [148]). The comparison against the **BaseRegressors** is quite natural and straightforward, as we aim to assess the performance improvements of the proposed XAI StatArb methods over the **BaseRegressors**. **Buy&hold** is instead a well-established quantitative strategy, and largely used as baselines to evaluate the profitability of other investment approaches [116, 113]. The strategy buys in our application in January 2007, and holds the S&P500 exchange-traded fund (SPY security) during the whole back-testing period, *i.e.*, until January 2016. This passive strategy runs without any trading signals.

5.4.3 Results and Discussion

The following section presents the results obtained by the proposed XAI StatArb strategy. In this regard, the results are discussed considering three perspectives: (i) goodness of fit of the ML models; (ii) returns generated over the entire trading period; and, finally, (iii) risk evaluation. Each of these evaluation results is quantified using the following metrics:

(i) Mean squared error (MSE) - one of the most popular goodness of fit measure for ML models for a continuous dependent variable, such as in our case.

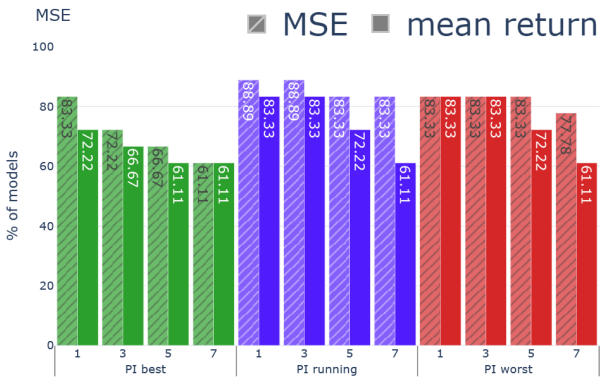
(ii) Annual return - represents the return on an investment generated over a year and calculated as a percentage of the initial amount of investment.

(iii) Maximum drawdown (MDD) - the maximum amount of wealth reduction that a cumulative return has produced from its maximum value over time.

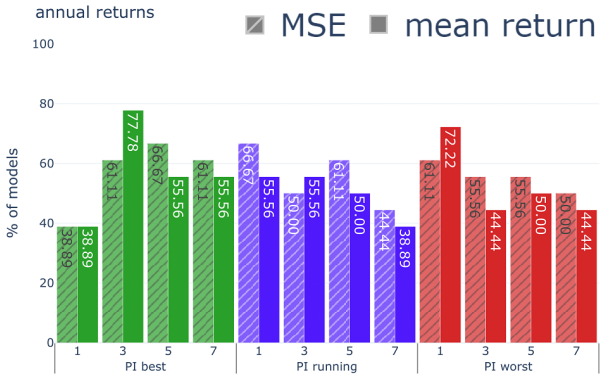
Each of these metrics have been previously discussed and formalized in Section 2.6.

Overview

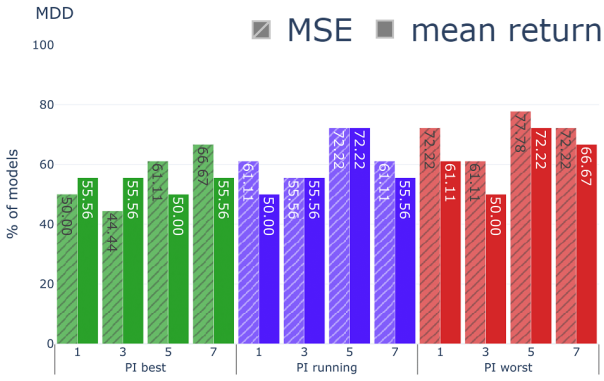
Briefly, three types of machine learning models are trained, *i.e.*, LGB, RF, and SVR. Each of them are fed with different types of input features, *i.e.*, LR, TI, or jointly LR and TI. Also, the models are built using two levels of data - stock or sector. Therefore, there are 3 types of models $\times$ 2 levels $\times$ 3 types of features, for an overall of 18 base regressors. For each of these combinations, the three strategies of feature selection are applied, namely *PI best*, *PI running*, and *PI worst*. To identify unimportant features (feature selection),to evaluate the loss two metrics are used, that is `MSE` or `mean return` (details provided in Section 5.4.1). Moreover, there are discarded $K = 1, 3, 5, 7$ unimportant features as given by each of the methods.



(a) MSE



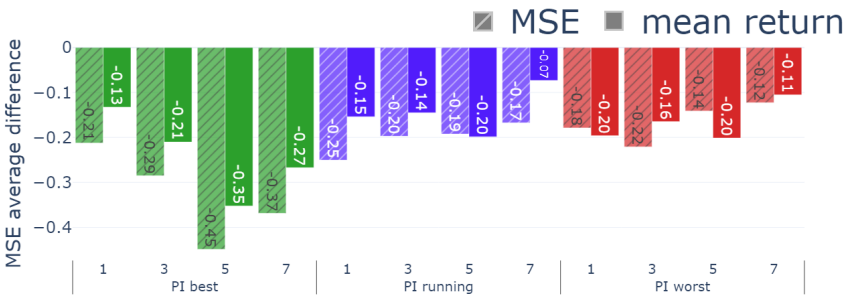
(b) Annual returns



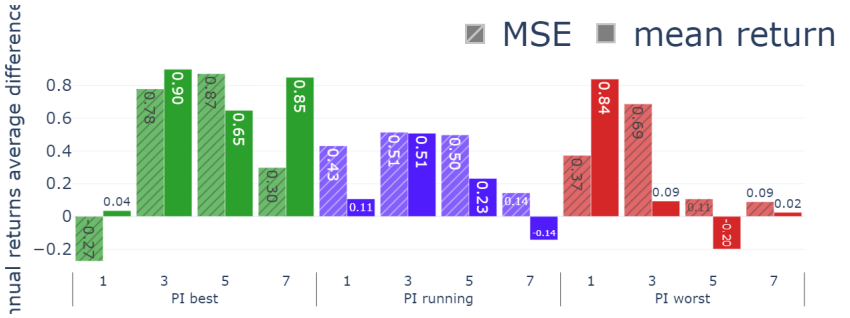
(c) MDD

Figure 5.6: Percentage of ML models that exhibit improvements over the **BaseRegressors** in terms of (a) MSE, (b) annual returns, and (c) MDD, developed from 2007 to 2016 for the three feature selection methods when removing $K = 1, 3, 5, 7$ features.

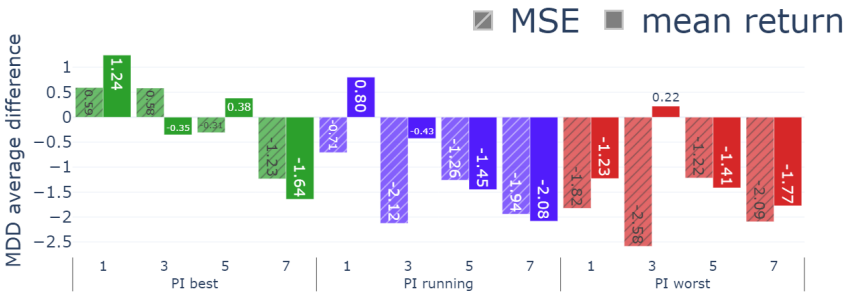
Firstly, Figure 5.6 shows an overview of the performances obtained by the proposed feature selection strategy compared to the **BaseRegressors**. Specifically, are shown the percentage of models that obtain improved performance compared to the **BaseRegressors**, given the three evaluation criteria, i.e., MSE, annual returns, and MDD. Additionally, side-by-side, the performances of the models are shown given that two loss metrics for feature



(a) MSE $\times 1e4$



(b) Annual returns



(c) MDD

Figure 5.7: Average difference between the ML models and the corresponding **BaseRegressors** in terms of (a) MSE, (b) annual returns, and (c) MDD, developed from 2007 to 2016 for the three feature selection methods when removing $K = 1, 3, 5, 7$ features.

selection, *i.e.*, `MSE` and `mean return`. Note that the `MSE` (as loss function for feature removal) is marked with diagonal lines, whereas the `mean return` has no filling pattern. For each performance evaluation criteria, we have on the x-axis the three feature removal methods strategies and the number of features that are discarded. The bars represent the percentage of models that have an improved performance over the corresponding **BaseRegressors**. Figure 5.6a (MSE evaluation) shows that more than half of the models (more than 9 models out of 18 for each x-value) have an improvement over the **BaseRegressors**, reaching high values such as 88.89%. The lowest performance is registered by *PI best*, which is an expected outcome given that unimportant features are removed, yet they might have higher importance than others. In terms of annual returns (Figure 5.6b), the performance improvements are not as prominent as in the case of the MSE criterion, nevertheless having the highest value of 78% and the lowest of 38%. Values below 50% indicate that the **BaseRegressors** have a better performance than the feature selection method in more than half of the used setups, i.e. ML model type, data level, input feature type. For the MDD, Figure 5.6c shows an interesting trend where all the models, for each method and number of features removed, have an enhancement. Also, one can see that the improvement is almost opposite to the percentage of models that have an annual return increase, which is an indication that a higher return comes with the cost of higher risk. Comparing the feature selection loss metrics, i.e., `MSE` versus `mean return`, one cannot deduct a clear indication whether one metric is emerging over the other.

