UNIVERSITY OF LJUBLJANA

SCHOOL OF ECONOMICS AND BUSINESS

MASTER THESIS

ENHANCING THE TECHNICAL ANALYSIS WITH MACHINE AND DEEP LEARNING: AN EXAMPLE OF FORECASTING THE DIRECTIONAL CHANGE OF THE S&P 500 INDEX

Ljubljana, May 2021 ENRICO SCHINCARIOL

AUTHORSHIP STATEMENT

The undersigned Enrico Schincariol, a student at the University of Ljubljana, School of Economics and Business, (hereafter: SEB LU), author of this written final work of studies with the title Enhancing the Technical Analysis with Machine and Deep Learning: An Example of Forecasting the Directional Change of the S&P 500 Index, prepared under supervision of as. prof. dr. Igor Lončarski

D E C L A R E

1. this written final work of studies to be based on the results of my own research;
2. the printed form of this written final work of studies to be identical to its electronic form;
3. the text of this written final work of studies to be language-edited and technically in adherence with the SEB LU's Technical Guidelines for Written Works, which means that I cited and / or quoted works and opinions of other authors in this written final work of studies in accordance with the SEB LU's Technical Guidelines for Written Works;
4. to be aware of the fact that plagiarism (in written or graphical form) is a criminal offence and can be prosecuted in accordance with the Criminal Code of the Republic of Slovenia;
5. to be aware of the consequences a proven plagiarism charge based on the this written final work could have for my status at the SEB LU in accordance with the relevant SEB LU Rules;
6. to have obtained all the necessary permits to use the data and works of other authors which are (in written or graphical form) referred to in this written final work of studies and to have clearly marked them;
7. to have acted in accordance with ethical principles during the preparation of this written final work of studies and to have, where necessary, obtained permission of the Ethics Committee;
8. my consent to use the electronic form of this written final work of studies for the detection of content similarity with other written works, using similarity detection software that is connected with the SEB LU Study Information System;
9. to transfer to the University of Ljubljana free of charge, non-exclusively, geographically and time-wise unlimited the right of saving this written final work of studies in the electronic form, the right of its reproduction, as well as the right of making this written final work of studies available to the public on the World Wide Web via the Repository of the University of Ljubljana;
10. my consent to publication of my personal data that are included in this written final work of studies and in this declaration, when this written final work of studies is published.

Ljubljana, May 17th, 2021 Author's signature: Enrico Schincariol

TABLE OF CONTENTS

INTRODUCTION **2**
1 LITERATURE REVIEW **4**
2 METHODOLOGY **5**
2.1 Introduction 5
2.2 Data retrieval 6
2.3 Target variable 7
2.4 Explanatory Variables (Features) 7
2.4.1 Exponential Moving Average 8
2.4.2 Relative Strength Index 9
2.4.3 Chaikin Volatility Measure 11
2.4.4 Chaikin Accumulation and Distribution Oscillator 11
2.5 Explanatory Variables (Feature) analysis 12
2.5.1 Financial analysis of technical indicators 12
2.5.2 Statistical analysis of technical indicators 14
2.6 Learning algorithms 26
2.6.1 The problem of overfitting 27
2.6.3 Linear models 28
2.6.4 Non-linear models 35
2.7 Methods for evaluating classification models 43
2.7.1 Evaluating class predictions 43
2.7.1 Evaluating class probabilities 46
3 EVALUATING FORECASTING PERFORMANCE **48**
3.1 Evaluating the performance of training models 48
3.2 Evaluating the performance of testing models 50
3.3 Evaluating the performance of the chosen model on sequential data 51
4 TRADING STRATEGY **51**
REFERENCES **54**
APPENDICES **61**

LIST OF FIGURES

Figure 1: Distribution of Observations for the Set of Features 18
Figure 2: Q-Q Plots for the Set of Features 19
Figure 3: Distribution of Observations for the Set of Features After the Transformations 22
Figure 4: Q-Q Plots for the Set of Features After the Transformations 23
Figure 5: Correlogram of Features and Outcome 25
Figure 6: Example of a Support Vector Machine Algorithm with a Linear Classifier 33
Figure 7: Example of a K-Nearest Neighbors Algorithm for Binary Classification 36
Figure 8: Example of a Multi-Layer-Perceptron Algorithm 38
Figure 9: Example of a Random Forest Algorithm for Binary Classification 41
Figure 10: Example of a Confusion Matrix for Binary Classification 43
Figure 11: Example of a ROC Space, Curves and AUC's for Binary Classification 47
Figure 12: Changes in the AUC's as a Change in the Value of Tuning Parameters 49
Figure 13: Relative Returns for Both Strategies Compared 52

LIST OF TABLES

Table 1: Descriptive Statistics of the Feature Set for the Studied Period 13
Table 2: Results of Shapiro-Wilk's Test for Normality on the Set of Features 17
Table 3: Correlation Coefficients P-Values 26
Table 4: Class Predictions and Probabilities Metrics for the Best Training Models 49
Table 5: Class Predictions and Probabilities Metrics for the Best Testing Models 50
Table 6: Class Predictions Metrics for the MLP Model for the Selected Period 51

LIST OF ABBREVATIONS

sl. – Slovene

ACC – (sl. Natančnost); Accuracy

AUC – (sl. Površina pod krivuljo); Area Under the Curve

CAD – (sl. Chaikinova akumulacija-distribucija); Chaikin Accumulation Distribution

CI – (sl. Interval zaupanja); Confidence Interval

DRC – (sl. Sprememba smeri gibanja); Directional Change

EMA – (sl. Eksponencialno drseče povprečje); Exponential Moving Average

ETF – (sl. Borzni sklad); Exchange Traded Fund

KNN – (sl. Metoda K-najbljižjih sosedov); K-Nearest Neighbours

LOG – (sl. Logistična regresija); Logistic Regression

MLP – (sl. Večslojni perceptron); Multi-Layer-Perceptrons

NIR – (sl. Stopnja ne-informativnosti); No-Information Rate

RF – (sl. Metoda naključnega gozda); Random Forest

ROC – (sl. Operativna značilnost prejemnika); Receiver Operating Characteristic

RSI – (sl. Indeks relativne moči); Relative Strength Index

SVM – (sl. Metoda podpornih vektorjev); Support Vector Machine

VOL – (sl. Volatilnost); Volatility

INTRODUCTION

Accurately forecast asset price movements is a major challenge confronting the majority of the economic classes, for instance, speculators, investors, and businesses. In their quest to forecast the markets, they assume that future occurrences are based at least in part on present and past events and data. However, financial time series are inherently noisy, non-stationary and deterministically chaotic, and therefore among the most difficult signals to forecast. These types of characteristics suggest that there is no complete information that could be obtained from the past behaviour of financial markets to fully capture the dependency between the future and the past price. This has led many economists to adopt the efficient market hypothesis, which states that price changes are independent of the past and follow a random walk. Malkiel (1999) provides the following definition of market efficiency.

A capital market is said to be efficient if it fully and correctly reflects all relevant information in determining security prices. Formally, the market is said to be efficient with respect to some information set, ϕ_t, if security prices would be unaffected by revealing that information to all participants. Moreover, efficiency with respect to an information set, ϕ_t, implies that it is impossible to make economic profits by trading on the basis of ϕ_t.

According to this hypothesis, price changes are therefore unpredictable with consistency, especially in the long run. Any change in price represents the immediate reaction to an instantaneous news event or new and unexpected changes in supply and demand forces. If any expected profit opportunity appears, then investors would immediately exploit the opportunity in a way that drives the price back to the level where any trading or investment strategy is not profitable anymore. From Malkiel's definition of an efficient capital market, we can retrieve three important concepts, specifically the importance of the information set, the ability to use this information set in a trading strategy and that the main proxy to test the efficient market hypothesis is economic profits. However, the concept of how the information variables in the information set are used to produce the actual forecasts is not present in Malkiel's definition. Therefore, we introduce an alternative, extended, definition of an efficient capital market proposed by Timmermann and Granger (2004).

A market is efficient with respect to the information set, ϕ_t, search technologies, S_t, and forecasting models, M_t, if it is impossible to make economic profits by trading on the basis of signals produced from a forecasting model in M_t defined over predictor variables in the information set ϕ_t and selected using a search technology in S_t.

Based on the definition above we can better investigate the role and importance of several statistical modelling frameworks, which include search technologies and forecasting models aimed at predicting a financial asset's evolution through time.

Although there has been a lot of debate about the efficient market hypothesis, it is hard either to prove or disprove it. Numerous researchers underline the fact that the efficient market hypothesis is difficult to test and that the methodologies used by its supporters are flawed and biased. Additionally, the proven existence of price trends in financial markets and the undiscounted serial correlations among fundamental events and economic figures affecting the markets, are two of many pieces of evidences against the efficient market hypothesis (Ţiţan, 2015). These findings introduce the possibility to continue and further develop the research related to evaluating the predictive power, consistency and practical importance of forecasting methods.

In contrast with Malkiel's statements about the weak use of forecasting asset prices using popular informative techniques, we assume there exists a set of information, models and approaches able to accurately and consistently predict the evolution of an asset price. Additionally, we assume that the latter set can be defined by a collection of the information generated by a well-known informative field called technical analysis. The aim of our thesis is not an attempt to disprove the works of the supporters of the random walk theory and the efficient market hypothesis, but rather to illustrate the possible use of technical analysis in a predictive modelling and trading context. The goal is then to explore the predictive power of forecasting approaches and models from the artificial intelligence branches of machine and deep learning. Briefly, using a set of inputs made of financial technical indicators we are training and testing several learning models to predict the daily directional change of the log-relative-return, abbreviated as $\log R_i$, on the closing price of the S&P 500 Index value. More about the characteristics and differences of models and technical indicators in Chapter 2. We are effectively tackling the forecasting task by reshaping it into a time-series classification problem. The goal is still to predict an event, yet it can be interpreted as a class rather than as a continuous value. Time-series classification consists of constructing algorithms dedicated to automatically label time-series data. These problems are differentiated from traditional classification problems because the attributes are ordered (Fawaz, Forestier, Weber, Idoumghar & Muller, 2019). At the end of our work, we want to test a trading strategy based on our daily predictions made for the period 01.01.2021-30.04.2021 against a buy-and-hold strategy, to possibly highlight some practical use of technical analysis and to discover further potential fallacies in the Efficient Market Hypothesis theory or to provide additional evidence for its support.

In the following Chapter 1 we are going to present several relevant works of literature concerned with forecasting financial asset prices, with a primary focus on market indexes, using statistical modelling methods incorporating multivariate information, especially financial technical indicators. Chapter 2 presents the methodology used to produce our work's results. In this chapter, the reader gets acquainted with the definition of the forecasting problem, the data retrieval and preparation, the algorithms used to generate the predictions and the performance evaluation methods. Chapter 3 follows with the performance evaluation of our forecasting models in the forecasting context, while Chapter 4 presents and evaluates the performance of a trading strategy based on our predictions, against a passive strategy such as the buy-and-hold one. Chapter 5 presents the conclusion of this thesis and our recommendations for further improvements.

1 LITERATURE REVIEW

One of the most extensively researched field in academic literature related to finance is concerned with forecasting asset prices. Although there are several sub-topics of this general forecasting problem, including stock, index, forex, commodity, bond price, as well as volatility and, recently, cryptocurrency price forecasting, the same underlying dynamics of the predictive modelling processes can be applied to all types of these topics. We could separate the existing academic works into two domains, namely research concerned with value prediction, such as, for example, predicting the future stock price value, and research related to predicting the directional changes of a financial asset characteristic, for example, the directional change of the return on a stock or an index. The first domain attempts to solve a chosen forecasting problem by using one or more predictive modelling techniques based on regression analysis to produce a continuous output, while the second domain focuses on using similar or even the very same modelling techniques to correctly identify the directional movement of an asset's value by reformulating the initial forecasting problem as a classification problem. (Krollner, Vanstone & Finnie, 2010). Our centre of attention, as stated in the introduction, is on the second domain. There are different approaches to both types of forecasting problems. While some studies propose to use models built exclusively on an asset's publicly available data attributes, such as open, close, high, low and volume, (Fischer & Krauss, 2018; Jiao, Yang, Jakubowicz & Jeremie, 2017; Zhong & Enke, 2019), others propose the inclusion of additional data such as, for example, technical indicators, fundamental analysis ratios, data from correlated global indexes, social media feeds and news from the media (Yıldırım, Toroslu & Fiore, 2021; Cao & Tay, 2003). We are particularly interested in the works where additional data inputs are given by a set of financial technical indicators and are then fed to a different machine and deep learning algorithms that aim at finding an acceptable solution to the chosen asset's directional change forecasting problem with supervised classification learning.

In the insightful article by Sezer, Gudulek and Ozbayoglu (2020), we conclude that the majority of the proposed solutions by academics to the aforementioned problem are generated with supervised deep learning models. Popular among the latter are convolutional and recurrent neural networks, long-short term memory networks and deep neural networks. These types of algorithms can achieve outstanding performance in a multivariate dataset context compared to other types of learning algorithms and less recent classical statistical models. Most of the works surveyed appear to have similar characteristics in terms of the chosen feature set, where the predominance is given to the mentioned combination of the OCHLV set and technical indicators. In addition to that, common choices for the prediction horizons are daily and weekly forecasts. Similar, if not identical, characteristics can be found also in the literature related to non-neural learning models. Popular choices are support vector machines, generalized linear models and K-nearest neighbours algorithms (Chen & Hao 2017; Chi-Jie, Tian-Shyug & Chih-Chou, 2009). Alternatively, the predictive power of decision trees algorithms in a financial classification problem is also being explored in the field of research, for example by Chang, Fan & Lin (2011). Based on the number of works and prominent results, we are encouraged to explore the predictive power of different classes of learning models tailored to our classification problem. We expect to produce statistical models able to correctly predict the daily directional changes of the S&P500 Index by using technical indicators as inputs, to achieve an overall satisfactory performance and to further support the use of technical analysis in financial predictive modelling contexts by developing a trading strategy that, according to the supporters of market efficiency, should not be consistently profitable.