These results are complemented by those depicted , where a qualitative assessment on the performance improvements between the different approaches is made. To note that in Figure 5.7 the same criteria are applied as for the experiments reported in Figure 5.6. Thus, for each feature selection method, loss metric, and number of removed features, Figure 5.7 illustrates the average differences of the MSE, annual returns, and MDD among the feature selection methods and their corresponding **BaseRegressors**. Starting with the forecasting performance evaluation, in Figure 5.7a negative values are sought, since lower MSE values relative to the ones obtained by the base regressors imply a better forecasting performance. Conversely, for annual returns (Figure 5.7b), positive values are preferable, since an increase of the annual return relative to the **BaseRegressors** assumes an improvement of the trading strategy when removing features. Furthermore, given that a higher MDD (Figure 5.7c) means a higher risk, negative values for this metric are desired as this denotes that the feature removal process decreases the trading strategy risk. Under these considerations, there can be stated that all the feature selection methods bring a performance improvement in the predictive performance (Figure 5.7a) since all the values are below 0.

In terms of both annual returns (Figure 5.7b) and MDD (Figure 5.7c), every feature removal method brings improvements with regards to the fi-

nancial performance, i.e., by increasing the returns or by lowering the risk. However, there are exceptions to the rule, such as *PI best*, when removing one feature. Also, it is evident a similar pattern as in Figure 5.6, namely, higher returns imply increased risk. Furthermore, crossing the two figures, another surprising finding can be deducted: although *PI best* has a lower number of models (for each x-value) that provide an improvement over the **BaseRegressors**, its improvements are significantly higher than *PI worst* or *PI running*.

Financial Performance of the XAI StatArb Trading Strategies

The results from the experiments reported in Figures 5.6 and 5.7 motivate us to take a closer look at the financial performance of the XAI StatArb strategies. Therefore, Figure 5.8 presents the return statistics and risk characteristics. Provided that in this Chapter a large number of models and configurations is built (3 types of models $\times$ 2 data levels $\times$ 3 types of features $\times$ 3 feature seclection methods $\times$ 2 feature selection metrics $\times$ 4 number of features removed $=$ 432 in total), in the following paragraphs are discussed only the performances of the best model per configuration as given by the annual return. Note that a configuration is provided by the ML model type, the data level, and the input feature type. Therefore, it is presented the XAI StatArb performances for 18 such configurations, in contrast to their corresponding **BaseRegressors** and **Buy&Hold** strategies, i.e., 6 XAI StatArb strategies per each ML model. Figure 5.8 shows a representative set of financial performance indicators, and, the right-most columns presents the differences between the annual returns/MDD of the XAI StatArb strategies and the **BaseRegressors**. The differences are color-coded: red colors represent values where the **BaseRegressors** have a better performance. Conversely, blue or green represent better performances of the strategies compared to the **BaseRegressors**. Also, the dimension of the bar is proportional to the difference.

As the reader can notice, all the models have a positive annual return, between 7% and 31%, well above the **Buy&Hold** strategy (4%). The best performing model is LGB with technical indicators (TI) as features. Moreover, LGB appears to be the best performing forecasting technique on average, with the highest annual returns in any given configuration. It reaches an average annual return of 18%, followed by RF with 14%, and, finally, SVR with 13%. This ranking is highly influenced by the data level as models per sector learn from a large number of observations. Consequently, SVR, that is notoriously unable to handle large datasets having a high number of observations,

Figure 5.8: Financial performance of the XAI StatArb trading strategies gross of trading costs and fees, over the trading period 2007-2016. The best two return performances are highlighted in bold.

Method	Feature Selection Metric	Feature Type	Model Type	Level	K	Annual return	Mean return	Q1	Q2	Q3	MDD	Annual return BaseRegressor	MDD BaseRegressor	Annual return difference	MDD difference
PI best	MSE	LR	LGB	sector	5	12.48	0.05	-0.37	0.03	0.45	25.22	9.67	26.44	2.81	-1.22
PI worst	MSE	LR+TI	LGB	sector	3	15.44	0.06	-0.36	0.04	0.46	30.53	10.22	38.89	5.22	-8.36
PI best	MSE	TI	LGB	sector	3	10.62	0.04	-0.33	0.03	0.42	31.04	8.06	32.71	2.56	-1.67
PI best	MSE	LR	LGB	stock	7	18.40	0.07	-0.34	0.06	0.46	14.54	16.43	18.76	1.97	-4.22
PI worst	mean return	LR+TI	LGB	stock	5	20.57	**0.08**	-0.32	0.06	0.45	14.39	18.43	16.90	2.14	-2.51
PI best	mean return	TI	LGB	stock	5	**31.15**	**0.11**	-0.30	0.08	0.49	12.36	**29.86**	12.20	1.29	0.16
Average return						18.11									
PI best	mean return	LR	RF	sector	5	16.93	0.07	-0.36	0.04	0.49	13.48	5.68	30.27	11.25	-16.79
PI worst	mean return	LR+TI	RF	sector	7	9.98	0.04	-0.41	0.02	0.43	27.12	10.01	40.76	-0.03	-13.64
PI running	mean return	TI	RF	sector	3	13.37	0.05	-0.36	0.04	0.42	32.90	11.10	33.13	2.27	-0.23
PI best	mean return	LR	RF	stock	5	15.78	0.06	-0.35	0.05	0.48	16.72	12.42	22.69	3.36	-5.97
PI worst	mean return	LR+TI	RF	stock	3	10.79	0.04	-0.39	0.03	0.46	30.10	8.21	27.00	2.58	3.1
PI best	mean return	TI	RF	stock	5	15.29	0.06	-0.35	0.03	0.42	17.56	11.96	22.64	3.33	-5.08
Average return						13.69									
PI best	mean return	LR	SVR	sector	3	11.88	0.05	-0.39	0.05	0.50	30.71	10.86	25.89	1.02	4.82
PI worst	mean return	LR+TI	SVR	sector	1	7.29	0.03	-0.32	0.04	0.42	51.67	2.42	51.44	4.87	0.23
PI best	MSE	TI	SVR	sector	3	6.92	0.03	-0.31	0.03	0.39	52.90	3.87	49.18	3.05	3.72
PI worst	mean return	LR	SVR	stock	1	**21.45**	**0.08**	-0.34	0.05	0.46	12.48	**19.25**	12.39	2.2	0.09
PI best	MSE	LR+TI	SVR	stock	7	12.54	0.05	-0.32	0.05	0.41	11.74	11.70	12.18	0.84	-0.44
PI best	mean return	TI	SVR	stock	7	17.65	0.07	-0.33	0.05	0.43	15.51	10.06	19.25	7.59	-3.74
Average return						12.96									
Buy-and-Hold						4.16	0.03	-0.48	0.06	0.60	56.47				

obtains the worst performance when trained at sector level versus stock level.

In terms of risk, all the models have a lower MDD than the **Buy&Hold**, with the lowest value (best performing) of 12% (SVR, stock level), and the highest value (worst performing) of 52% (SVR, sector level).

Concerning the best-performing feature selection method, the reader can notice that out of the (best) 18 configurations presented, *PI best* yields the best returns in 11 such cases, *PI worst* in 6 cases, and *PI running* only in 1. Furthermore, concerning the annual return difference, the *PI best* attains the leading positions with values such as 11.25 (sector RF with LR) and 7.59 (stock SVR with TI). The results confirm the findings in Figure 5.7, where the average difference between the annual returns of the feature selection methods and **BaseRegressors** is higher for the *PI best* strategy.

Concerning the feature selection metric, one can note in Figure 5.8 that the `mean return` has almost twice the occurrences over the `MSE`, meaning that changing the canonical MSE with the `mean return` introduces a significant benefit in the financial performance.

Although all the feature selection methods introduce an improvement in terms of returns, when it comes to the risk, six of them show a higher MDD compared to the **BaseRegressors** (i.e., positive values in the MDD difference column), signifying that these trading strategies are more exposed to financial risks. Moreover, four of the XAI StatArb strategies are based on SVR models. One could only suppose that XAI enabled SVR are more sensitive to outliers and produce forecasts less calibrated and accurate on the prediction interval

extremes. This translates into more volatile down movements.

Sub-Period Analysis

In the following, a sub-period breakdown is shown in order to provide more details about the performance of the XAI StatArb strategies in terms of returns and risk profiles. Table 5.3 presents such details. Additionally,Figure 5.9 illustrates the cumulative profits of the strategies.

Table 5.3: Sub-period analysis of the XAI StatArb trading strategies performance. The best two financial returns are highlighted in bold.

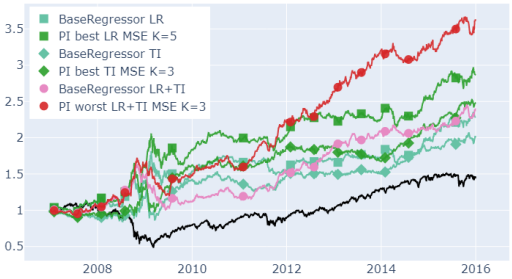
Method	Feature Selection Metric	Feature Type	Model Type	Level	K	Low Volatility Bull Market						GFC Crash						New Normal					
									Base regressor						Base regressor						Base regressor		
						Mean Return	SD	MDD	Mean Return	SD	MDD	Mean Return	SD	MDD	Mean Return	SD	MDD	Mean Return	SD	MDD	Mean Return	SD	MDD
PI best	MSE	LR	LGB	sector	5	0.01	0.54	9.34	0.01	0.54	9.34	0.16	1.76	25.22	0.12	1.74	24.60	0.03	0.74	14.92	0.03	0.72	15.16
PI worst	MSE	LR+TI	LGB	sector	3	-0.03	0.41	7.69	-0.03	0.42	8.29	0.09	1.33	21.60	0.05	1.24	29.25	0.06	0.65	13.50	0.04	0.68	16.20
PI best	MSE	TI	LGB	sector	3	-0.07	0.49	11.07	-0.07	0.48	10.58	0.06	1.61	31.04	0.02	1.61	32.71	0.05	0.61	12.03	0.05	0.63	17.84
PI best	MSE	LR	LGB	stock	7	**0.10**	0.53	4.45	**0.10**	0.51	3.73	0.17	1.18	14.54	0.11	1.15	18.76	0.04	0.66	11.35	0.05	0.65	11.95
PI worst	mean return	LR+TI	LGB	stock	5	**0.11**	0.45	4.16	**0.11**	0.45	4.16	0.15	1.37	12.58	**0.15**	1.38	13.42	0.06	0.61	14.39	0.05	0.61	16.90
PI best	mean return	TI	LGB	stock	5	0.07	0.61	5.56	0.03	0.58	6.77	**0.24**	1.18	12.36	**0.22**	1.14	12.20	**0.08**	0.65	10.68	**0.08**	0.65	10.22
PI best	mean return	LR	RF	sector	5	-0.04	0.47	9.32	0.02	0.52	7.03	0.13	1.24	13.48	0.04	1.45	30.27	0.06	0.68	11.52	0.02	0.71	16.07
PI worst	mean return	LR+TI	RF	sector	7	-0.05	0.49	12.49	-0.04	0.48	11.39	0.08	1.60	27.12	0.04	1.69	40.07	0.04	0.69	11.95	0.05	0.70	11.98
PI running	mean return	TI	RF	sector	3	0.01	0.51	5.21	-0.03	0.50	6.06	0.06	1.74	32.90	0.07	1.73	33.13	0.06	0.67	12.06	0.05	0.66	14.96
PI best	mean return	LR	RF	stock	5	0.06	0.51	4.46	0.08	0.49	3.38	0.08	1.20	14.13	0.04	1.26	22.69	0.06	0.65	16.72	0.05	0.64	15.94
PI worst	mean return	LR+TI	RF	stock	3	0.06	0.52	3.69	0.02	0.49	4.79	0.15	1.32	13.77	0.12	1.29	13.58	0.02	0.68	30.10	0.02	0.68	27.00
PI best	mean return	TI	RF	stock	5	0.01	0.55	9.35	0.02	0.56	8.48	0.11	1.30	17.51	0.05	1.40	22.64	0.05	0.58	10.00	0.05	0.59	15.29
PI best	mean return	LR	SVR	sector	3	0.02	0.43	3.71	-0.03	0.49	6.32	0.09	1.81	30.71	0.08	1.70	25.89	0.04	0.69	10.27	0.04	0.69	11.78
PI worst	mean return	LR+TI	SVR	sector	1	0.02	0.42	3.13	0.01	0.44	4.39	-0.06	1.47	42.00	-0.07	1.47	42.22	0.05	0.62	21.79	0.03	0.62	21.10
PI best	MSE	TI	SVR	sector	3	0.06	0.50	6.14	0.04	0.47	6.47	-0.11	1.85	52.90	-0.09	1.93	47.93	**0.07**	0.61	8.78	0.05	0.60	14.40
PI worst	mean return	LR	SVR	stock	1	0.05	0.48	3.64	0.05	0.46	4.08	0.15	1.62	12.48	0.13	1.59	12.39	**0.07**	0.64	10.05	**0.06**	0.64	10.41
PI best	MSE	LR+TI	SVR	stock	7	0.01	0.44	5.19	0.00	0.44	5.19	0.09	0.99	7.87	0.09	1.00	8.13	0.04	0.60	11.74	0.04	0.59	12.18
PI best	mean return	TI	SVR	stock	7	0.01	0.54	7.30	-0.01	0.51	6.59	**0.24**	1.38	10.63	0.09	1.37	19.25	0.03	0.58	15.51	0.03	0.58	18.33
Average Returns						0.03			0.01			0.10			0.07			0.05			0.04		
Buy&Hold						0.03	0.80	6.42				-0.11	2.32	56.47				0.05	1.18	19.42			

For this round of experiments, three study periods are scrutinized.

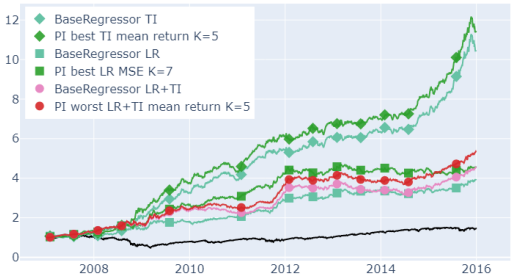
Low Volatility Bull Market - represents a short period of only seven months, ranging from January 2007 to August 2007. In this particular period, the XAI StartArb methods show a large diversity of returns which is hard to explain. The mean return is either strongly positive and higher than the **Buy&Hold** (0.03%), e.g., *PI worst* LGB stock level LR+TI (0.11%), or economically non-significant, e.g., *PI worst* RF sector LR+TI (-0.05%). Works such as [113, 111] attribute similar results to the increase of computational power that made algorithmic trading readily available, and the consequent imprint in the market behavior and decrease of the returns. By comparing the average returns

Figure 5.9: Daily compounded returns gross of trading costs and fees for the XAI StatArb trading strategies compared to their corresponding **BaseRegressors** and **Buy&Hold**, respectively. The trading period ranges from 2007 to 2016. On the left-hand side, Figures 5.9a, 5.9c, and 5.9e represent *sector* level models, whereas, on the right-hand-side, Figures 5.9b, 5.9d, and 5.9f represent *stock* level models. The **Buy&Hold** is represented by a continuous black line. Input features are represented as: LR - squares, TI - diamonds, and LR+TI - circles.