2 METHODOLOGY

2.1 Introduction

Recall, the goal of this work is to use a collection of statistical procedures and models from the fields of machine and deep learning to generate daily predictions based on a set of technical indicators for the Standard and Poor's 500 Index daily directional change. As illustrated in Section 3.3, we are dealing with a classification problem. The goal in classification problems is to take an input vector, say X, and assign it to one of the discrete classes C_k, where $k = 1, \dots, K$. In our case, the classification problem consists of separating two classes, the upward and downward directional changes. Ideally, these classes are taken to be disjoint, so that each input is assigned correctly to one and only one class for every single data point. The input space is divided into decision regions, whose boundaries are called decision surfaces or decision boundaries. In this chapter, we present a set of statistical modelling processes widely used in the field of data analysis and science. We start by presenting what kind of data we are retrieving and how, with the help of the latter, can we formulate the problem as a dependent variable (the directional change), that takes any of the two different values (classes) depending on the values from the set of features (technical indicators). Next, we perform statistical analysis on the set of features to assess their characteristics, their possible interdependence and how could all this impact the results of our forecasting models. We also study the possibility of utilizing mathematical transformations on the latter set to avoid common issues related to the chosen learning models, such as over and underfitting of data. Once we finalize the dataset, we turn our focus on separating the original dataset into two parts. The first part is reserved for the training of our models, where the values of the target variable are known, while the second part of the dataset is given to the testing of our models, where the outcomes are not known to the models, which should apply what they have learned from the training phase in an out-of-sample context. Additionally, we introduce the concept and methods of resampling and why it can come in handy in better tackling our problem. We then move to the presentations and mathematical definitions of the chosen learning models. Finally, the chapter is concluded with the evaluation of the forecasting models and the performance of a trading strategy based on our predictions.

Computations are conducted using R, an open-source programming language designed by Robert Gentleman, the Executive Director of the Center for Computational Biomedicine at Harvard Medical School, and Ross Ihaka, former Associate Professor of Statistics at the University of Auckland, New Zealand (Ihaka & Gentleman, 1996). The integrated development environment used is RStudio, supported by the R Foundation for Statistical Computing. R offers several advantages, such as exemplary support for data wrangling, statistical operations, including machine and deep learning modelling, high compatibility and powerful tools for exploratory data analysis and visualization (Irizarry, 2019). We make extensive use of the following libraries: *tidyverse, tidyquant, caret, psych* and *pROC*.

Note, several words in this work can be used interchangeably. These include features and predictors, target variable and outcome, technical indicators and indicators, and most importantly classification and prediction.

2.2 Data retrieval

The first step is to retrieve data that enables us to tackle the forecasting, or, as introduced, the classification problem. As we are building models around technical indicators, we require such data, that provides us with a base upon which we calculate the mentioned indicators. Note, the definition of the outcome variable is given in Section 3.3, while the definitions of the added features follow in the next Section 3.4. This section is meant to create a simple visualization of what kind of dataset are we building.

The starting dataset used for modelling the solution is constructed on publicly available data retrieved from the *tidyquant* API service. We retrieved a total of 9074 daily observations for seven of the S&P 500 Index attributes, $Date$, $Open$, $High$, Low, $Close$, $AdjClose$ and $Volume$, for the period starting at 01.01.1985 to 31.12.2020. Due to the incompleteness of the dataset, we are unable to retrieve older daily observations. The dataset is subsequently extended with the addition of an array consisting of the daily returns on the S&P500 closing price and new data attributes, also referred to as features, computed based on the log-returns of the originally retrieved ones. The features represent the set of the chosen technical indicators, namely the exponential moving average, the relative strength index, the Chaikin volatility measure and the Chaikin accumulation distribution oscillator. In order of appearance, the latter is going to be abbreviated as EMA, RSI, VOL and CAD. These are calculated at different times, each with a different time horizon. More about it in Section 3.4. Finally, we add the array containing the two possible values for the daily directional change of the log-relative-return on the index's closing price and we trim the first non-numeric entries caused by the nature of the technical indicators' value generative process. We now have the complete dataset for generating daily forecasts at our disposal, consisting of original attributes, 12 new added features and the target variable. Note, the dataset is furtherly expanded for testing the trading strategy.

2.3 Target variable

We now move on with the mathematical definitions of our target variable and features. The target variable, which is defined as the upward or downward movement of the S&P 500's daily log-relative-return on the closing price, abbreviated as DRC, is a categorical variable that is set to take only two possible values when defined as:

$$DRC_t = \begin{cases} 1 & \text{if } \log R_{C_t} > \log R_{C_{t-1}} \\ -1 & \text{else.} \end{cases} \qquad (1)$$

The two possible outcomes are our classes, which our models are going to predict based on the feature set of technical indicators. There is a total of 4886 upward and a total of 4137 downward directional changes for the log-relative-return on the closing price in the studied period after accounting for non-numerical entries produced by technical indicators, due to the specific time frames.

2.4 Explanatory Variables (Features)

As introduced, we are going to use a set of technical indicators as the inputs to our learning models. Technical indicators, in a financial context, are mathematical calculations based on historic information of price, volume and/or open interest of a security, or a financial market used to forecast the chosen asset's market direction. Several advantages are linked to the use of the mentioned tools. Above all, technical indicators provide a quantified framework for processing data into organized information about actual observed market behaviour of supply and demand forces. Calculations can be formulated for all three types of directional movement, namely upwards, downwards and sideways movements, and for any kind of market trading securities, such as stocks, features, commodities and currencies. In addition to that, technical indicators are flexible quantitative tools that can be parametrized for any time frame adapting to dominant major trends (yearly), intermediate-term movements (weekly to monthly), day-to-day minor trends and even momentary high-frequency fluctuations (Colby, 2002). Furthermore, these tools can be adapted to easily incorporate up-to-date mathematical and statistical models, especially for risk control, and can be used alongside other types of information generating processes. Thus, we could argue that such tools can provide significant and fast quantitative information, are easy to compute and have a high degree of flexibility and adaptivity (Neely, Rapach, Tu & Zhou, 2014).

The set of all technical indicators could be divided into two complementary classes, namely oscillators and overlays. Oscillators are a special class of technical indicators that oscillates between a local minimum and maximum and focuses on market momentum. They are best used to provide readings of overbought and oversold price movements. Traders and investors define price turns and reversals within ranging markets using oscillators because they swing within a generally defined range. Overlays are special types of technical indicators used by traders and investors to identify overbought and oversold levels. They provide insight into the supply and demand of a stock. A well-known type of overlays indicators is moving averages (Silva, Neves & Horta, 2015). In an attempt to capture a broader informative picture out of our dataset, we decide to use both types of classes, specifically, we choose to use a series of exponential moving averages, relative strength indexes, Chaikin volatility measures and Chaikin accumulation and distribution oscillators. Below we provide the mathematical definitions for each indicator. Section 3.5 follows with the statistical analysis of the newly generated series of values and the general interpretations of the latter for the studied period.

2.4.1 Exponential Moving Average

The exponential moving average is a widely used technical indicator able to generate quantitative information from a time series. It is a type of moving average that applies weighting factors in an exponentially decreasing manner so that the weighting for each data point decreases exponentially towards zero as the data point is older in time. In mathematical terms, it is a first-order infinite impulse response filter. In a general financial context, an exponential moving average would be calculated on the closing prices (James, 1968). In our calculations, we opt to calculate this indicator by taking the log-relative-return on the opening price at time t instead. This choice is constrained by the nature of the problem and the process of generating an answer. Suppose we want to predict at time t. If any of the predictors used is calculated by including the closing price at time t then we would nullify any level of logic and ultimately provide wrongful answers based on illogical procedures.

The exponential moving average at time t can be calculated recursively by the following difference equation (Klinker, 2010):

$$EMA_t = \alpha \log R_{O_t} + (1 - \alpha) EMA_{t-1} \tag{2}$$

where α is a constant smoothing factor between 0 and 1 representing the degree of weighting decrease, defined as:

$$\alpha = \frac{2}{(n + 1)} \tag{3}$$

where n is the number of chosen periods to average over, $\log R_{O_t}$ the log-relative-return on the opening price of the S&P 500 Index at time t and EMA_{t-1} is the previous output. We are free to choose the set of values over which we calculate this technical indicator. There is no analytical solution to the problem of choosing the optimal number of periods to calculate the average. Choices may vary depending on the type of prediction problem, its time frame and the choice of the predictors' set composition. For the latter reasons, we opt to choose a set of time frames in line with existing academic research (Shynkevich, McGinnity, Coleman, Belatreche & Li, 2017). The chosen set of time frames for our EMA predictors is then $n \in \{5,8,15,20\}$.

2.4.2 Relative Strength Index

The Relative Strength Index is a price-following oscillator with a range of values between 0 and 100 firstly introduced by Welles Wilder (1978), which measures the velocity and magnitude of price movements. The indicator can be interpreted in the context of tops and bottoms. Wilder states that when a price moves up at a rapid speed it can be considered overbought and oversold when it rapidly declines. A top is achieved when the RSI reaches a value above the threshold of 70, signalling for a condition of overbuying, while a bottom happens when the RSI's value drops below the threshold of 30, indicating a condition of overselling security. A value of 50 indicates a sign of no trend. Another way to interpret it is to look for a divergence in which the security is either make a new high, yet the RSI reaches a lower high or a new low, yet the RSI reaches a higher low. Such divergences are called bearish and bullish divergences, respectively. Furthermore, an additional interpretation tries to determine and confirm an upward or downward trend. Upward trends are trading contexts where the RSI takes on values in the range of 40 to 80, while downward trends are usually happening when the indicator's values are in the range of 20 to 60. Moreover, a bearish divergence occurs exclusively in upward trends and it often leads only to a short correction of the trend rather than a full reversal (Hill, 2019). The RSI is defined as (Wilder, 1978):

$$RSI_t = 100 - \frac{100}{(1 + RS_t)} \tag{4}$$

where the relative strength factor, denoted by RS, is a ratio of two exponentially smoothed averages with a different decaying factor (Wilder 1978). We will denote the latter as α_{RS} and compute it as:

$$\alpha_{RS} = \frac{1}{n} \tag{5}$$

The relative strength factor is then formulated as:

$$RS_t = \frac{EMA_t(U_t, n)}{EMA_t(D_t, n)} \tag{6}$$

where:

$$\begin{gathered} U_t = O_t - O_{t-1} \\ D_t = O_{t-1} - O_t \end{gathered} \tag{7}$$

For the logical constraints explained in the previous sub-section, we substitute the closing price with the log-relative-return of the opening prices. Wilder (1978) argues to use a 14-day RSI, yet widely used are also 9-day and 25-day periods (Shynkevich, McGinnity, Coleman, Belatreche & Li, 2017). We choose the set of values of $n \in \{6,10,14,20\}$, respectively, as it produces better results than other sets of values under several metrics used for the evaluation of the models' predictive power.

2.4.3 Chaikin Volatility Measure

As volatility measures the risk of an investment in any financial security its estimation has significant practical implications in constructing and managing an optimal portfolio. The classical method of measuring volatility is computed as the standard deviation of prices or returns within a fixed time frame, for example, 5 months. An implicit assumption is to assume that volatility is constant within the selected time frame, which might appear unrealistic, especially in shorter frames. Part of the literature opus regarding the topic focuses on estimating the historical volatility of a security from its trading range with the works of Garman and Klass (1980) and Rogers, Satchell and Yoon (1991. Such estimators are also known as range estimators as they rely upon information on the daily trading range. A widely used dispersion measure in the industry is the Chaikin volatility measure retrieved by computing the difference between two moving averages of a volume-weighted accumulation-distribution line. By comparing the spread between a security's high and low prices, it quantifies volatility as a widening of the range between the high and the low price. A viable interpretation of its values derives from the assumption that, when the market reaches a top, increased volatility follows due to investors indecisive behaviour. Conversely, a decrease in volatility happens when a market drops to a bottom. Moreover, a decrease in volatility over a longer time horizon suggests an incoming market top (Di Lorenzo, 2012). As we are working with log-relative-returns, we formulate this indicator as:

$$VOL_t = \frac{EMA(\log R_H)_n - EMA(\log R_L)_m}{EMA(\log R_L)_m} \cdot 100 \tag{7}$$

where n is the number of chosen periods to average over for the first exponential moving average and m represent the number of chosen periods for the second exponential moving average. Typical values for both parameters are ten and twenty days, yet again we choose to try also different values for both parameters, namely we choose $n \in \{10,15,20\}$ and $m \in \{20,30,40\}$. Note that the log-relative returns of high and low prices taken into consideration are those before time t. The indicator can take values in the range of [-100, 100]. Values closer to 100 indicate a period of high volatility, while values closer to -100 the contrary.

2.4.4 Chaikin Accumulation and Distribution Oscillator

Lastly, we make use of the Chaikin accumulation and distribution oscillator as the only technical indicator based on the volume attribute. The calculation itself can be considered as an indicator of an indicator, as it measures the momentum of the accumulation-distribution line technical indicator to anticipate directional changes in the latter line by measuring the momentum of the line’s movements. Mathematically, it is computed as the difference between a three-day and a ten-day exponential moving average of the accumulation distribution line and the range of values is both positive and negative. A move into the positive domain suggests that the accumulation-distribution line is rising and buying pressure prevails. A move into the negative domain, on the other hand, indicates that the accumulation-distribution line is falling and selling pressure prevails (Di Lorenzo, 2012). Again, we base our calculations on the log-relative-returns of previous day log-relative-returns of high and low prices together with the opening price at time t and lastly the previous day log-relative-return on volume. We now move on to the analysis of the technical indicators’ results for the whole period and subsequently, we study the dependence between the set of features and the outcome, and ultimately the relationship between the features themselves.

2.5 Explanatory Variables (Feature) analysis

2.5.1 Financial analysis of technical indicators

Now that our target variable and features are defined, we want to retrieve useful financial and statistical information from our constructed dataset. At first, we are interested in an informative financial outlook on the results generated by the set of indicators. For this reason, below we provide a table (1) with descriptive statistics calculated on this set followed by the interpretation of the results for the whole period.

Looking at the set of exponential moving averages, we can argue that the S&P500 Index is slowly growing in the studied period based on the mean values for all four indicators. Additionally, by considering their standard deviations, we can observe that the spread of the values for these indicators is decreasing when the time frame chosen is increasing. The latter could be supported by including the results for the min, max and range values. Although further analysis would be required to firmly state the following, the results for the exponential moving average set of indicators with longer time frames could suggest that a strategy based on them would be characterized by steady growth with lower volatility compared to a strategy based on shorter time horizons. The relative strength index set of features, having a mean value of roughly 50 and a low standard deviation, can indicate that on average trends tend to expire with longer periods. Moreover, if we focus on their standard deviations and the min-max values, or the range itself, we understand that, again, shorter periods are characterized by increased volatility, thus supporting the previous analysis. Conversely, the Chaikin volatility measures produce peculiar results that appear contrary to the abovementioned. We can observe that as the time frame increases, the measures of central tendency and dispersion increase as well. Finally, the Chaikin accumulation-distribution oscillator shows that on average a buying pressure is prevalent over a selling pressure, with a positive range value furtherly supporting the previous claim.