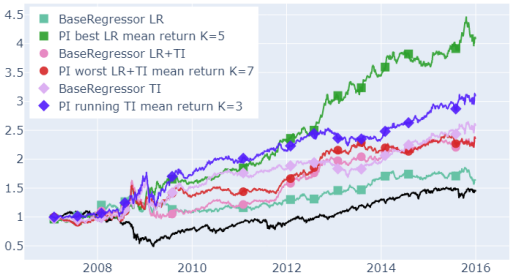
(a) LGB sector



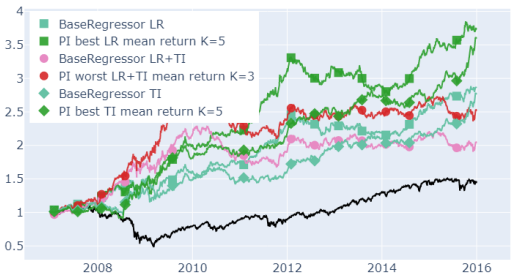
(b) LGB stock



(c) RF sector

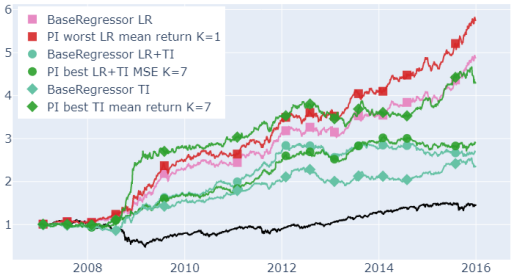


(d) RF stock



(e) SVR sector

(f) SVR stock



of the XAI StatArb methods to the average returns of the **BaseRegressors** (second-last row in Table 5.3), one can assess that the feature selection process, on average, introduces an improvement, that is, the average return increases from 0.01% to 0.03%.

GFC Crash - corresponds to the period of the Global Financial Crisis (GFC) starting, specifically, in August 2007, and finishing in April 2009. In this turbulent period, with the market with a steep decline and a very high level of VIX index [111], the XAI StatArb strategies are most proliferating. For example, *PI best* LGB TI stock obtains an all-time high of 0.24% daily mean return. By comparison, in the previous period, it reaches only a 0.07% of daily mean return, or, in the next period, only a 0.08%. The performance of the portfolios alternates rapidly between significant rises and falls, a fact confirmed by the higher standard deviations (SD) of the returns. The same can be reaffirmed by looking at the cumulative returns presented in Figure 5.9. This figure also reveals that this behavior is more noticeable in the case of sector-level models. Comparable findings have been confirmed in the literature as well. For example, [113] investigated the role that input features play in the StatArb financial performance, in the period of GFC, reporting for the best performing combination of ML models and input features a daily mean return of 0.3%. However, the author have reported as the average of daily returns of the models as being 0.01%, that is significantly lower than the one of our XAI StatArb strategies (0.10%) or our **BaseRegressors** (0.07%). Even though most XAI StatArb methods exhibit positive daily returns, not all models perform notably well in this period; e.g., *PI best* SVR sector TI is the laggard of the whole set of methods and has a performance similar to the **Buy&Hold** both in terms of returns and risk. Both *PI best* SVR sector TI and **Buy&Hold** have a mean return of -0.11%. The MDD for *PI best* SVR sector TI is 53% and 56% for **Buy&Hold**. However, in Table 5.3, only 2 out of 18 methods register negative returns during this period. [113] reported 17 out of 44 models with negative mean daily returns, the worst performing model having -0.28%.

New Normal - the period ranges from 2009-04-01 to 2015-12-31 and corresponds to the recovery after the GFC, including short-lived market events such as the European Debt Crisis of 2011, the period between April and October 2014, or the market down-turn in August-September 2015. Nevertheless, this time-span accounts for a bull market period. In the literature, for the same period, [111, 122] reported modest returns by comparison with the other periods, even reaching a plateau. In this interval, investing strategies such as **Buy&Hold** are particularly challenging as baselines, as the literature brings empirical evidence that StatArb trading strategies are not as profitable as they used to be. Even under such setback, Table 5.3 shows that the XAI StatArb manages to obtain mean daily returns above the **Buy&Hold**. Indeed, the **Buy&Hold** strategy attains a return of 0.5% on par with the average

returns of the XAI StatArb methods. Inspecting Figure 5.9, the reader can notice that most of the models show a positive trend. But indeed, some of them reach a plateau after 2012, e.g., *SVR stock level LR+TI* or *LGB stock level LR*. Contrarily, *PI best LGB stock level TI* has a steep and positive curve in the most recent period between 2014-2015. Such results are attributed to the trait that tree-based methods such as LGB have, that is, their robustness to noise and outliers that make them particularly suitable to a financial forecasting setting.

5.5 Conclusions

This chapter proposes a machine learning approach powered by explainability techniques as an integrative part of an algorithmic trading pipeline. It aims at bridging the gap between typical financial practice (employing machine learning approaches) and robust explainable artificial intelligence applied on a large stock set. At the same time, this work aims to extend the existing literature on statistical arbitrage trading based on machine learning. We follow the well-established steps of a statistical arbitrage trading system: firstly forecasting the stock price returns, then ranking the stocks based on the predicted returns, and lastly, trading several pairs of stocks. The presented applications and forecasting models, focusing on the U.S. market, are based on various input features and report results on a trading period ranging from 2007-2016. Different than standard machine learning approaches, we do not simply use a framework to produce buy and sell signals. Instead, we introduce a feature selection step so that the machine learning algorithms predict the next day returns on a representative/informative feature set. To achieve this goal, we propose three methods for feature selection: *PI best*, *PI running*, and *PI worst*. The methods have the key traits of being model agnostic, and of performing the feature selection process in a post-hoc manner without interfering with the prediction and, most importantly, learning the relevant features both at stock/sector level and in cross-sections (*i.e.*, globally). We test our proposed methods by considering several setups: various machine learning algorithms (*i.e.*, Gradient Boosting, Random Forests, and Support Vector Regressors), different input feature sets (*i.e.*, lagged returns, technical indicators, and their combination), and distinct data levels (*i.e.*, sector or stock levels), which yielded to an overall of 432 such models. By conducting such extensive testing from predictive as well as financial performance angles, several discussion points emerge. The feature selection methods introduce clearly improvements both in terms of predictive and financial performance. All raw returns are positive. Given that more predictors translate into potentially more noise, there is a clear improvement over the compared base regressor (*i.e.*, the machine learning model without any features removed)

when dropping several features. Besides, replacing the widely-used MSE with the mean return as a loss function within the feature selection process constitutes an essential change of the XAI StatArb strategies. In terms of annual returns, our approach brings the most significant enhancements over the base regressors.

Chapter 6

Conclusions

> *"Though we live in a world that dreams of ending*
> *That always seems about to give in*
> *Something that will not acknowledge conclusion*
> *Insists that we forever begin"*
>
> **Brendan Kennely**

6.1 Concluding Discussion

Along the different chapters of this thesis, our goal was twofold as reflected by its title: "Artificial Intelligence-driven Systems for Financial Forecasting". In this dissertation, with careful consideration the building blocks of a trading system are explored, and most importantly how artificial intelligence models can play an integrative role in such a system. Furthermore, overarching aim is at shedding new light on artificial intelligence-driven trading system assessing its capabilities, inner workings, and, ultimately, interpretability.

First, in Chapter 4 of this dissertation it is proposed a general approach for risk-controlled trading which employs machine learning to perform forecasting and statistical arbitrage as a trading strategy. The aim is forecasting the intraday returns by using a heterogeneous ensemble of learning models. To achieve the heterogeneity of the ensemble, first a pool of regressors is built using various learning algorithms such as Light Gradient Boosting, Random Forests, Support Vector Machines and even statistical methods like ARIMA, thus enforcing *model diversity*. Each of the models is trained with a complementary set of features *i.e.*, lagged returns and technical indicators, to account for *feature diversity*. Furthermore, the *data diversity* is ensured by simultaneously training models across several stocks (as given by their sector) and, conversely, models for each stock. In conclusion, for each stock, 3 types of machine learning algorithms are trained and each of them for 2 levels of

data, and 2 types of features. This yields: 3 types of algorithms $\times$ 2 data level $\times$ 2 types of features = 12 models for each stock. The proposal capitalizes on the fact that for a given input feature type, the performance of one machine learning algorithm may not be correlated with that of another algorithm. As a consequence, in this thesis it is proposed a mechanism to select the best model per machine learning algorithm type, resulting in 3 models. To these 3 regressors, for each stock, the ARIMA model is added, which is most suitably used with univariate time series (equivalent to stock level and lagged returns features).

Without a doubt, the increase in processing capacity make statistical arbitrage easier to implement. Nonetheless, profiting from statistical arbitrage remains challenging. According to the non-arbitrage principles, statistical arbitrage opportunities are temporary and cannot last for long. Despite arbitrage opportunities being generated by market frictions and price variances, they can also be immediately eliminated by markets' self-restoring function, making them difficult to identify. To tackle this challenge, a dynamic asset selection mechanism is developed which operated based on models' most recent prediction performance.

The proposed approaches is evaluated over several year that encompass high market turmoil such as the global financial crisis or bull periods (market growth) such as periods following 2012, thus, enabling a deep insights in terms of return and risk exposure.

Chapter 5 of this dissertation analyzed and discussed the interpretability of financial forecasting in the eyes of variable importance measures. Our empirical analysis is motivated by the fact that in the machine learning industry there are several frameworks that aid practitioners in developing derived features from raw data. Moreover, along decades scientist from the financial domain have done unremitting efforts to build statistical tools such as the technical indicators, in order to build better trading strategies. These tools are widely used now as input features to the machine learning models for forecasting and *smart* algorithmic trading. While extensive feature engineering may bring the benefit of better training the models, it is very likely to yield overfitted models in most scenarios.

The core of our contributions rests in the introduction a feature selection step so that the machine learning algorithms predict the next day returns on a representative/informative feature set. To achieve this goal, in this dissertation three methods for feature selection are proposed: *PI best*, *PI running*, and *PI worst*. The methods have the key traits of being model agnostic, and of performing the feature selection process in a post-hoc manner without interfering with the prediction and, most importantly, learning the relevant features both at stock/sector level and in cross-sections (*i.e.*, globally). Therefore the method is general and applicable to any predictive model. Chapter 5 is structured in two parts. The first one assesses the proposed methods from

a goodness of fit perspective and focuses solely on the predictive performance of the employed machine learning model,*i.e.*, Random Forests. In this preliminary part, the focus is on removing a single noisy feature. The second part, presents the feature selection methods in an algorithmic trading scenario where the underlying trading strategy is statistical arbitrage. Furthermore, it extends the empirical study from a single type of predictive model to three types of models, *i.e.*, Light Gradient Boosting, Random Forests, Support Vector Machines and removes up to 7 features from the input features used.

Overall, although forecasting is a problem that machine learning algorithms are good at solving, not all of the analysed periods yield valuable forecasts. The reader needs to bear in mind that the forecasting performance is assessed through the lens of financial performance of the entire trading system. In the empirical study, returns are demonstrably stronger when volatility is high. Conversely, when volatility is low, successful forecasting is demonstrated only on limited time periods. The same pattern is shown in Chapter 4 and Chapter 5. This is attributed to several factors. Statistical arbitrage relies on market pricing anomalies. Therefore, the increase in arbitrage risks during periods of panic was outweighed by a corresponding decrease in market efficiency. Thus, some the models have overcome worsening arbitrage opportunities to successfully exploit mispricings that appear to be abundant in such turbulent periods. Conversely, in periods of bull market, the market exhibits a stronger form of efficiency from which our machine learning-enabled strategies do not thrive. The second reason behind a decreased forecasting efficiency is the fact during the last decade the adoption of machine learning in the financial sector has increased. As a testimony stands several trust funds like Two Sigma, Point 72, or Bridgewater, to name a few, that promote the usage of artificial intelligence models in their trading activities. In this context, markets tends to correct itself with a higher frequency, fewer mispricing occur, with no discernible patters. Despite this unfavorable context, in both empirical studies of this thesis economically significant results are achieved.

On another train of thought, the question that arises is whether the benefits associated to the machine learning in an algorithmic trading system can justify their usage in production. Unlike statistical models, which are designed as an aggregation of rules to be directly executed in a production system, machine learning models are built on algorithms requiring intensive computations and, consequently, high training time. For instance, in the approach proposed in Chapter 4, one may notice that the most intensive step is represented by the training stage. This block's cost is proportional to the number of stocks or sectors and model hyperparameters. Then, the most computationally expensive process of the testing stage is the calculation of the rolling mean directional accuracy, which can be considered negligible in the general mechanics of the proposed method. Although I argue that the integration of such an architecture into a production environment is feasible, to

reduce the computational burden, more elaborate training implementations, such as parallel model training, must be taken into account. In Chapter 5 it is proposed a strategy of feature selection based on the feature importance score that, in turn, is derived by a permute-and-predict approach. Then, the features that have a score below a certain threshold are removed and a new model is trained. Therefore, the computational cost that is payed for whitening the "black-box" learning model is proportional to the number of models (*i.e.*, per stock or industry) and the features that are discarded. I argue that such a pipeline is computationally intensive only at the training stage and can be readily used in a production scenario.

6.2 Future Work

While the proposed approaches have proven to yield economically significant results, they show limitations as well. In Chapter 4, the ensembling strategy has a fixed-length window size on which it operates, *i.e.*, it "learns" the best performing model during a validation period. By the same token, the feature selection strategies that are proposed in Chapter 5 use a fixed-length validation period to determine in cross-section the important features. Although both the proposed approaches are in line with the standard time-series methodologies, *i.e.*, we do not introduce any look-ahead bias, the procedure falls short when considering the rapid changes in market dynamics and the fact that more recent observations have higher semantics than the ones from more distant history. On the other hand, shortening the validation length to less than one year (as in the carried out) may be detrimental to the base-learner evaluation as well as feature score computation and feature selection. That is because the metrics computed would not be statistically significant. Therefore, one direction that could further improve the results is to move toward an online learning approach. Specifically for ensemble creation better suited could be a *dynamic ensemble creation*, which means selecting on the fly the pool of experts that participate in the average ensemble. For the feature selection process, a key research direction is dynamically adjusting the length of the score computation window to be able to capture the local time variations of the features.

Moreover, given that is widely acknowledged that the financial market has structural breaks and "suffers" from regime switches, another direction of research could be adding to the input features set, metrics that quantify the likelihood of structural breaks.

Also, considering that majority of algorithms perform better both in terms of predictive performance and financial performance when using a model for each stock instead model for each sector, another setup that can be investigated is considering models for stocks that take part in the same sub-industry,

which would lead to 44 models. In other words, is necessary to find the right size for the group of stocks, as an increased number of models can prove costly to train.

Overall, machine learning holds great promise for financial prediction and insight, but it is crucial for investors to apply domain expertise and intuition as part of the process. What machine learning does today is limited to automating specific tasks within asset management, often with some form of human intervention at the implementation stage. In fact, there is not much new about the machine learning techniques used in finance, and they have been around as part of statistics for a long time. Machine learning's ability to capture complex and nonlinear relationships from the ever-growing volumes of data, including textual ones that are relatively time-consuming for humans to analyze, has proven to be highly beneficial. One can imagine that machine learning's footprint will only increase as financial practitioners compete for more information at higher speeds.