The results of this analysis provide us with insights into the financial context of our work and a basic statistical overview. We are, therefore, not yet ready to feed the data to our algorithms, as we want to furtherly investigate the statistical behaviours and relationships among the set of features and the outcome.

Table 1: Descriptive Statistics of the Feature Set for the Studied Period

Features	Mean	Sd	Median	Trimmed	Mad	Min	Max	Range	Skew	Se
EMA_05	0.0095	0.0081	0.0081	0.0091	0.0069	- 0.0752	0.0387	0.1139	0.2405	0.0001
EMA_08	0.0095	0.0075	0.0077	0.0089	0.0061	- 0.0478	0.0350	0.0828	0.5600	0.0001
EMA_15	0.0095	0.0071	0.0073	0.0088	0.0053	- 0.0214	0.0333	0.0547	0.8691	0.0001
EMA_20	0.0095	0.0069	0.0071	0.0087	0.0049	- 0.0158	0.0326	0.0484	0.9681	0.0001
RSI_06	49.6996	7.7849	49.7862	49.7656	7.7394	10.5768	78.6996	68.1228	- 0.1348	0.0819
RSI_10	49.8413	4.6637	49.9250	49.9020	4.3669	17.0230	67.8250	50.8021	- 0.2839	0.0491
RSI_14	49.8942	3.3353	49.9675	49.9451	3.0111	21.1349	63.8207	42.6858	- 0.3938	0.0351
RSI_20	49.9303	2.3433	50.0018	49.9710	2.0682	25.3528	60.3790	35.0262	- 0.5197	0.0247
VOL_10	0.0400	0.3374	- 0.0170	- 0.0011	0.2292	- 0.5403	4.6654	5.2057	3.5886	0.0035
VOL_15	0.0411	0.3471	- 0.0203	- 0.0028	0.2198	- 0.5358	4.6245	5.1603	3.6991	0.0037
VOL_20	0.0433	0.3664	- 0.0206	- 0.0048	0.2240	- 0.5473	6.0290	6.5763	4.5205	0.0039
CAD	132,458,986.0905	83,056,159.2634	121,650,297.4541	128,835,806.7110	98,734,891.0328	984,625.4032	301,666,682.7060	300,682,057.3028	0.2659	873,888.1276

Source: Own work.

2.5.2 Statistical analysis of technical indicators

A recurring challenge in model building is the process of understanding how the available data behaves. We are interested in discovering whether data follows certain objects, what characterizes these objects and whether these are sufficient to give us a general understanding of the data we are working with. In statistical analysis, such objects are known as distributions. Distributions are of key importance in statistical modelling, as they help us understand how data behaves, which assumptions are we going to make and why, and how can we use several measures, as well as how to test their significance.

2.5.2.1 Log-normality assumption

Throughout our work, we frequently use the term log-relative-returns. It is common practice, when working with financial data, to not use raw prices as inputs to our forecasting models, but instead, we use log-returns or log-relative-returns depending on the type of problems we are undertaking. In our case, we use log-relative-returns, by assuming that the various prices of the S&P500 Index are log-normally distributed. Let us provide the definition of a log-normal distribution first and then enumerate the advantages of working with such data.

Let X be a continuous random variable representing any price of an asset, be that the open, high, low, close, adjusted or the volume. If X is following a log-normal distribution, then $Y = \ln(X)$ is normally distributed. Likewise, if Y follows a normal distribution, then its exponential, $X = \exp(Y)$, is log-normally distributed. The probability density function of X is then defined as:

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} exp[-\frac{(\ln(x)-\mu)^2}{2\sigma^2}] \tag{8}$$

The parameters of this distribution are the expected value (mean), denoted by μ and the standard deviation σ, which are not the two moments of the variable X itself, but the variable's natural logarithm. Then, as we are working with relative returns, we define Y to be:

$$Y = \ln(\frac{X_t - X_{t-1}}{X_t}) \tag{9}$$

which can be rewritten as:

$$Y = \ln(X_t - X_{t-1}) - \ln(X_t) \tag{10}$$

There are several advantages linked to the use of the log-normal assumption and the related distributions. First, log-relative returns would be normally distributed. The normal distribution has a set of properties, or attributes, which comes very handy in statistical analysis and model building. An important attribute comes from information theory, which states that when we define a data set with only the mean and the variance, the one distribution, which allows us to continue work probabilistically while making minimal assumptions about the data, is, precisely, the normal distribution. This is achieved through the distribution's property of maximizing its entropy, which is the expected value of the negative logarithms of data points' probabilities. Additionally, the normal distribution is characterized by the symmetry property, which helps us avoid the problem of model bias. Model bias, or bias error, is an error occurring due to erroneous assumptions in the learning algorithm, which can lead to an underestimation of the relevant relations between predictors and outcomes (Kiseon & Shevlyakov, 2008). Next, we want to test whether the assumption about log-normality of prices and the normality of our features holds and what can be mathematically done to approach this ideal scenario.

2.5.2.2 Testing the log-normality and normality assumptions

There are several ways to explore how a continuous random variable might be distributed, ranging from statistical tests to exploratory data visualization. We are not required to test both types of distribution, but only whether our feature set follows a normal distribution. If so, then we can claim that prices do follow a log-normal distribution (Mishra et al., 2019). We choose to assess the normality distribution for each predictor with the Shapiro-Wilk test, and two exploratory visualization methods, namely histograms and quantile-quantile plots, abbreviated as Q-Q. The goal is to understand better how the data is distributed for each predictor and ultimately, if necessary, choose an appropriate transformation to reshape these distributions so we can work with normally distributed data or, at least, data that is behaving with a higher degree of normality. Let us begin by presenting the abovementioned statistical test.

2.5.2.2.1 Shapiro-Wilk test for normality of features

The Shapiro-Wilk test is being widely used in statistical analysis mainly due to its highest degree of power for a given significance among other similar methods, such as the Anderson-

Darling, the Kolmogorov-Smirnov and the Lilliefors tests. The null hypothesis of this test states that a sample comes from a normally distributed population and the test statistic is computed as:

$$W = \frac{(\sum_{i=1}^{n} a_i \, x_{(i)})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{11}$$

where $x_{(i)}$ is the ith order statistic and $\bar{x}$ is the sample mean, while the coefficients a_i are computed as:

$$(a_1, \dots, a_n) = \frac{m^T V^{-1}}{C} \tag{12}$$

by the vector $m = (m_1, \dots, m_n)^T$, which includes the expected values of the order statistics of independent and identically distributed random variables drawn from the standard normal distribution, the covariance matrix of the normal order statistics denoted by V and the vector norm $C = ||V^{-1}m||$. To test the null hypothesis we choose an alpha level $= 0.05$. The only disadvantage of this test is that we are limited by the size of the sample. The authors argue, that as data samples become larger, the probability of rejecting the null hypothesis increases (Nornadiah & Bee, 2011). For the latter reason, we take a simple random sample of 5000 entries for each feature and assume this sample is representative of the whole dataset. The results are reported in the table below. From the obtained p-values we are suggested to reject the null hypothesis, yet, as previously mentioned, we want to investigate further with exploratory data analysis methods.

Table 2: Results of Shapiro-Wilk's Test for Normality on the Set of Features

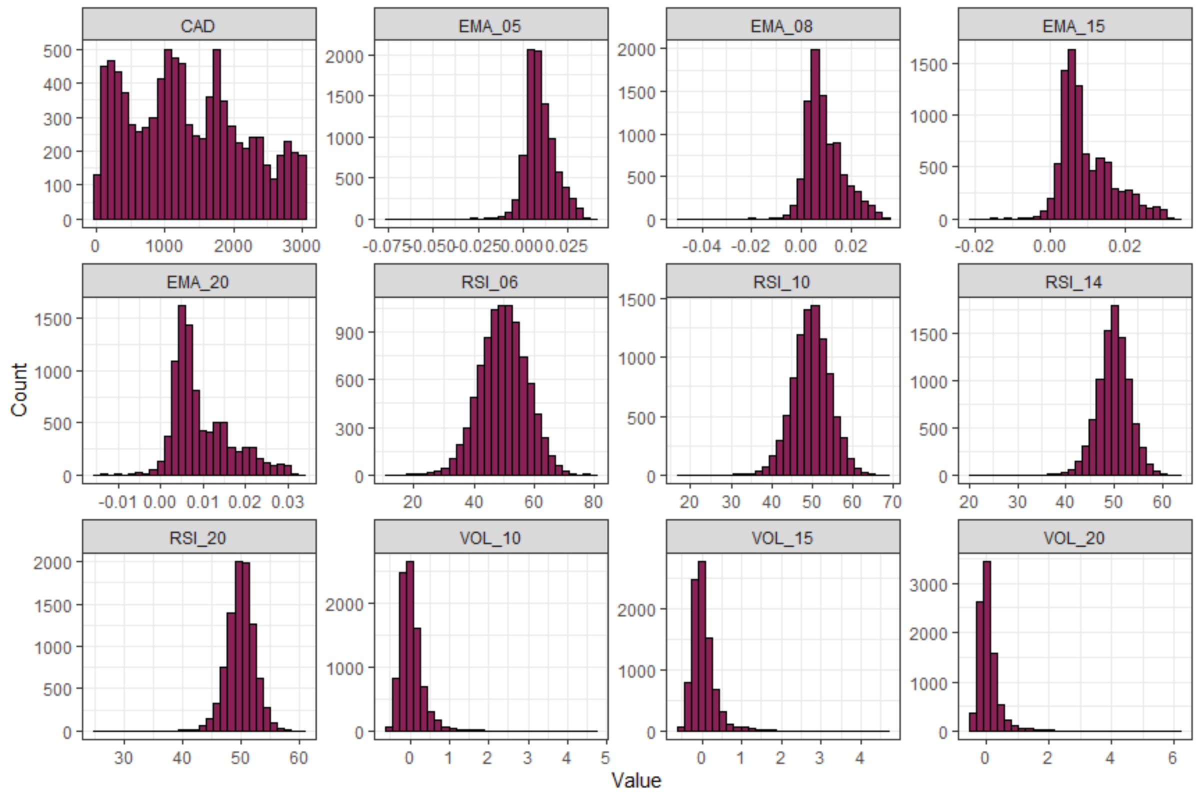
Feature	W	P-value
EMA_05	0.9608	0.00000
EMA_08	0.9518	0.00000
EMA_15	0.9278	0.00000
EMA_20	0.9146	0.00000
RSI_06	0.9980	0.00001
RSI_10	0.9908	0.00000
RSI_14	0.9834	0.00000
RSI_20	0.9737	0.00000
VOL_10	0.7800	0.00000
VOL_15	0.7491	0.00000
VOL_20	0.7214	0.00000
CAD	0.9580	0.00000

Source: Own work.

2.5.2.2.2 Exploratory data analysis test for normality of features

To visualize the distribution of the observations we gather all our 12 features and produce a grouped histogram plot. With this type of visualization, we are collecting more user-friendly insights on the features' distributions.

Figure 1: Distribution of Observations for the Set of Features



Note. For ease of representation, the CAD feature is divided by 100.000.

Source: Own work.

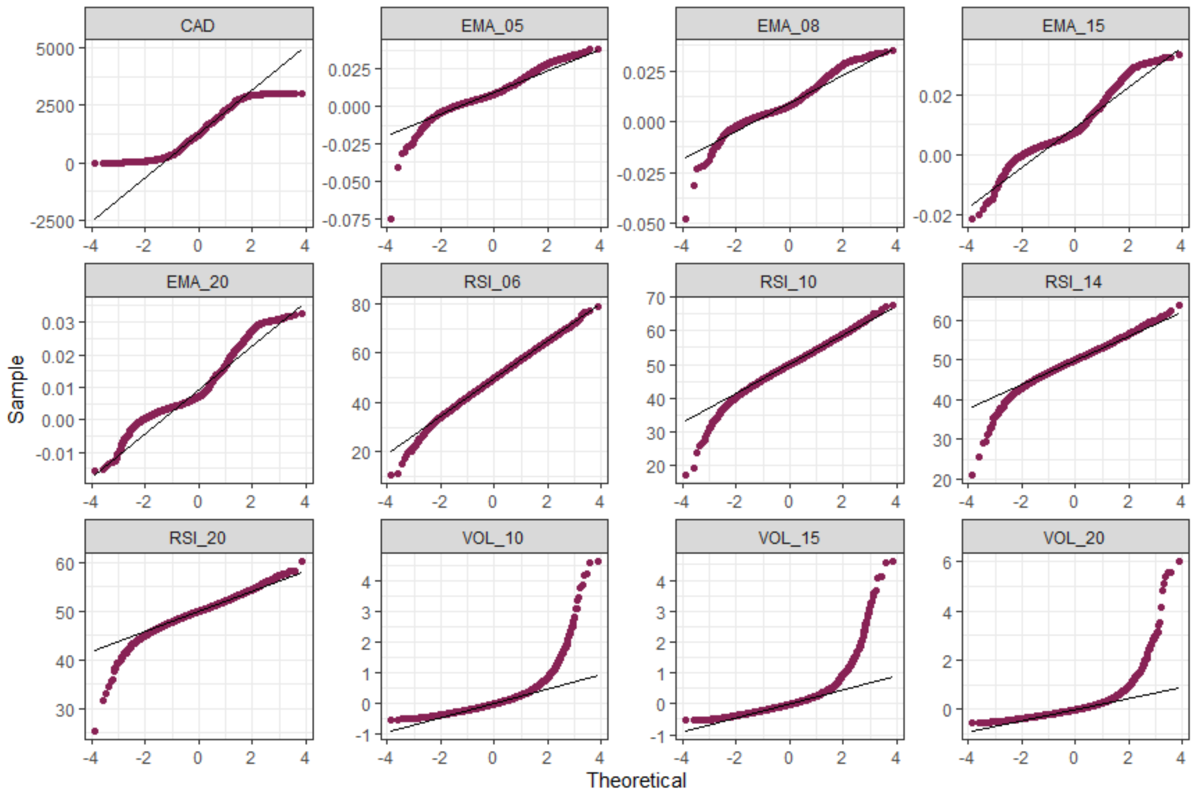
As expected, we can notice how the Chaikin accumulation-distribution results are not normally distributed. This is most likely caused by the possibility, that the volume attribute of the S&P500 Index is not a log-normally distributed process, but rather an explosive one. Additionally, we can observe how the infinitesimally small tails of the exponential moving average set of features change in magnitude as the time frame chosen increases. In other words, shorter time frames are characterized by a higher number of negative values than wider horizons, as imagined in Section 3.5.1. The actual number of negative observations for both EMA_05 and EMA_08 features, for which values majorly deviate from the mean, is considerably low and therefore we deem to classify such data points as outliers. This can be confirmed in Figure (2). We explain the lengthening of the right tail as a result of the underlying opening price being a quasi-log-normally distributed variable. Similar conclusions can be made for the set of RSI indicators, although the change in the magnitude of the tails is the opposite and overall this set of variables seems more normally distributed than the previous. The VOL set of features might be following a log-normally distribution, instead of a normal one. This can be additionally supported by the Q-Q plot below.

As a final test for supporting the interpretation of the above results, we are employing the second visualization method mentioned above, the Q-Q plots. These compare two probability distributions, a sample of data points for a feature and the theoretical distribution of the feature. Specifically, we are looking at a special case of a P-P plot, short for probability plot, namely the normal probability plot. Let S, a cumulative distribution function, be a sample of the set X. The normal probability plot is formed by the ordered response values on the vertical axis and the normal order statistic medians on the horizontal axis. The observations are plotted as a function of the corresponding normal order statistic medians. Each observation is defined as:

$$N_i = S^{-1}(U_i) \tag{13}$$

for $i = 1,2,\dots,n$, where S^{-1} is the inverse of the cumulative distribution function S, while U_i are the uniform ordered statistic medians, approximated by $U_i = 1 - U_n$ for $i = 1$, $U_i = \frac{i-0.3175}{n+0.365}$ for $i = 2,3,\dots,n-1$ and ultimately $U_i = 0.5^{\frac{1}{n}}$ for $i = n$ (Chambers, Cleveland, Kleiner & Tukey, 2017). Below the visualizations.

Figure 2: Q-Q Plots for the Set of Features



Note. Deviations of sample points from the diagonal line indicate departures from the theoretical distribution, which in our case is set to be the normal distribution.

Source: Own work.

Indeed, the CAD feature can not be defined as a normally distributed continuous random variable. As mentioned in the previous histogram analysis, the number of observations highly deviating from the mean is low for the set of EMA's. Moreover, we can confirm that, as we increase the time frame, the indicator seems to achieve a higher degree of normality. The RSI set of features appear to have the strongest degree of normality out of all the 12 features, while the VOL set does accumulate a relatively significant amount of data points deviating from their mean. Overall, we could be satisfied with the shape of our data, yet we want to take it a step further and apply three simple transformations to hopefully achieve an even higher degree of normality. In the next section, we introduce the Yeo-Johnson power transformation and the concepts of centring and scaling a continuous random variable.

2.5.2.3 Transformations applied to the set of features and interpretation of results

We chose to apply a Yeo-Johnson transformation on the predictors' data. This is a type of power transformation, which is applied to data to create a monotonic transformation it using power functions. It is widely considered as an extension of the Box-Cox transformation. The advantages of using such a technique are the ability to reshape the distribution of a continuous random variable into a normal or quasi-normal distribution, stabilize the variance and augment the validity of measures of association. Several authors advocate the use of power transformations in predictive modelling, with notable improvement in forecasting accuracy (Proietti & Lütkepohl, 2013; see also Kuhn & Johnson, 2013). The Box-Cox transformation is generally defined as:

$$y_i^{(\lambda)} = \frac{y_i^{\lambda} - 1}{\lambda} \tag{14}$$

and:

$$y_i^{(\lambda)} = \ln(y_i) \tag{15}$$

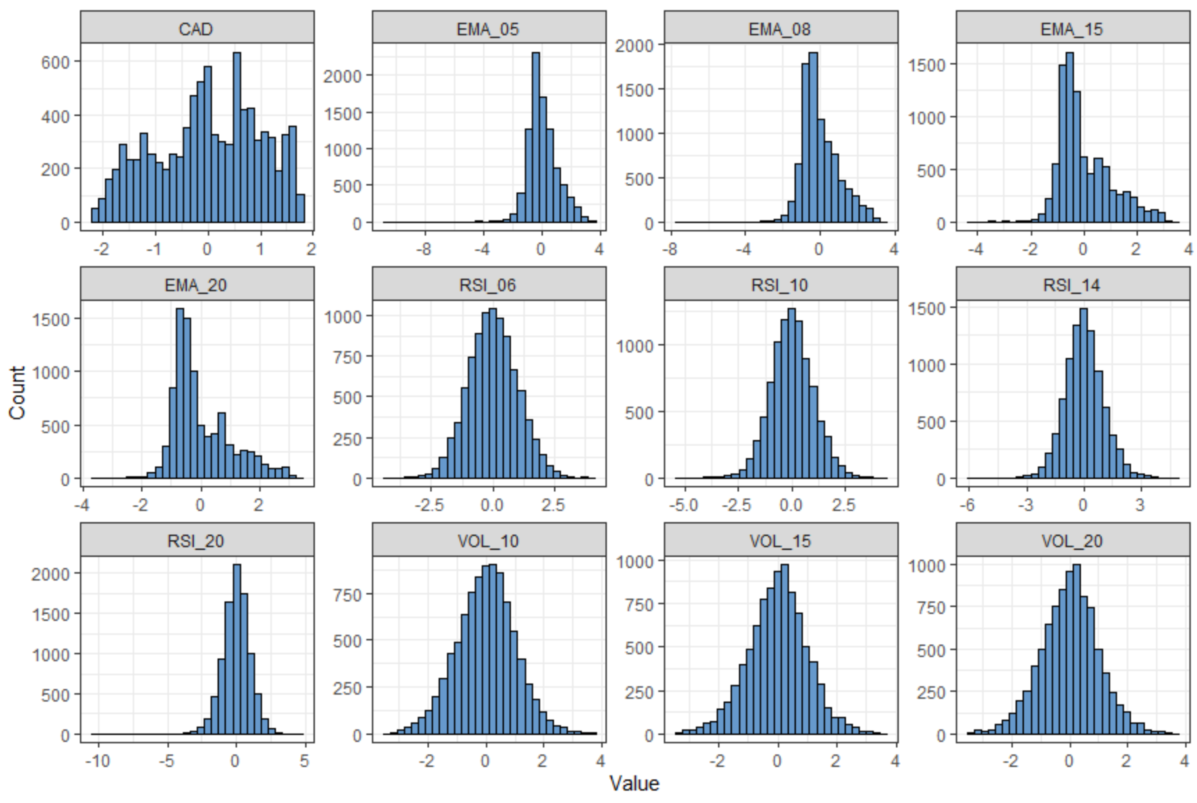
for $\lambda \neq 0$ and $\lambda = 0$, respectively. The optimal value for the parameter λ is chosen by computing the correlation coefficient of the normal probability plot. The value of λ corresponding to the maximum correlation coefficient is then the optimal choice for λ (Box & Cox, 1964). The Yeo-Johnson transformation allows also for zero and negative values of y and introduces a change in the natural logarithm for $\lambda = 0$ (Yeo & Johnson, 2000). Note, that we are not required to pass all 12 features through the power transformation. Only seven features are deemed to transformed by the *preProcess* function of the *caret* package in R, namely RSI_06, RSI_10, RSI_14, the VOL set and the CAD feature.

The next step is to perform feature centring and scaling. It is shown that this procedure can significantly improve the performance of several learning models, such as distance-based algorithms, while not affecting the performance of an algorithm, which is invariant to this technique (Zheng & Casari, 2018). Another group of models affected by a difference in the non-centrality and scale of features are gradient descent-based algorithms, such as logistic regression and neural networks, which make use of gradient descent as a technique to find the minima of a programming function. Tree-based algorithms are instead unaffected by the shape of the features' distributions and their scale (Kuhn & Johnson, 2013). Thus, we apply the process of standardization to our features, which consists of subtracting the mean from each observation and divide by their standard deviation. We then end up with a set of standardized values for each feature denoted as Z and defined as:

$$Z = \frac{X - \mu}{\sigma} \tag{16}$$

Finally, we show the results for the applied transformations. Again, we first perform a Shapiro-Wilk test and then additionally evaluate the changes with histograms and Q-Q plots. The test statistics produced do not substantially differ from Table 2, confirming that, according to the test, we should reject the null hypothesis and accept that our features are not normally distributed. Moreover, the visualization methods give us again another perspective.

Figure 3: Distribution of Observations for the Set of Features After the Transformations

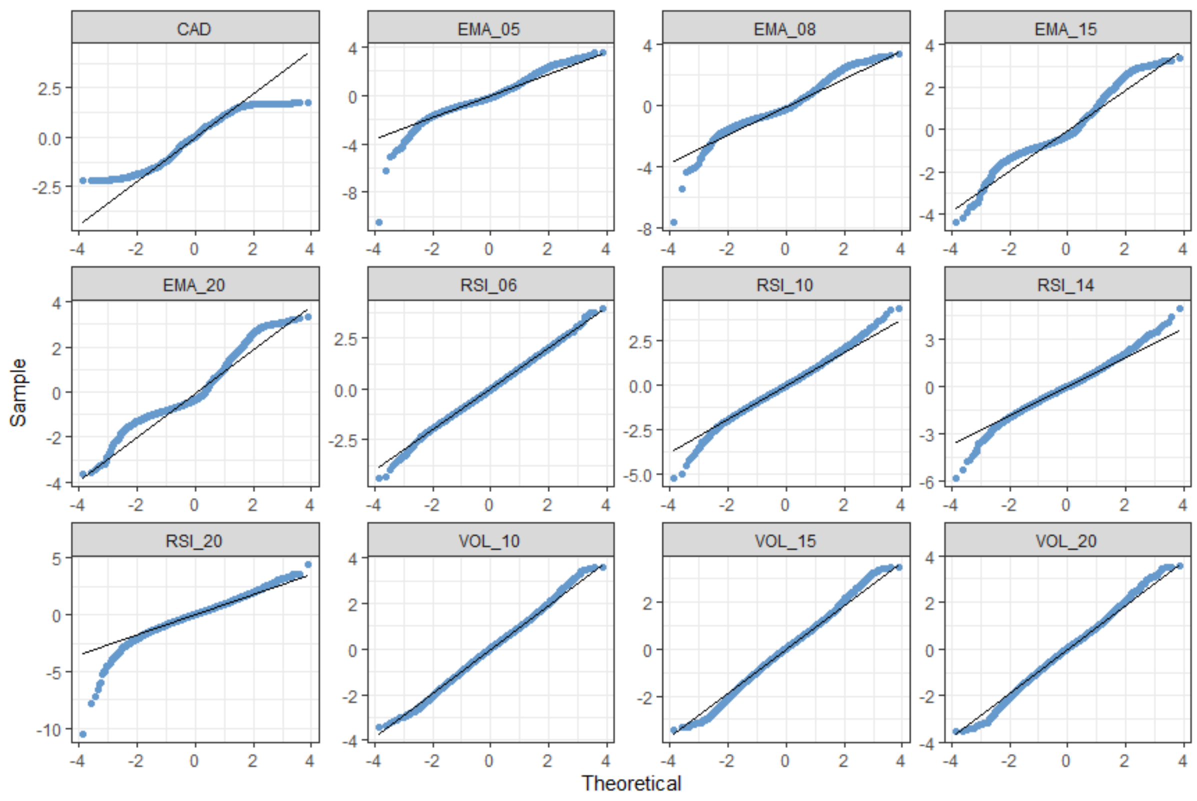


Note. For ease of representation, the CAD feature is divided by 100.000.

Source: Own work.

Clearly, from the figure above, we can see that we have improved the normality of all the set of features, except for the CAD feature, for which we are still unable to achieve a higher level of it. Our statements are furtherly supported by the following Q-Q plot.

Figure 4: Q-Q Plots for the Set of Features After the Transformations



Note. Deviations of sample points from the diagonal line indicate departures from the theoretical distribution, which in our case is set to be the normal distribution.

Source: Own work.

At this point, we conclude with the improvements on our dataset. We are satisfied with the quality of data and are ready to employ the learning algorithms to produce the desired daily forecasts. Before jumping straight to the introductions and definitions of the chosen set of algorithms, we want to quickly explore the statistical relationships between the features and the outcome, as well as between the features themselves, by conducting a correlation analysis.