Bibliography

[1] Salvatore Carta, Diego Reforgiato Recupero, Roberto Saia, and Maria Madalina Stanciu. A general approach for risk controlled trading based on machine learning and statistical arbitrage. In *The Sixth International Conference on Machine Learning, Optimization, and Data Science (LOD 2020)*, volume 12565, pages 489–503. Lecture Notes in Computer Science, 2020.

[2] Salvatore Carta, Sergio Consoli, Sebastian Podda, Diego Reforgiato Recupero, and Madalina Stanciu. Ensembling and dynamic asset selection for risk-controlled statistical arbitrage. *IEEE Access*, 9:29942–29959, 2021.

[3] Salvatore Carta, Sebastian Podda, Diego Reforgiato Recupero, and Maria Madalina Stanciu. Explainable AI for financial forecasting. In *The 7th International Conference on Machine Learning, Optimization, and Data Science (LOD 2021)*. Lecture Notes in Computer Science, to appear, 2021.

[4] Gero John S and Fay Sudweeks. *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*. Springer, Dordrecht, 1996.

[5] S. Chatrchyan, V. Khachatryan, and A.M. Sirunyan. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012.

[6] A. Criminisi and J. Shotton, editors. *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer London, London, 2013.

[7] David A. Ferrucci. Ibm's watson/deepqa. *SIGARCH Comput. Archit. News*, 39(3), jun 2011.

[8] Söhnke M Bartram, Jürgen Branke, and Mehrshad Motahari. Artificial intelligence in asset management. *Centre for Economic Policy Research Discussion Paper*, 2020.

[9] Cristina Abad, Sten A Thore, and Joaquina Laffarga. Fundamental analysis of stocks by two-stage dea. *Managerial and Decision Economics*, 25(5):231–241, 2004.

[10] Thomas Oberlechner. Importance of technical and fundamental analysis in the european foreign exchange market. *International Journal of Finance & Economics*, 6(1):81–93, 2001.

[11] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, and Herbert Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, 124:226–251, 2019.

[12] Yumo Xu and Shay B. Cohen. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[13] Burton Gordon Malkiel. *A random walk down Wall Street: including a life-cycle guide to personal investing*. Norton, New York, 5th ed edition, 1990.

[14] Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970.

[15] William F. Sharpe. Mutual fund performance. *The Journal of Business*, 39(1):119–138, 1966.

[16] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley Publishing, 1st edition, 2018.

[17] Joseph Simonian and Frank J. Fabozzi. Triumph of the empiricists: The birth of financial data science. In *The Journal of Financial Data Science*, 2019.

[18] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[19] Stefan Jansen. *Machine learning for algorithmic trading predictive models to extract signals from market and alternative data for systematic trading strategies with Python, second edition*. Packt Publishing, 2020. OCLC: 1266678629.

[20] Charles D. Kirkpatrick and Julie R. Dahlquist. *Technical analysis: the complete resource for financial market technicians*. FT Press Financial Times, Upper Saddle River, N.J, 2007. OCLC: ocm67345945.

[21] John J. Murphy and John J. Murphy. *Technical analysis of the financial markets: a comprehensive guide to trading methods and applications*. New York Institute of Finance, New York, 1999.

[22] Jack D. Schwager. *Getting started in technical analysis*. Getting started in. John Wiley, New York, 1999.

[23] Jack D. Schwager. *Stock market wizards*. Harper Collins, 2001.

[24] Paul V. Azzopardi. *Behavioural technical analysis: an introduction to behavioural finance and its role in technical analysis*. Harriman House, Hampshire, 2010.

[25] Andrew W. Lo and J. Hasanhodzic. *The Evolution of Technical Analysis: Financial Prediction from Babylonian Tablets to Bloomberg Terminals*. Bloomberg. Wiley, 2011.

[26] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[28] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA, 1990.

[29] J. Scott Armstrong. Principles of forecasting. a handbook for researchers and practitioners. *Technological Forecasting and Social Change*, 69(3):313 – 316, 2002.

[30] Adebiyi A. Ariyo, Adewumi. O. Adewumi, and Charles K. Ayo. Stock price prediction using the arima model. In *16th International Conference on Computer Modelling and Simulation (UKSim-AMSS)*, pages 106–112, 2014.

[31] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time series analysis: forecasting and control*. Wiley series in probabilit and statistics. John Wiley & Sons, Inc, Hoboken, New Jersey, fifth edition edition, 2016.

[32] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14:771–780, 1999.

[33] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[34] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[35] Leo Breiman. Bias, variance, and arcing classifiers. *The Annals of Statistics*, 26:801–824, 1996.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd edition)*. Springer Series in Statistics, New York, 02 2009.

[37] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154, 2017.

[38] Vladimir N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[39] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 155–161. MIT Press, 1997.

[40] Luca Barbaglia, Sergio Consoli, Sebastiano Manzan, Diego Reforgiato Recupero, Michaela Saisana, and Luca Tiozzo Pezzoli. *Data Science Technologies in Economics and Finance: A Gentle Walk-In*, pages 1–17. Springer International Publishing, Cham, 2021.

[41] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable machine learning – a brief history, state-of-the-art and challenges. In I. Koprinska et al., editor, *ECML PKDD 2020 Workshops*, pages 417–431, Cham, 2020. Springer International Publishing.

[42] Amina Adadi and Mohammed Berrada. Peeking inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[43] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18, March 2010.

[44] Aaron J. Fisher, Cynthia Rudin, and Francesca Dominici. Model class reliance: Variable importance measures for any machine learning model class, from the "rashomon" perspective. In *arXiv:1801.01489*, 2018.

[45] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, pages 431–439. Curran Associates, Inc., 2013.

[46] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[47] Eugene F. Fama and Kenneth R. French. Forecasting profitability and earnings. *The Journal of Business*, 73(2):161–175, 2000.

[48] Darren D. Lee, Howard Chan, Robert W. Faff, and Petko S. Kalev. Short-term contrarian investing—is it profitable? ... yes and no. *Journal of Multinational Financial Management*, 13(4):385–404, 2003. Globalization and Financial Market Integration.

[49] Marco Avellaneda and Jeong-Hyun Lee. Statistical arbitrage in the us equities market. *Quantitative Finance*, 10(7):761–782, 2010.

[50] Marco Lazzarino, Jenny Berrill, Aleksandar Šević, Marco Lazzarino, Jenny Berrill, and Aleksandar Šević. What is statistical arbitrage? *Theoretical Economics Letters*, 8:888–908, 3 2018.

[51] Babak Mahdavi Damghani. The non-misleading value of inferred correlation: An introduction to the cointelation model. *Wilmott*, 2013(67):50–61, 2013.

[52] James Jayko. A Demon of Our Own Design: Markets, Hedge Funds, and the Perils of Financial Innovation. *Journal of Pension Economics and Finance*, 7(3):363–363, 2008.

[53] Evan G. Gatev, William N. Goetzmann, and K. Geert Rouwenhorst. Pairs Trading: Performance of a Relative-Value Arbitrage Rule. *The Review of Financial Studies*, 19(3):797–827, 02 2006.

[54] Christopher Krauss. Statistical arbitrage pairs trading strategies: Review and outlook. *Journal of Economic Surveys*, 31(2):513–545, apr 2017.

[55] Evan G. Gatev, William N. Goetzmann, and K. Geert Rouwenhorst. Pairs trading: Performance of a relative value arbitrage rule. *Capital Markets: Market Efficiency*, 1998.

[56] Andrew W. Lo. *Hedge Funds: An Analytic Perspective*. Princeton University Press, 2010.

[57] Amir E. Khandani and Andrew W. Lo. What happened to the quants in August 2007? Evidence from factors and transactions data. *Journal of Financial Markets*, 14(1):1 – 46, 2011.

[58] Richard C. Grinold. The fundamental law of active management. *The Journal of Portfolio Management*, 15(3):30–37, 1989.

[59] Eugene F. Fama. Efficient capital markets: Ii. *The Journal of Finance*, 46(5):1575–1617, 1991.

[60] George S. Atsalakis and Kimon P. Valavanis. Surveying stock market forecasting techniques – Part II: Soft computing methods. *Expert Systems with Applications*, 36(3):5932–5941, 2009.

[61] Rodolfo C. Cavalcante, Rodrigo C. Brasileiro, Victor L.F. Souza, Jarley P. Nobrega, and Adriano L.I. Oliveira. Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, 55:194–211, 2016.

[62] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, Jul 2019.

[63] Ankit Thakkar and Kinjal Chaudhari. A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions. *Expert Systems with Applications*, 177:114800, 2021.

[64] Gourav Kumar, Sanjeev Jain, and Uday Pratap Singh. Stock market forecasting using computational intelligence: A survey. *Archives of Computational Methods in Engineering*, 28(3):1069–1101, May 2021.

[65] Yun peng Wu, Jia min Mao, and Wei feng Li. Predication of futures market by using boosting algorithm. In *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1–4, March 2018.

[66] Ya-Wen Chang Chien and Yen-Liang Chen. Mining associative classification rules with stock trading data–a ga-based method. *Knowledge-Based Systems*, 23(6):605–614, 2010.

[67] Bin Weng, Lin Lu, Xing Wang, Fadel M Megahed, and Waldyn Martinez. Predicting short-term stock prices using ensemble methods and

online data sources. *Expert Systems with Applications*, 112:258–273, 2018.

[68] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[69] Jie Wang, Jun Wang, Wen Fang, and Hongli Niu. Financial time series prediction using elman recurrent random neural networks. *Computational intelligence and neuroscience*, 2016, 2016.

[70] Jie Wang and Jun Wang. Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks. *Neurocomputing*, 156:68–78, 2015.

[71] João Nobre and Rui Ferreira Neves. Combining principal component analysis, discrete wavelet transform and xgboost to trade in the financial markets. *Expert Systems with Applications*, 125:181 – 194, 2019.

[72] Chi-Jie Lu. Integrating independent component analysis-based denoising scheme with neural network for stock price prediction. *Expert Systems with Applications*, 37(10):7056–7064, 2010.

[73] Sheikh M. Idrees, M. Afshar Alam, and Parul Agarwal. A prediction approach for stock market volatility based on time series data. *IEEE Access*, 7:17287–17298, 2019.

[74] Salvatore Carta, Andrea Medda, Alessio Pili, Diego Reforgiato Recupero, and Roberto Saia. Forecasting e-commerce products prices by combining an autoregressive integrated moving average (arima) model and google trends data. *Future Internet*, 11:5, 2019.

[75] H. P. S. D. Weerathunga and A. T. P. Silva. Drnn-arima approach to short-term trend forecasting in forex market. In *International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 287–293, Sep. 2018.

[76] Jianxue Chen. Svm application of financial time series forecasting using empirical technical indicators. In *2010 International Conference on Information, Networking and Automation (ICINA)*, volume 1, pages V1–77–V1–81, 2010.

[77] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, 2015.

[78] Iffat A Gheyas and Leslie S Smith. A novel neural network ensemble architecture for time series forecasting. *Neurocomputing*, 74(18):3855–3864, 2011.

[79] Xun Liang, Haisheng Zhang, Jianguo Xiao, and Ying Chen. Improving option price forecasts with neural networks and support vector regressions. *Neurocomputing*, 72(13-15):3055–3065, 2009.

[80] Deepak Gupta, Mahardhika Pratama, Zhenyuan Ma, Jun Li, and Mukesh Prasad. Financial time series forecasting using twin support vector regression. *PLOS ONE*, 14(3):1–27, 03 2019.

[81] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172, mar 2015.

[82] Shom Prasad Das and Sudarsan Padhy. A new hybrid parametric and machine learning model with homogeneity hint for european-style index option pricing. *Neural Computing and Applications*, 28(12):4061–4077, 2017.

[83] Salvatore Carta, Anselmo Ferreira, Diego Reforgiato Recupero, Marco Saia, and Roberto Saia. A combined entropy-based approach for a proactive credit scoring. *Eng. Appl. Artif. Intell.*, 87, 2020.

[84] Mattia Atzeni and Diego Reforgiato Recupero. Multi-domain sentiment analysis with mimicked and polarized word embeddings for human–robot interaction. *Future Generation Computer Systems*, 110:984–999, 2020.

[85] Y. Li and Z. Yang. Application of eos-elm with binary jaya-based feature selection to real-time transient stability assessment using pmu data. *IEEE Access*, 5:23092–23101, 2017.

[86] M. Khalaf, H. Alaskar, A. J. Hussain, T. Baker, Z. Maamar, R. Buyya, P. Liatsis, W. Khan, H. Tawfik, and D. Al-Jumeily. Iot-enabled flood severity prediction via ensemble machine learning models. *IEEE Access*, 8:70375–70386, 2020.

[87] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[88] Anazida Zainal, Mohd Aizaini Maarof, Siti Mariyam Hj. Shamsuddin, and Ajith Abraham. Ensemble of one-class classifiers for network intrusion detection system. In *IAS*, pages 180–185. IEEE Computer Society, 2008.

[89] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 8(4), 2018.

[90] Bangzhu Zhu, Shunxin Ye, Ping Wang, Kaijian He, Tao Zhang, and Yi-Ming Wei. A novel multiscale nonlinear ensemble leaning paradigm for carbon price forecasting. *Energy Economics*, 70:143 – 157, 2018.

[91] Andrea P. Ratto, Simone Merello, Luca Oneto, Yukun Ma, Lorenzo Malandri, and Erik Cambria. Ensemble of technical analysis and machine learning for market trend prediction. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2090–2096, Nov 2018.

[92] Shaolong Sun, Yuying Sun, Shouyang Wang, and Yunjie Wei. Interval decomposition ensemble approach for crude oil price forecasting. *Energy Economics*, 76:274 – 287, 2018.

[93] Kim S. Gan, Kim O. Chin, Patricia Anthony, and Sim V. Chang. Homogeneous ensemble feedforward neural network in cimb stock price forecasting. In *International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pages 1–6, Nov 2018.

[94] Yishan Ding. A novel decompose-ensemble methodology with aic-ann approach for crude oil forecasting. *Energy*, 154:328 – 336, 2018.

[95] Rui Gonçalves, Vitor Miguel Ribeiro, Fernando Lobo Pereira, and Ana Paula Rocha. Deep learning in exchange markets. *Information Economics and Policy*, 47:38 – 51, 2019. The Economics of Artificial Intelligence and Machine Learning.

[96] Sotirios P. Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis, and Nikos Vlachogiannakis. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications*, 112:353 – 371, 2018.

[97] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, March 2017.

[98] Sergio Consoli, Diego Reforgiato Recupero, and Michaela Saisana. *Data science for economics and finance : methodologies and applications*. Springer, Berlin, Germany, 2021.

[99] Ganapathy Vidyamurthy. *Pairs Trading : Quantitative Methods and Analysis*. John Wiley & Sons, 2004.

[100] Cari Kaufman and Duncan Temple Lang. Pairs trading. *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving*, 1(April 2015):241–308, 2015.

[101] Christian L Dunis, Jason Laws, and Ben Evans. Modelling and trading the gasoline crack spread: A non-linear story. *Derivatives Use, Trading & Regulation*, 12(1):126–145, 2006.

[102] Christian L Dunis, Jason Laws, and Ben Evans. Modelling and trading the soybean-oil crush spread with recurrent and higher order networks: A comparative analysis. In *Artificial Higher Order Neural Networks for Economics and Business*, pages 348–366. IGI Global, 2009.

[103] Christian L Dunis, Jason Laws, Peter W Middleton, and Andreas Karathanasopoulos. Trading and hedging the corn/ethanol crush spread using time-varying leverage and nonlinear models. *The European Journal of Finance*, 21(4):352–375, 2015.

[104] Taewook Kim and Ha Young Kim. Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. *Complexity*, 2019:3582516, Nov 2019.

[105] Andrew Pole. *Statistical arbitrage: algorithmic trading insights and techniques*, volume 411. John Wiley & Sons, 2011.

[106] Nikos S Thomaidis, Nick Kondakis, and George D Dounias. An intelligent statistical arbitrage trading system. In *Hellenic Conference on Artificial Intelligence*, pages 596–599. Springer, 2006.