2.5.2.4 Correlation analysis

For the last sub-section of section 3, we assume that the relationship between the predictors is linear, and we perform correlation analysis to evaluate the strength of such a relationship for each pair of the transformed predictors. The purpose of performing this analysis is to find whether we are incurring a problem of multicollinearity, which can be a method for explanatory variable (feature) selection (Blessie & Karthikeyan, 2012). Detecting multicollinearity through correlation analysis is one of the several tested methods used in model building across several fields of science (Mansfield & Helms, 1982; Dohoo et al., 1997). Formally, multicollinearity is the event in which a predictor can be linearly and accurately predicted from the other predictors in the set. If perfect multicollinearity exists, then a feature, say X_1, can be written as:

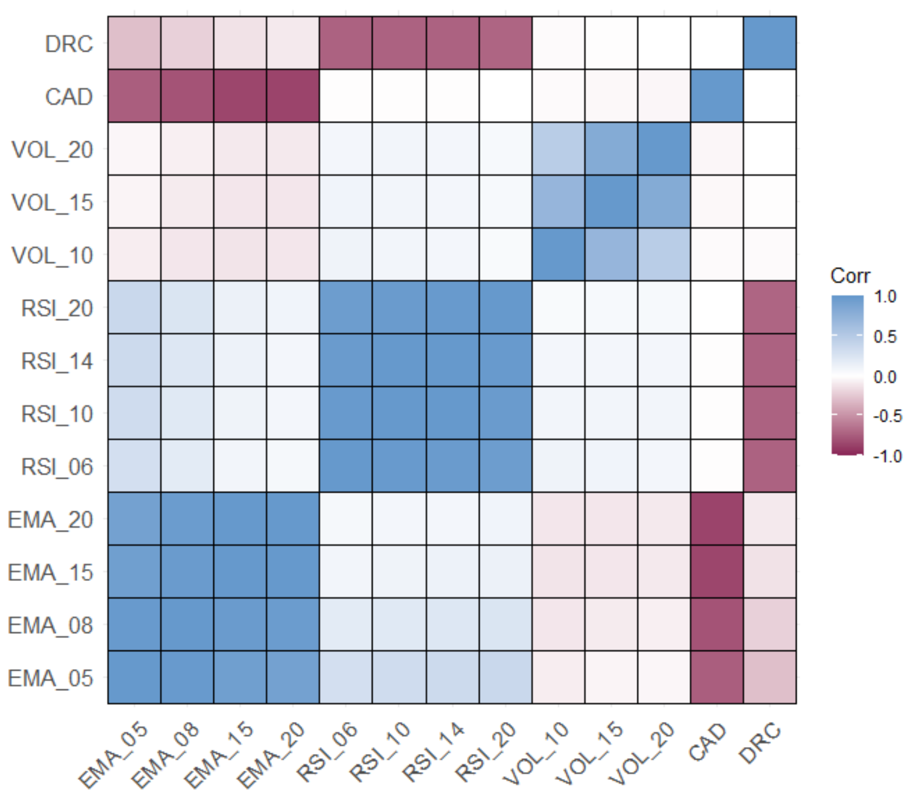
$$X_1 = a_0 + a_2 X_2 + \cdots + a_k X_k \tag{17}$$

where a's are constants. A possible consequence of multicollinearity in the predictors' set is a loss in precision for the estimation of one predictor's impact while controlling for the others, on the outcome variable DRC. Other consequences are information redundancy, deterioration of predictors' importance, increase in standard errors of coefficients for collinear predictors and model overfitting (Mansfield & Helms, 1982). To mitigate the issues, correlation measures, which can indicate the presence of multicollinearity, are performed via the Pearson product-moment correlation coefficient estimated using the sample correlation coefficient for each pair of predictors:

$$r_{X_1 X_2} = \frac{\sum_{i=1}^{N} (X_{1_i} - \widehat{X_1})(X_{2_i} - \widehat{X_2})}{(n-1) s_{X_{1_i}} s_{X_{2_i}}} \tag{18}$$

where $\widehat{X_1}$ and $\widehat{X_2}$ are the sample means of predictors X_1 and X_2, and $s_{X_{1_i}}$ and $s_{X_{2_i}}$ are the sample standard deviations of the respective predictors. Below we present the correlogram.

Figure 5: Correlogram of Features and Outcome



Source: Own work.

Regarding the relationship between each feature and the outcome, we can notice that every feature is slightly negatively correlated with the target variable DRC, except for the VOL set of features, which seems to be slightly uncorrelated. Moreover, we can notice how the correlations between the EMA's is quite important and the same accounts for the RSI set, while the same holds more weakly for the VOL and the CAD set. Using a table of p-values for the correlation coefficients in the table (3), we can determine, whether the resulting correlogram is hiding some relevant information. We can see, that the almost zero correlation between the VOL set and the target variable is insignificant, as the p-values produced are way higher than the 0.05 threshold. The same conclusion is made for the correlation between the RSI set and the CAD. We deem that the discovered strong correlations between EMA's and, in turn, RSI's, can be used as an argument for the presence of multicollinearity and could potentially be excluded from the set of features. Yet, we do not intend to use correlation analysis for feature selection, as we think, that the dimension of the feature set does not require any reduction, due to a potential loss of information provided by both sets. We believe, that having multiple versions of both indicators provide us with a bigger and more informative picture of what could happen in daily trade. We are now ready to present the learning algorithms.

Table 3: Correlation Coefficients p-Values

Features	EMA_05	EMA_08	EMA_15	EMA_20	RSI_06	RSI_10	RSI_14
EMA_05	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
EMA_08	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
EMA_15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
EMA_20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
RSI_06	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
RSI_10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
RSI_14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
RSI_20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
VOL_10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
VOL_15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
VOL_20	0.0004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CAD	0.0000	0.0000	0.0000	0.0000	0.3131	0.4558	0.5914
DRC	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Features	RSI_14	RSI_20	VOL_10	VOL_15	VOL_20	CAD	DRC
EMA_05	0.0000	0.0000	0.0000	0.0000	0.0004	0.0000	0.0000
EMA_08	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
EMA_15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
EMA_20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
RSI_06	0.0000	0.0000	0.0000	0.0000	0.0000	0.3131	0.0000
RSI_10	0.0000	0.0000	0.0000	0.0000	0.0000	0.4558	0.0000
RSI_14	0.0000	0.0000	0.0000	0.0000	0.0000	0.5914	0.0000
RSI_20	0.0000	0.0000	0.0000	0.0000	0.0000	0.7282	0.0000
VOL_10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0195	0.1478
VOL_15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0013	0.6235
VOL_20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.7416
CAD	0.5914	0.7282	0.0195	0.0013	0.0001	0.0000	0.7786
DRC	0.0000	0.0000	0.1478	0.6235	0.7416	0.7786	0.0000

Source: Own work.

2.6 Learning algorithms

In our work, we employ two classes of classification models, linear and non-linear models, for a total of five different learning algorithms. Under the class of linear models, we choose two models, namely a logistic regression and a support vector machine with a linear kernel. For the class of non-linear models, we use a K-nearest-neighbors and a multi-layer-perceptron model. Finally, for the third class, we choose a simplified random forest algorithm. We remark that each model is chosen to be presented based on its recurrence in the existing literature, where such models are usually appraised for their performances in financial applications, especially stock and index predictions. Before introducing the selected algorithms, we are providing a section on data partitioning and model fitting, as these are two processes of crucial importance in building machine and deep learning algorithms with significant performance.

2.6.1 The problem of overfitting

Modern classification and regression models are capable of modelling complex relationships mainly due to their high level of adaptability and learning power. On the other hand, models can easily overemphasize certain patterns that might not be reproducible in out-of-sample situations. This phenomenon is known as over-fitting, a modelling error that occurs when a model is too closely fit a limited set of data points and consequently leads to a deterioration of the model's predictive power when using new samples of data. In different terms, over-fitting occurs when a model is memorizing training data rather than learning to generalize from it (Dwyer, 2005). In this section, we present some statistical methods that help us to overcome the overfitting problem and to consequently produce powerful and consistent models to be used in an out-of-sample application.

A common characteristic of modern models is to have a set of parameters, which can not be directly estimated from the sample of data. These parameters are also referred to as tuning parameters. As an example, consider the K-nearest-neighbors classification algorithm, where a new sample is predicted based on the K-closest data points. The modeller has to choose several K-neighbors that does not underfit and simultaneously overfit the data. Too few neighbours may lead to overfitting, whereas too many neighbours may not be sensitive enough to produce acceptable performance. The answer to this problem has no analytical form and therefore the set of possible optimal, or sub-optimal, parameters have to be determined using a different method. Support vector machines can also have several tuning parameters, one being the price for misclassified samples in the training set known as the cost parameter (Irizarry, 2019). Again, we are not able to analytically determine the right value for the cost. What we can do instead, is to train our models each time with a different set of values for a specific tuning parameter and then aggregate the results. Our goal in this part is to set up exactly this scenario, where we can train our models to find the best possible tuning parameter that allows us to achieve the best possible level of a chosen performance metric and then to test these models on new samples of data.

One way to achieve the above is to first split our final version of the dataset into two separate sets, the training and the testing set respectively. The training set is used to fit a chosen model, while the test set is used exclusively to provide inputs to the best-trained model so that predictions are generated and compared to the actual outcomes. This is exactly how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output. There are several ways to split the data and the simplest way is to take a simple random sample and attribute it to the training set while attributing the remaining observations to the test set. As we are working with time-series data, simple random sampling might not produce reliable results, as it would not make sense, for example, to use future values to predict past ones. For this reason, we choose to use time-series cross-validation to train and test the performance of our models. Although it is common practice to split the dataset into two subsets, we are not guaranteed that our estimates can be fully trustworthy. For this reason, we make use of cross-validation, particularly time-series cross-validation. Also known as rotation estimation, cross-validation aims at testing a model's ability to correctly classify or predict new data samples, that are not used in estimating it. Generally, the process starts by first partitioning a set of data into several complementary subsets, second by testing the model on one subset, the training set, and thirdly by validating the testing on another subset, the testing set. To account for variability, most methods perform these steps multiple times and use different partitions of data. Finally, measures of fitness are aggregated to derive a more accurate estimate of model classification or prediction performance. The idea of time-series cross-validation is to start with a small subset of data for training purpose, forecast/classify for the later data points and then checking the accuracy for the forecasted/classified data points. The same data points are then included as part of the next training dataset and subsequent data points are forecasted or classified. We therefore must choose an initial window, which is the initial number of consecutive values in each training set, and a horizon, which is the number of consecutive values in each testing set (Kuhn & Jonhson, 2013). Again, there is no trivial solution to this problem and we could be free to test different values for both parameters. We choose to set the initial window to 70 and the horizon to 30. We now move to the presentation of the chosen learning algorithms.

2.6.3 Linear models

Generally referred to as classifiers, in such a context, linear models use linear functions to separate classes. Formally, we are referring to such models, by which the decision boundaries, also referred to as decision surfaces, are linear functions of an input vector, say X, and hence are defined by $D - 1$-dimensional hyperplanes within the D-dimensional input space, where $D = 12$, as we have 12 input vectors (predictors). The classification is then decided based on the values of a linear combination of all the input vectors, also known as feature values and feature vectors respectively. Generally, if the input vector X is a real vector, then the outcome denoted by Y is defined as:

$$y(X) \quad = f(W^T X + W_0) \tag{19}$$

where W^T is a real transposed vector containing the weights learned from the set of labelled training samples, W_0 a bias (not in the statistical sense) and f a threshold function, also referred to as the activation function, for which is not required to be linear. The activation function assigns the values of the dot product $W^T X$ plus the bias to either one class or another depending on the threshold, which is the negative of the bias, namely:

$$y(X) = \begin{cases} 1 & \text{if } W^T X > - W_0 \\ -1 & \text{else} \end{cases} \tag{20}$$

In the context of classification, we have three major classes of methods used for determining the parameters of the vector W, namely discriminative, generative and alternative-generative models. Generative models are used for modelling the conditional probability function $p(C_k|X)$ at an inference stage and then use the distribution to make an optimal classification decision. Alternative-generative models take the same conditional probability function together with the prior probabilities $p(C_k)$ for the classes and then compute the posterior probabilities using Bayes' theorem. The simplest of the three approaches is given by discriminative models, which are the models chosen in this section of the work. This class of models includes classifiers such as logistic regression, perceptron and support vector machines, to cite a few. The common goal of these models involves constructing a discriminant function that directly maps each input vector X to a specific class. This is achieved through supervised discriminative training, which is the process of training linear classifiers with an optimization algorithm that is given a training set, the desired outputs and a loss function. The loss function measures the differences between the output given by the linear classifier and the desired output. Therefore, the learning algorithm seeks a solution to an optimization problem of the form:

$$\underset{W}{\operatorname{argmin}} R(W) + C \sum_{i=1}^{N} L\,(y_i, W^T x_i) \tag{21}$$

where $R(\cdot)$ is a regularization function preventing the parameters from taking extreme values, which are the cause for model overfitting, L is the abovementioned loss function and C is a scalar constant used to control the balance between $L(\cdot)$ and $R(\cdot)$, or, in other words, to achieve a balance between regularization and the loss function. For linear logistic regression, the loss function takes the form of the log loss function, while for support vector machines the hinge loss function is usually preferred. When the regularization function $RR(\cdot)$ is convex, then the optimization problem can be interpreted as a convex problem. In such case the solution to the convex problem can be given by many algorithms, such as gradient descent, coordinate descent and Newton methods (Witten, Eibe & Hall, 2011; Curtis & Scheinberg, 2017).

When a data set contains classes that can be separated exactly by linear decision boundaries, then we are referring to a linearly separable set. We can expect that in most cases classes are not exactly linearly separable, yet we should not completely neglect the use of this type of models for several reasons. Most commonly, linear models are preferred for their ease of use, speed of computation, especially when dealing with a high number of dimensions, and interpretation. Following are the description of the two linear classification models used in this work, the logistic regression and the support vector machine with a linear kernel (Kuhn & Johnson, 2013).

2.6.3.1 Logistic regression

Before dwelling into the description of logistic regression, we deem it important to introduce the distribution of our outcome variable. A binomial distribution is a probability distribution that is often used to model events with two possible outcomes. Such distribution has only one parameter, that is the probability of a specific outcome denoted by p and its likelihood function defined as:

$$L(p) = \binom{N}{k} \cdot p^k q^{N-k} \tag{22}$$

where N is the number of total outcomes and k is the number of events with the outcome chosen to be positive, for example, an upward movement or a win at a lottery. The value of p that generates the largest value for $L(p)$ is found by the maximum likelihood estimator, which is the sample proportion of positive outcomes to total outcomes. The positive outcome, which in this work is the upward movement, is affected by multiple factors, or better, the predictors, represented by the technical indicators. We, therefore, want to create a model that uses the predictors to generate a more refined probability estimate and therefore we re-parametrize the binomial distribution model in such a way that p becomes a function of the predictors.

This is made possible by the logistic regression model. Used to model the probability of an event or class with binary values, the logistic regression is a statistical model that utilizes a logistic function to model a dependent variable, which can take two values, yet more complex extensions exist allowing the dependent variable to take more than two values. The logistic model contains the slope parameters for every predictor and an intercept. A problem would arise when we are not guaranteed that a slope and intercept model allows the probability of the outcomes to be in the range [0,1]. Logistic regression avoids this problem by modelling the log odds of the positive outcome, denoted by $\frac{p}{1-p}$, as a linear function defined as:

$$\log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_P x_P \tag{23}$$

where P is the number of predictors, and then, since the range of such function is $(-\infty, \infty)$, the logistic model reshapes the log odds into a function of the event probability p known as a sigmoidal function, that constrains the probability estimates to the desired [0,1] range:

$$p = \frac{1}{1 + \exp[-(\beta_0 + \beta_1 x_1 + \ldots + \beta_P x_P)]} \tag{24}$$

The classification boundaries produced are then linear, regardless of the nonlinear nature of the outcome function p. The answer to the question of why can a logistic regression model be referred to as a linear model is found in the fact that the outcome function is modelled using linear predictors (see: equation above). The next step is then to find the candidate values for the β's, the slopes of the predictors. This is done by using our data to compute a value of the likelihood function and, once we find the β values that maximize the likelihood function for our data, we can use these values to predict the outcomes (Dobson, 2002).