[107] Nicolas Huck. Pairs selection and outranking: An application to the S&P 100 index. *European Journal of Operational Research*, 196(2):819–825, 2009.

[108] Nicolas Huck. Pairs trading and outranking: The multi-step-ahead forecasting case. *European Journal of Operational Research*, 207(3):1702 – 1716, 2010.

[109] Lawrence Takeuchi. Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks. Technical report, Stanford University, 2013.

[110] Matthew Dixon, Diego Klabjan, and Jin Hoon Bang. Classification-based financial markets prediction using deep neural networks. *Algorithmic Finance*, 6(3-4):67–77, 2017.

[111] Christopher Krauss, Xuan Anh Do, and Nicolas Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, 259(2):689–702, 2017.

[112] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[113] Nicolas Huck. Large data sets and machine learning: Applications to statistical arbitrage. *European Journal of Operational Research*, 278(1):330–342, 2019.

[114] J. Knoll, J. Stübinger, and M. Grottke. Exploiting social media with higher-order factorization machines: statistical arbitrage on high-frequency data of the s&p 500. *Quantitative Finance*, 19(4):571–585, 2019.

[115] Matthias Schnaubelt, Thomas G Fischer, and Christopher Krauss. Separating the signal from the noise–financial machine learning for twitter. *Journal of Economic Dynamics and Control*, 114:103895, 2020.

[116] Andrea Flori and Daniele Regoli. Revealing pairs-trading opportunities with long short-term memory networks. *European Journal of Operational Research*, in press, 2021.

[117] Ming-Chi Lee. Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, 36(8):10896–10904, 2009.

[118] Reza Hafezi, Jamal Shahrabi, and Esmaeil Hadavandi. A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. *Applied Soft Computing*, 29:196–210, 2015.

[119] Alexandre Pimenta, Ciniro AL Nametala, Frederico G Guimarães, and Eduardo G Carrano. An automated investing method for stock market based on multiobjective genetic programming. *Computational Economics*, 52(1):125–144, 2018.

[120] Hakan Gunduz. An efficient stock market prediction model using hybrid feature reduction method based on variational autoencoders and recursive feature elimination. *Financial Innovation*, 7(1):28, Apr 2021.

[121] Xin Man and Ernest P. Chan. The best way to select features? comparing mda, lime, and shap. *The Journal of Financial Data Science*, 2020.

[122] Salavtore M. Carta, Sergio Consoli, Lucas Piras, Sebastian Podda, and Diego Reforgiato Recupero. Explainable machine learning exploiting news and domain-specific lexicon for stock market forecasting. *IEEE Access*, 9:30193–30205, 2021.

[123] Mark M. Carhart. On persistence in mutual fund performance. *The Journal of Finance*, 52(1):57–82, 1997.

[124] David Enke and Suraphan Thawornwong. The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, 29(4):927–940, 2005.

[125] Mark T. Leung, Hazem Daouk, and An-Sing Chen. Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of Forecasting*, 16(2):173 – 190, 2000.

[126] Stephen J Brown, William Goetzmann, Roger G Ibbotson, and Stephen A Ross. Survivorship bias in performance studies. *The Review of Financial Studies*, 5(4):553–580, 1992.

[127] A. Lo and A. C. MacKinlay. Data-snooping biases in tests of financial asset pricing models. *The Review of Financial Studies*, 3(3):431–467, 2015.

[128] Dong Lou, Christopher Polk, and Spyros Skouras. A tug of war: Overnight versus intraday expected returns. *Journal of Financial Economics*, 134(1):192 – 213, 2019.

[129] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38(5):5311–5319, 2011.

[130] Christian Leistner, Amir Saffari, Peter M. Roth, and Horst Bischof. On robustness of on-line boosting - a competitive study. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1362–1369, 2009.

[131] Minqi Jiang, Jiapeng Liu, Lu Zhang, and Chunyu Liu. An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms. *Physica A: Statistical Mechanics and its Applications*, 541:122272, 2020.

[132] Jinchang Ren. Ann vs. svm: Which one performs better in classification of mccs in mammogram imaging. *Knowledge-Based Systems*, 26:144–153, 2012.

[133] Halil Bisgin, Tanmay Bera, Hongjian Ding, Howard G. Semey, Leihong Wu, Zhichao Liu, Amy E. Barnes, Darryl A. Langley, Monica Pava-Ripoll, Himansu J. Vyas, Weida Tong, and Joshua Xu. Comparing svm and ann based machine learning methods for species identification of food contaminating beetles. *Scientific Reports*, 8(1):6532, Apr 2018.

[134] Tanvi Sahay, Arpit Aggarwal, Annu Bansal, and Mahesh Chandra. Svm and ann: A comparative evaluation. In *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, pages 960–964, 2015.

[135] Galina Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777 – 781, 1994.

[136] Véronique Genre, Geoff Kenny, Aidan Meyler, and Allan Timmermann. Combining expert forecasts: Can anything beat the simple average? *International Journal of Forecasting*, 29(1):108 – 121, 2013.

[137] Allan Timmermann. Chapter 4 forecast combinations. In G. Elliott, C.W.J. Granger, and A. Timmermann, editors, *Handbook of Economic Forecasting*, volume 1, pages 135 – 196. Elsevier, 2006.

[138] James H Stock and Mark W Watson. Combination forecasts of output growth in a seven-country data set. *Journal of Forecasting*, 23(6):405–430, 2004.

[139] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70 – 83, 2018.

[140] Oliver J. Blaskowitz and Helmut Herwartz. Adaptive Forecasting of the EURIBOR Swap Term Structure. *Journal of Forecasting*, 28(7):575–594, 2009.

[141] Leland E Farmer, Lawrence Schmidt, Allan Timmermann, Frank Diebold, Xavier Gabaix, Bradley Paye, and Hashem Pesaran. Pockets of Predictability. Technical report, SSRN. Available at `https://ssrn.com/abstract=3152386`, 2019.

[142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[143] Athanassia Chalimourda, Bernhard Schölkopf, and Alex J Smola. Experimentally optimal ν in support vector regression for different noise models and parameter settings. *Neural Networks*, 17(1):127 – 141, 2004.

[144] Peter F. Christoffersen and Francis X. Diebold. How relevant is volatility forecasting for financial risk management? *The Review of Economics and Statistics*, 82(1):12–22, 2000.

[145] Johannes Stübinger and Sylvia Endres. Pairs trading with a mean-reverting jump–diffusion model on high-frequency data. *Quantitative Finance*, 18(10):1735–1751, 2018.

[146] Whitney K. Newey and Kenneth D. West. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3):703–708, 1987.

[147] Con Keating William F. Shadwick. A Universal Performance Measure. Technical report, The Finance Development Centre, London. Available at `https://www.actuaries.org.uk/system/files/documents/pdf/keating.pdf`, 2002.

[148] Bin Li and Steven C. H. Hoi. Online portfolio selection: A survey. *ACM Computing Survey*, 46(3), January 2014.

[149] Hossein Rad, Rand Kwong Yew Low, and Robert Faff. The profitability of pairs trading strategies: distance, cointegration and copula methods. *Quantitative Finance*, 16(10):1541–1558, 2016.

[150] N. Baltas, D. Jessop, S. Jones, P. Winter, S. Wu, O. Antrobus, and P. Stoltz. Quantitative monographs: "stock selection using machine learning". In *UBS Global Research*, 2015.

[151] John Kingston. Artificial intelligence and legal liability. In *International conference on innovative techniques and applications of artificial intelligence*, pages 269–279. Springer-Verlang, 11 2016.

[152] Joshua Kroll, Joanna Huey, Solon Barocas, Edward Felten, Joel Reidenberg, David Robinson, and Harlan Yu. Accountable algorithms. *University of Pennsylvania Law Review*, 165:633–705, 02 2017.

[153] Zhenyu Zhao, Radhika Anand, and Mallory Wang. Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 442–452, 2019.

[154] Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.

[155] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):307, Jul 2008.

[156] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):25, Jan 2007.

[157] Mathias Kraus and Stefan Feuerriegel. Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, 104:38–48, 2017.