The advantages of the logistic regression model for classification problems are represented by the model overall simplicity, the ability to conduct a formal hypothesis test to assess the statistical significance of the predictors' slope coefficients, usually through a Z statistic, and consequently assess the significance of the relationship between the predictors and the outcome. A disadvantage, relative to other classification models, can be the requirement to manually identify the effectiveness of the relationship between the predictors and the outcome, and the statistical significance of the slope of the predictors (Kuhn & Johnson, 2013).

2.6.3.2 Support vector machines

Another group of models that can be used in a classification problem is delineated by the group of support vector machines. These are a type of algorithms that given a set of observations alongside an outcome, known also as training examples, can build a model that assigns new samples to one class or another in a non-probabilistic way. A support vector machine is therefore a supervised non-probabilistic classifier, which can be used for both linear and non-linear classification. Extensions of this group of models exist also for unsupervised learning problems. Briefly, support vector machines treat data points as a N-dimensional vectors and separate these points with a $N - 1$ dimensional hyperplane, known as a classifier. The classifier is chosen based on maximizing the distance between the classifier itself and the nearest data points, which also provides the lowest generalization error possible. Such a classifier is known as the maximum-margin classifier and the generalization error is a measure of how accurately the algorithm can generate accurate outcomes for any new sample of data (Cristianini & Ricci, 2008).

Recall, we are dealing with two possible outcomes. When the training examples are completely separable, then we chose two parallel hyperplanes that maximize the distance between the maximum margin hyperplane, which is the hyperplane that lays in the middle of the region bounded by the two hyperplanes, and the two classes of training examples. When the predictors' data is normalized or standardized, then the two hyperplanes forming this region, called margin, are defined as:

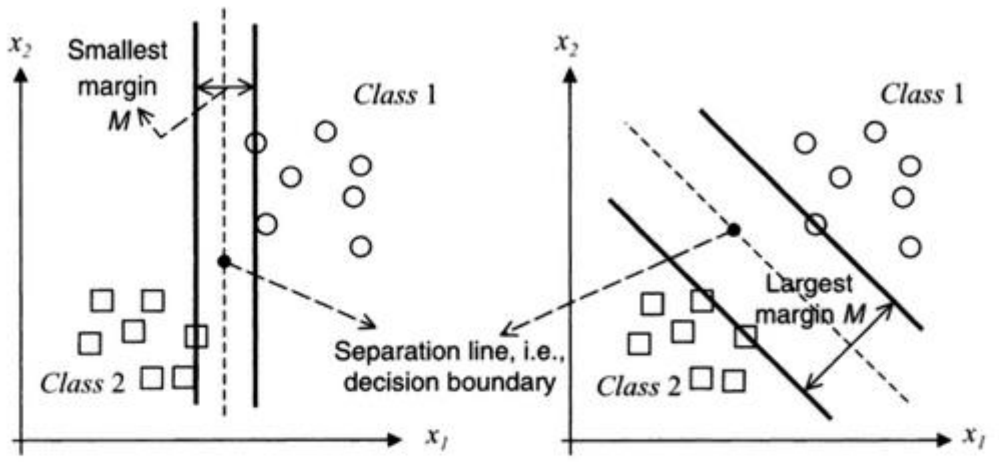
$$W^T X - B = 1 \tag{25}$$

for which any training example above this hyperplane is classified as an upward change (class 1) and:

$$W^T X - B = -1 \tag{26}$$

for which the training examples below this hyperplane are classified as a downward change (class 2). We provide an illustrative example of the algorithm and its decision boundaries below.

Figure 6: Example of a Support Vector Machine Algorithm with a Linear Classifier



Source: Wang (2005).

The geometrical distance between these two hyperplanes is computed as $\frac{2}{\|W\|}$. Since we want to find the classifier that maximizes this distance, we need to minimize $\| W \|$. The optimization problem is formulated by adding the following constraint:

$$y_i(W^T x_i - B) \geq 1 \tag{27}$$

for all $1 \leq i \leq n$. This prevents the training examples from falling into the margin region, consequently stating that each training example must be on the correct side of the margin. The minimization problem is then formulated as:

$$\min \| W \| \tag{28}$$

subject to $y_i(W^T x_i - B) \geq 1$ for $i = 1, \dots, n$. On the contrary, when two or more classes are not completely separable we can introduce an extension to the maximum margin classifier in the form of a cost on the sum of the training examples that are located on the boundary or wrongly classified. This type of classifier is called a soft-margin and the cost function is usually referred to as the hinge loss function, denoted as λ. Now the optimization problem changes into:

$$\min[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(W^Tx_i-B)\right)]+\lambda \parallel W \parallel^2 \tag{29}$$

This problem can be solved by the classical approach, which consists of reducing this optimization problem to a quadratic programming problem, or with modern approaches such as gradient and coordinate descent alongside their numerous formulations. In this work, we opt for the gradient descent approach. Gradient descent is a first-order iterative algorithm useful for finding a local minimum of a differentiable function by taking consecutive steps in the opposite direction of the gradient of the objective function. The learning rate, which we denote as η is the only parameter of the gradient descent. This determines the step size at each iteration while progressing towards the minimum of the function above. We must be careful at choosing the learning rate, as a too high rate might lead to overlooking the minimum (known as overshooting), while a too low rate might significantly slow the rate of convergence to the minimum or lead the descent to be stuck in a local minimum. Our choice for the learning rate of 0.01 is based on the common practice in the existing literature. Moreover, we opt for an alternative form of descent, namely the stochastic gradient descent, which is a stochastic approximation of the gradient descent algorithm. Instead of calculating the actual descent, which is computed on the entire set of predictors' observations, we are estimating a descent, which is calculated as an average of the gradient descents of randomly selected subsets (through resampling) of the observations. The advantages in comparison to the traditional gradient descent are reduced computational times, faster iterations and reduced risk of getting stuck in a local minimum, while a disadvantage is represented by a lower convergence rate (Witten, Eibe & Hall, 2011).

Note that with the addition of λ the classifier is now penalized due to misclassification in the process of estimating weight values. The cost value is a tuning parameter that controls the complexity of the maximum margin classifier by penalizing the number of errors. As the value of the cost increases, the classifier shifts and reshapes itself to correctly classify as many of the training examples as possible. For this reason, increasing the cost value leads to higher model complexity which can, in turn, lead to overfitting and consequently an increase in the generalization error. An acceptable value for the cost, which finds a reasonable balance between over- and under-fitting, is then found with the help of an arbitrarily selected grid of possible cost values, ranging from zero to two (with a total length of 20 values), and the resampling process presented in the Section 3.7.1. What we do is then training 20 different support vector machines, each with a different cost value, and then we select the model based on the cost value that yields the highest accuracy (Curtis, & Scheinberg, 2017).

The linearity of the presented support vector machine model lays in the fact that predictors enter the dot product $W^T X$ linearly. We centred and scaled the predictors' data to avoid the problem of having attributes with large values in magnitude that can dominate the dot product calculation. Note, that we can substitute the dot product with a kernel function $K(\cdot,\cdot)$ that allows us to create extremely flexible margin classifiers, which implicitly learn to create linear separators in higher dimensional spaces, which are then non-linear in the original dimension space. Popular kernel functions are the polynomial, radial basis and hyperbolic tangent, yet we are not providing further explanation for the latter. In this work, we are interested in using a support vector machine with a linear classifier, therefore the choice for the kernel is the linear kernel function, which is simply the dot product $W^T X$ itself (Kuhn & Johnson, 2013). We now turn to the chosen non-linear models.

2.6.4 Non-linear models

The two classification models presented above use linear functions to separate classes. When the data points are not linearly separable, we can depend on the model, introduce a cost function for misclassification. Another approach is to use non-linear classification models, which use non-linear functions to separate classes. Popular non-linear models used also in this work, are the K-nearest neighbours (KNN), support vector machines (SVM) with non-linear kernels (not used in this work), multi-layer-perceptrons (MLP), and random forests (RF), to name a few. Some models, such as KNN's and SVM's use the concept of similarity and distance, respectively, to train the classification process, while models such as MLP's and CT's generate classifications based on the characteristics of the predictors (Witten, Eibe & Hall, 2011).

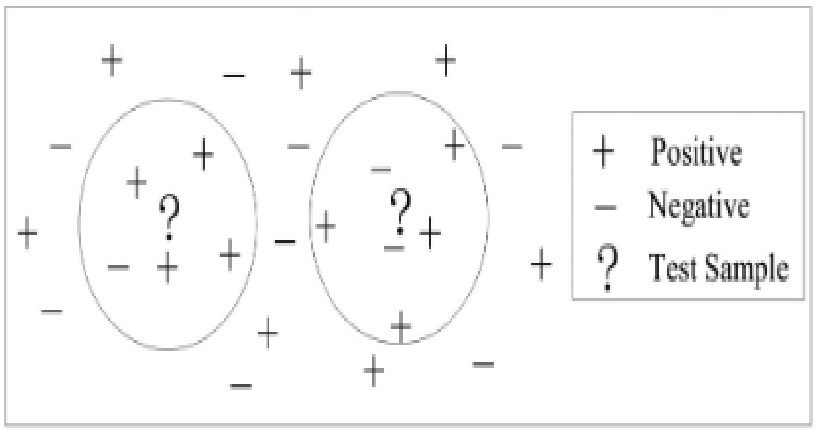
2.6.4.1 K-nearest-neighbors

Abbreviated as KNN, this model is a type of non-parametric algorithm used for both classification or regression problems. In contrast to other methods, a KNN algorithm for classification uses the geographic neighbourhood, known as K-closest points, of a training example to determine the belonging class. The K-closest points are determined by a distance metric, usually a Euclidean or a Minkowski metric. Our KNN algorithm is based on the Euclidean distance metric, defined as:

$$(\sum_{i=1}^{N}(x_{a_i} - x_{b_i})^2)^{\frac{1}{2}} \tag{30}$$

For this reason, it is important to apply a scaling transformation on the whole set of predictors, as the original measurement scales of the predictors are generally different and thus we want to avoid biased calculations. When a new sample of data is introduced, the algorithm maps the sample's k-nearest neighbours in the predictor space and calculates the response as a mean of the K-neighbors' responses. In other words, class probability estimates are calculated as the proportion of training examples neighbours in each class. If the new data point predicted class has a higher probability than the other class, then this new data point is assigned to the first class instead of the second, which logically has a lower probability. In the case of equal probabilities, the new data point is either randomly assigned to a class or is assigned based on the $K + 1$ closest neighbour (Guo, Wang, Bell, Bi & Greer, 2003). Again, we must make use of the presented resampling technique to estimate an optimal number of the model's only tuning parameter, the number of K-neighbors. We provide the algorithm with the instruction to search for the optimal number of K-neighbors in the range $[1, .., 12]$, where twelve is the number of our predictors and, with the help of cross-validation, we find the optimal value for K. A visual example of a KNN algorithm is found below.

Figure 7: Example of a K-Nearest Neighbors Algorithm for Binary Classification



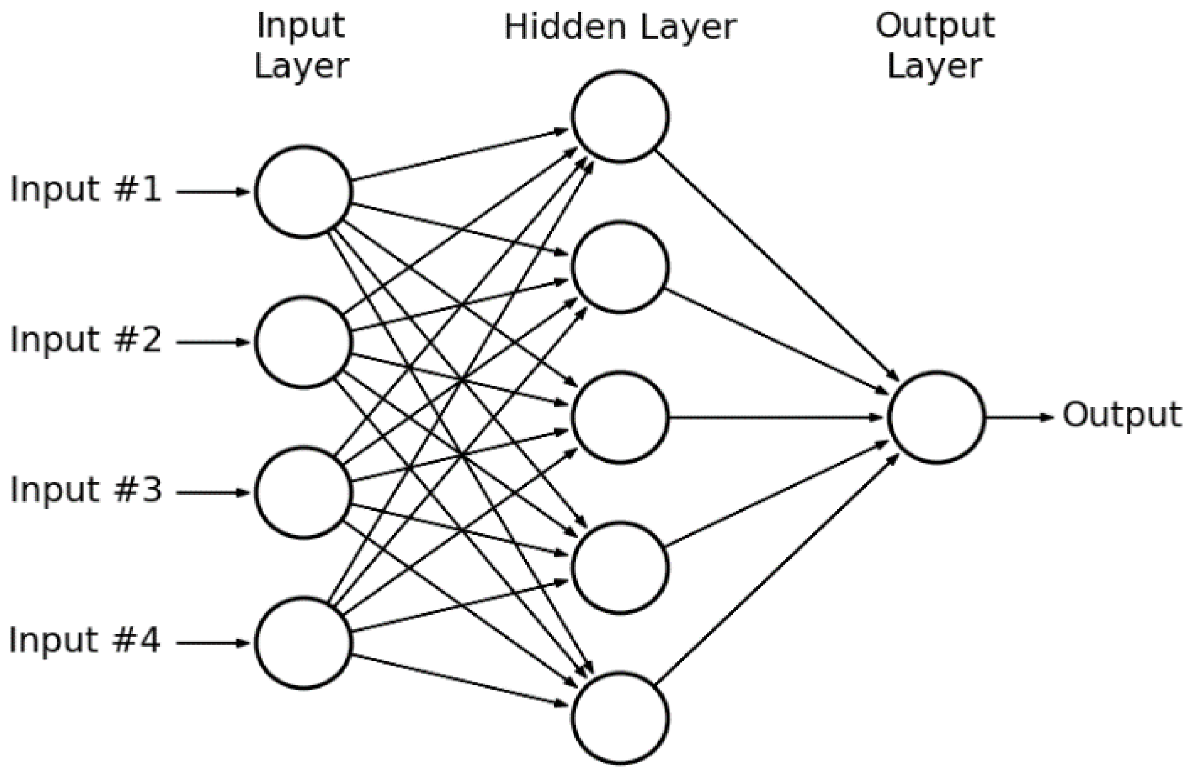
Source: Zhang, Li, Zong, Zhu & Cheng (2016).

Overall, we are working with an elementary version of the KNN model, which could be able to produce satisfactory results and be easy to interpret. On the other hand, its simplicity comes with two common drawbacks, namely computational time and the possible discrepancy between the local structure of the predictors and the KNN's classification power. For every sample we want to make a classification, we must compute the distances between the latter and all other samples. Therefore, computation time increases by a factor of n. Regarding the local structural problem, when a predictor has an irrelevant local structure, this can lead to distancing similar samples, in the predictor space, away from each other. We mitigate the local structure problem by weighting the neighbour's contribution to the prediction of a new sample based on their distance to the new sample. Differently, training examples closer to the new sample have a higher contribution rate in the classification process and need to be weighted accordingly. In our case, the weight is calculated simply as $\frac{1}{K}$ (Guo, Wang, Bell, Bi & Greer, 2003).

2.6.4.2 Multi-layer-perceptron neural network

The second non-linear model used is the multi-layer-perceptron algorithm, abbreviated as MLP, which is a type of neural network. Unlike the KNN model, the MLP generates classifications based on the characteristics of the predictors. More precisely, the MLP is a feedforward neural network, which is a type of artificial neural network composed of an input layer, a hidden layer (or more) and an output layer. Each layer is made of nodes, usually called artificial neurons, that are connected by the edges (like synapses in a biological brain). The nature of the connections, which transport information, depends on the type of the neural network. The term feedforward relates to the characteristic of such a neural network. In fact, in a feedforward network, the information is moving in only one direction from the input layer through the hidden layer(s), arriving at the output layer. Therefore, in an MLP there are no loops in the network, which translates to the absence of edges between nodes in the same layer and, as mentioned, unidirectional movement of information. The information is a real number that travels through the edges to the nodes of each layer, where output is generated by a non-linear activation function of the sum of inputs except for, obviously, the input layer. All nodes are assigned weights that are gradually updated. The output of the output layer is the value we are looking for, the classification (Murtagh, 1991). Below, a visual representation of the algorithm.

Figure 8: Example of a Multi-Layer-Perceptron Algorithm



Note. The number of nodes in the figure above (21) does not represent the total number of nodes for our MLP algorithm, which is 32 total nodes with 12 input nodes, 15 hidden nodes and two output nodes for the directional change value.

Source: Hassan, Negm, Zahran & Saavedra (2015).

Formally, the MLP is a supervised learning algorithm that, based on the training examples, learns a function:

$$f(\cdot): R^m \mapsto R^o \tag{31}$$

where m is the number of dimensions for input and o is the number of dimensions for output. Our MLP model consists of three layers: an input, a hidden and an output layer. The input layer is constructed by the predictors, therefore we have twelve nodes in the first layer ($m =$

12). The values of the input layer travel through the edges to the three hidden layers, comprising nine, six and three nodes respectively. The choice for the optimal number of hidden layers and nodes is still an open debate in the academic world, as no analytical solution to the problem has been found up to this date. For the number of hidden layers, we stick with the rule of thumb in Hornik (1991), where it states that several hidden layers larger than two can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy. The number of hidden nodes is chosen based on the proportionality rule presented in Cybenko (1989). We start with twelve nodes and we continue by subtracting three nodes at each layer, which means we have nine, six and three nodes in the hidden layers, respectively. Finally, we have two nodes for the output, one for the upward and one for the downward directional change. We are not able to perform a grid search to look for the optimal number of both components as with the previous two algorithms due to the extremely long computational times related to such a process. Finally, we end up with an output layer consisting of a single node, for which the output given is the classification. We are left to chose which activation function processes the information and in what way the algorithm learns to assign weights to each node. Regarding the first matter, two common activation functions are sigmoids functions. These are a hyperbolic tangent:

$$y(u_i) = \tanh(u_i) \tag{32}$$

and a logistic function:

$$y(u_i) = (1 + e^{-u_i})^-1 \tag{33}$$

where y_i is the output of the $i - th$ node and u_i is the weighted sum of the input edges. The first function has outputs in the range $[-1,1]$, while the logistic function in the range $[0,1]$. Our MLP algorithm uses the logistic function as the activation one. Regarding the second matter, this type of neural network learning is an example of supervised learning, which means we are analyzing the difference between the actual output and the algorithm's output. More specifically, assigning weights is done by backpropagation, which is an algorithm used for computing the gradient descent of the weights. The degree of error can be defined as:

$$e_j(n) = d_j(n) - y_i(n) \tag{34}$$

where $d_j(n)$ is the actual output of the $j-th$ node at the $n-th$ data point and $y_j(n)$ is the output generated by the algorithm. Based on the latter the node weights can be adjusted with gradual corrections that aim at minimizing the error rate in the entire output (Curtis & Scheinberg, 2017):

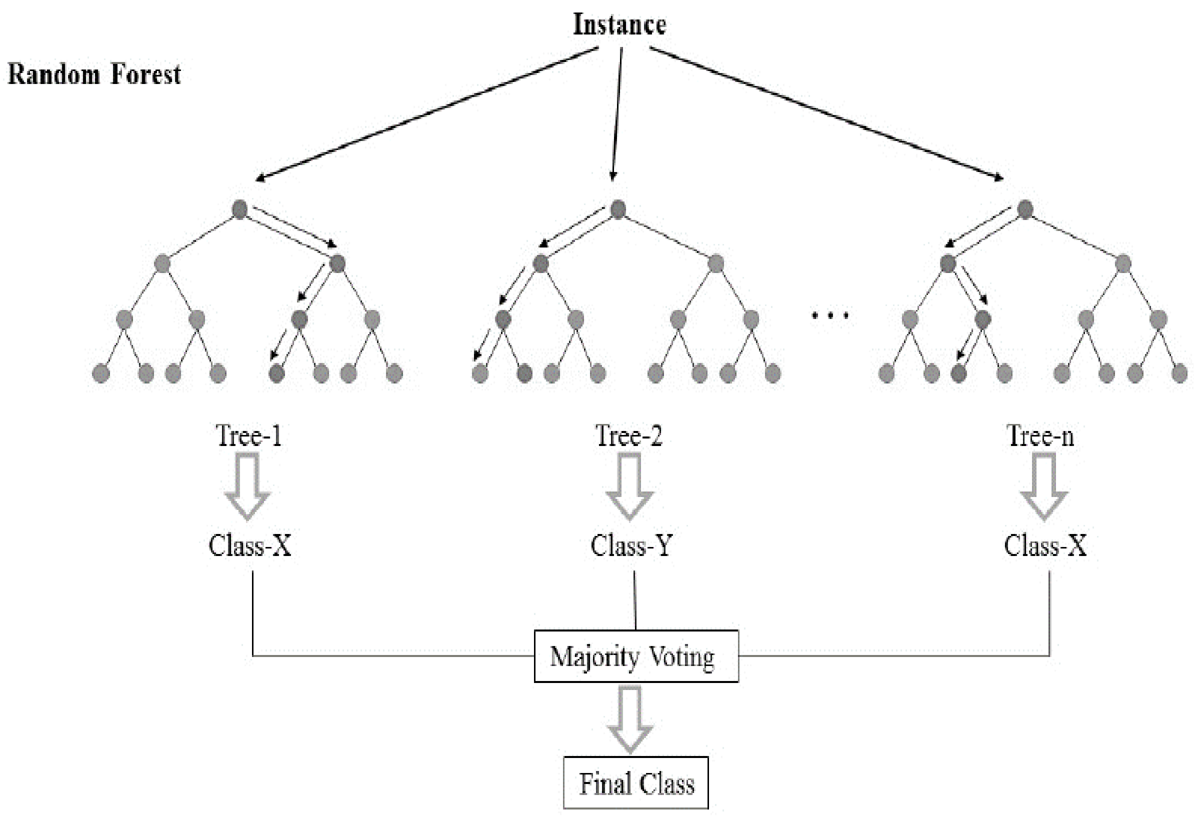
$$\epsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{35}$$

As for the support vector machine algorithm we chose a stochastic gradient descent with a learning rate $\eta = 0.01$. Our MLP algorithm is now finalized, and now we are ready to present the last algorithm used in this work, the random forest.

2.6.4.3 Random forest

A random forest is a supervised learning method used for classification and regression problems. Specifically, it is a collection of decision trees. In graph theory, a decision tree is an undirected graph in which any two vertices are connected by exactly one path. A forest, on the other hand, is an undirected graph in which any two vertices are linked by at most one path (Bender & Williamson, 2010). The forest is then an ensemble of decision trees, usually trained with the bagging method, for which the idea is that a combination of learning models (decision trees) increases the overall result in comparison to a single learning model (a single decision tree). Below, a hypothetical example of a random forest algorithm.

Figure 9: Example of a Random Forest Algorithm for Binary Classification



Source: Biau & Scornet (2016).

Given a set of training examples with respective responses, bagging selects a random sample with replacement M times from the set of training examples and fits every decision tree to these samples. Predictions for new samples of data are then made by averaging the predictions of all decision trees sampled during training:

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} y_b\,(x') \tag{36}$$

where x' is the set of new samples of data and M represents the number of samples, or, in this context, the number of trees. The advantage of bagging is represented by a reduction in the variance of the model without increasing the bias because the predictions generated by a single decision tree might be highly sensitive to noise in its training set, yet the average of many trees is not so sensitive as long as the decision trees are not correlated. On the other hand, training on a single tree might produce highly sensitive predictions. A measure for the uncertainty of the prediction can be defined as:

$$\sigma = \sqrt{\frac{\sum_{m=1}^{M}(y_m(x') - \hat{y})^2}{M-1}} \tag{37}$$

In a random forest algorithm, the procedure of bagging is slightly different. At each candidate split in the learning process, the algorithm chooses a random subset of the features (predictors). This method is called attribute or feature bagging and it is implemented to overcome the potential problem of highly correlated features of the original bagging method. In the original method if one or more features are important predictors for the output, then these features are selected frequently in M-trees, leading to high correlation. Lastly, the number of predictors used at each split is $\sqrt{P}$. Note that for a regression problem this number is different. The only parameter to be chosen in our random forest model is B, the number of trees. Again, there is no analytical method to retrieve the optimal number of trees, yet there are several proposed solutions. Usually, a few hundred to thousand trees are used depending on the size and nature of the predictors set. We settle for $B = 500$, as we find that the performance of our model stabilizes when the number of trees is equal to or higher than 500. Other proposed solutions in the literature are found using cross-validation or the out-of-bag error method.

As with the previous algorithms, the random forest model comes with advantages and disadvantages. One of the key advantages of this model is its avoidance of the overfitting problem when a sufficient number of trees is used in training. Another advantage is represented by the presence of only one tuning parameter. An important drawback is computation time. When B is large, so is the time required to generate predictions. In general, such a type of algorithm is fast to train, yet quite slow to create a prediction. In our case, computation time is not deemed as an important issue (Ho, 1998). Note, we provide the computation time for every algorithm used in the performance evaluation section. Having defined our dataset, analysed it from both the financial and statistical point of view, presented the learning models that will generate the daily predictions, we now turn our focus on how to evaluate the performance of our models.

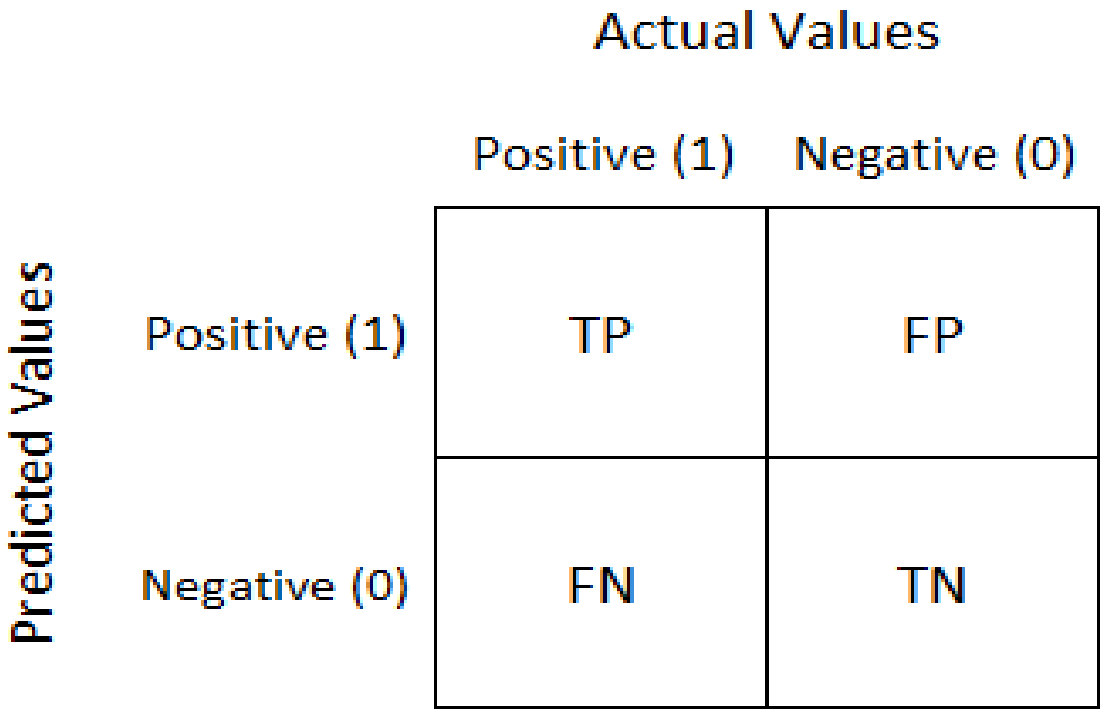
2.7 Methods for evaluating classification models

Intuitively, we want to understand how our learning models perform. Note, in a classification problem we need to evaluate model performance using different metrics than in a regression problem since measures such as the root mean square error (RMSE) and R^2 are not suitable for performance evaluation. Classification models generate two types of predictions, class predictions, which come in the form of a discrete variable, -1 and 1 in our case, and class probabilities, which are continuous-valued prediction in the form of a probability statistic ranging in the interval [0, 1]. A class prediction is generally useful in decision-making processes, while a class probability is more suitable to ascertain the confidence of our predictions (Gorunescu, 2011). Let us start by introducing several methods on how to evaluate class predictions.

2.7.1 Evaluating class predictions

A general method for describing the performance of a classification model is given by the confusion matrix, also known as the error matrix, which is a special kind of contingency table of the observed and predicted classes. An illustrative example follows below.

Figure 10: Example of a Confusion Matrix for Binary Classification



Predicted Values		Actual Values: Positive (1)	Actual Values: Negative (0)
	Positive (1)	TP	FP
	Negative (0)	FN	TN

Source: Own work.

The entries TP and FP stand for true and false positives, while FN and TN for false and true negatives. This table, therefore, reports the number of correctly and incorrectly classified instances.

The simplest evaluation metric, calculated using the confusion matrix, is the overall accuracy rate, also known as the error rate, reported as the percentage of correctly predicted classes. The issue with this metric is that it does not make a distinction about the type of errors being made, which are, depending on the context, of crucial importance (Provost, 1998). In this context, we can use the algorithms to create trading or investment strategy and therefore the cost of misclassifying the directional change can be great. Another issue with overall accuracy is given by ignoring the natural frequencies of each class. We prefer a metric that takes into consideration the weights of the two classes to avoid having a biased understanding of the models' performance. An example can be the case where the number of upward directional changes is far greater than the case of downward ones. One possible solution to this is to use balanced accuracy, which takes into consideration the latter issue. It is simply the sum of sensitivity and specificity divided by two, explained in the upcoming paragraphs.

On the other hand, a helpful metric when the number of instances for the classes is equal or insignificantly different is the no-information rate. The no-information rate is defined as the accuracy rate that can be achieved without a model. A simple computation is given by $\frac{1}{C}$, where C is the number of classes. This metric can be compared to overall accuracy. However, as the overall accuracy, the no-information rate suffers from potential disproportions in the classes and therefore might not be extremely indicative for the performance of a model (Kuhn & Johnson, 2013).

An additional performance metric built on the confusion matrix is the Kappa statistic. Also known as Cohen's kappa, this metric takes into account the accuracy that would be generated by chance. It is calculated as:

$$\text{Kappa} = \frac{O - E}{1 - E} \tag{38}$$

where O is the observed accuracy and E represents the expected accuracy based on the marginal totals of the confusion matrix. The range of values for this statistic is $[-1,1]$, where a value of 1 implies a perfect agreement between predicted and actual classes, a value of 0 means there is no agreement and a value of -1 indicates that the predicted classes are in the opposite direction of the actual ones. When the distributions of classes' instances are equal, the Kappa values produced are proportional. As a rule of thumb, Kappa values from 0.30 onward indicate reasonable agreement between predictions and actual values. Thus, this metric is employed as a measure of agreement (Cohen, 1960).

There are several other measures of agreement. We are especially interested in measures, which are insensitive to disparities in class proportions. In a two-class problem, where one class is interpreted as the event of interest (say one of the two-directional changes), arguably the most famous metrics are sensitivity and specificity. The sensitivity is defined as the rate of the event of interest that is correctly predicted for all samples of data, where the event happens:

$$\text{Sensitivity} = \frac{\text{number of samples with the event and predicted to have the event}}{\text{number of samples having the event}} \qquad (39)$$

On the other hand, specificity is defined as the rate that non-event samples are predicted as non-events:

$$\text{Specificity} = \frac{\text{number of samples without the event and predicted as non-event}}{\text{number of samples without the event}} \qquad (40)$$

Sensitivity is usually referred to as the true positive rate, as it measures the accuracy in the event population. Conversely, if we subtract specificity to 1 we achieve the false-positive rate. When sensitivity is increased, a model is likely subject to a loss of specificity, since more samples of data are being predicted as events of interest. The relevance of such a trade-off depends on the context. For example, in spam filtering the general focus is on specificity, as most users of email are willing to accept incoming spam emails if emails from a known group of individuals are not discarded (Loong, 2003). We can quantify the relationship between these two measures through the receiver operating curve presented in the next section (Kuhn & Johnson, 2013).

We must underline the probabilistic nature of sensitivity and specificity. Both measures are conditional. Namely, specificity is the accuracy rate for only the non-events of interest population and sensitivity for only the event of interest population. Usually, in a decision-making process, we are interested in the unconditional probabilities, for example, what is the probability that the directional change is positive. To answer this question, we need to account for sensitivity, specificity and prevalence of the event of interest in the population. By doing so, the analogue to sensitivity and specificity are the positive (PPV) and negative (NPV) predicted value, respectively. The positive predicted value provides the answer to the question "What is the probability that this sample is an event?". Both measures are formulated as (Loong, 2003):

$$PPV = \frac{\text{Sensitivty} \cdot \text{Prevalence}}{\text{Sensitivty} \cdot \text{Prevalence} + ((1 - \text{Specificity}) \cdot (1 - \text{Prevalence}))} \tag{41}$$

and:

$$NPV = \frac{\text{Specificity} \cdot (1 - \text{Prevalence})}{\text{Prevalence} \cdot (1 - \text{Sensitivity}) + (\text{Specificity} \cdot (1 - \text{Prevalence}))} \tag{42}$$

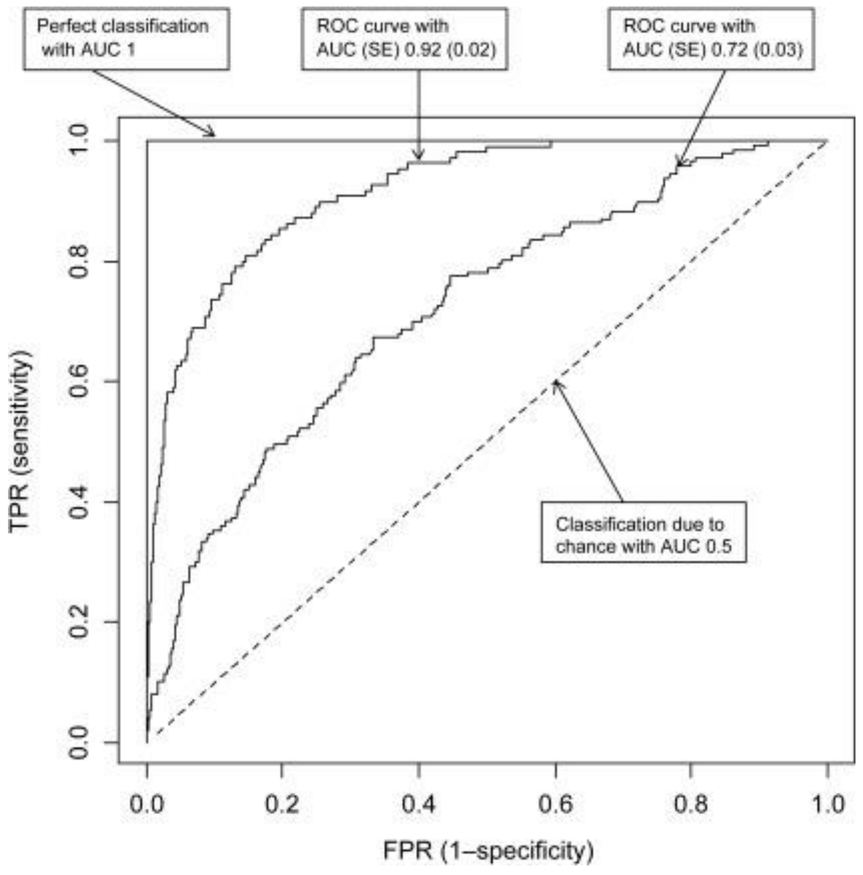
In our performance evaluation analysis, in Chapter 4, we provide the results for all the listed metrics alongside the interpretations. We now move on to the methods for analyzing class probabilities.

2.7.1 Evaluating class probabilities

As mentioned before, each classification comes with the probabilities of a data sample being classified either as 1 or -1. To determine whether a data sample falls into one class or the other, we make use of decision boundaries, as introduced in section 3.6.3. A threshold is usually determined by an optimization problem, namely, we can instruct our algorithm to take such decision boundary, or boundaries, that maximize the accuracy metric, for example. Theoretically, we can build thresholds on any existing metric derived from the confusion matrix and one popular way to effectively choose a decision boundary is the receiver operating characteristics curve, abbreviated as ROC. This curve is generated by evaluating the class probabilities of a model across a range of thresholds. Namely, the true-positive rate and the false-positive rate, derived from the confusion matrix, are plotted against each other for each candidate threshold. Therefore, each instance of a confusion matrix represents one point on the ROC figure. This tool helps choose such a threshold, that maximizes the trade-off between sensitivity and specificity. Differently put, the ROC illustrates the diagnostic ability of a classifier when its discrimination threshold is changed. It can also be used as a quantitative method to assess model performance. A model able to perfectly separate two classes has a specificity and sensitivity value of 100%, which on the figure below would be denoted by the top-left corner point, (0,1) coordinates, while a completely random classifier would be on any point on the diagonal line of the figure.

The aggregate measure of performance across all possible thresholds is instead given by the area under the curve, also known as AUC. The AUC comes with two important properties, namely, it is scale-invariant, as it measures how well predictions are ranked, rather than their absolute values, and is classification-threshold-invariant because it measures the quality of the model's predictions irrespective of what classification threshold is chosen (Altman & Bland, 1994). An illustrative example of a ROC graph used for performance evaluation follows below.

Figure 11: Example of a ROC Space, Curves and AUC's for Binary Classification



Source: Kumar & Indrayan (2011).

The most important advantage of the ROC curve is that, as it represents all the possible trade-offs between the true-positive and false-positive rates, it allows us to analytically choose the desired level of sensitivity and specificity depending on our classification objective. We want to choose such a model that maximizes the chance of correctly predicting upward and downward directional changes, as this is of key importance to our trading strategy since it is based on the generated predictions. For this reason, we choose the model that maximizes the area under the curve.

On the other hand, a disadvantage of the ROC method is the absence of a guarantee, that a model is uniformly better than another because it can have an initially steep ROC curve slope above the diagonal threshold while having a lower AUC than another model. If so, in case we are interested in the region of the curve with a low false-positive rate, then the AUC would not help identify the best model (Fawcett, 2006), yet we are not exclusively concerned with this region and therefore assume that, based on the AUC statistic, we are producing reliable and effective predictions to then be used in our trading strategy.

3 EVALUATING FORECASTING PERFORMANCE

3.1 Evaluating the performance of training models

To have a clear overview of the forecasting models' performance, first, we must take a look at the metrics' results of the best training model for all the five algorithms used. For ease of comparison, we create a table where we gather the results for all the five best training models in terms of optimal value for tuning parameters, the AUC, sensitivity and specificity metrics. Additionally, we plot the changes in the AUC values as a change in the values of the underlying tuning parameters. The algorithm that maximizes the AUC metric in the training phase is the multi-layer-perceptron neural network with three hidden units. Overall, we can notice how non-linear models perform slightly better than linear ones. Considering the computation times, we would be induced to choose either the latter or the logistic model, yet we first need to analyse their performance on the test set. Below the training set summary table and AUC evolution figure.

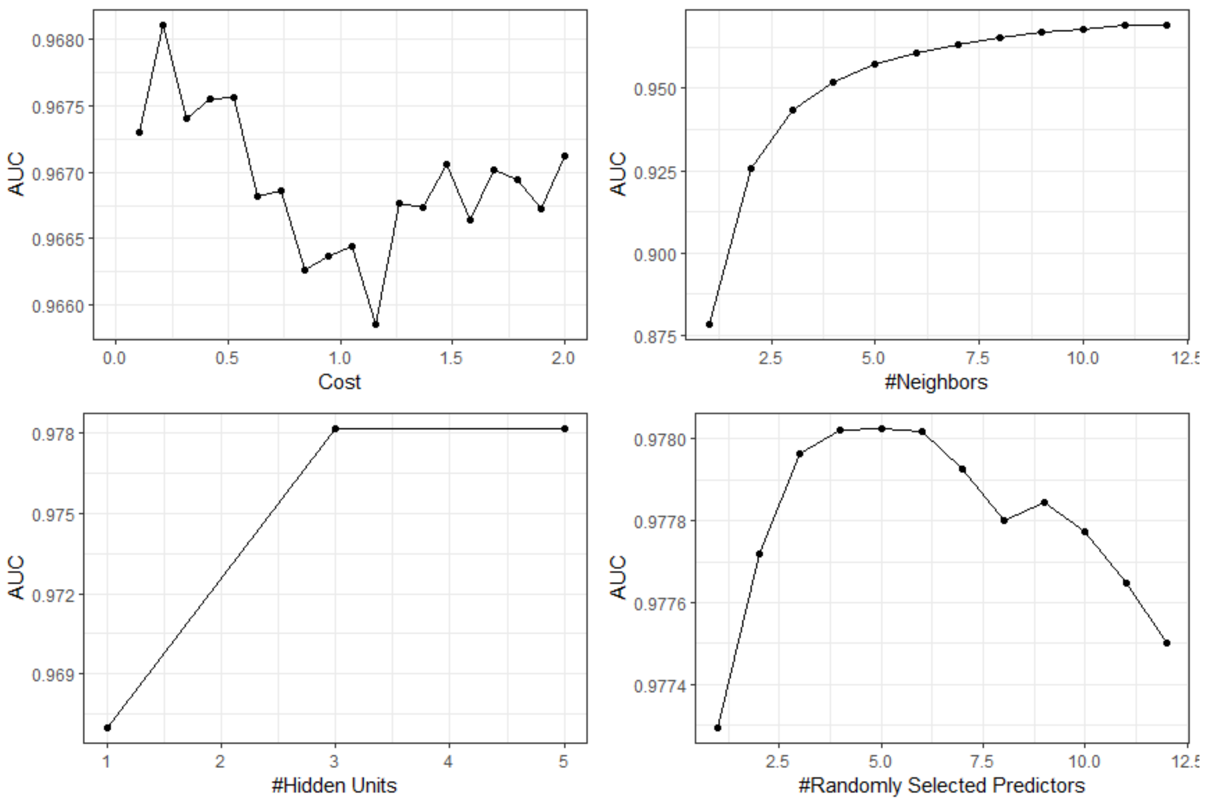
Table 4: Class Predictions and Probabilities Metrics for the Best Training Models

Model	AUC	Sensitivity	Specificity
LOG	0.967935	0.929355	0.903721
SVM	0.968110	0.930032	0.904131
KNN	0.969313	0.922920	0.892231
MLP	0.978191	0.935380	0.908322
RF	0.978027	0.933350	0.904913

Model	Tuning Parameter	Value	Computation Time
LOG	-	-	3.310714 (s)
SVM	Cost	0.2105	15.1143 (m)
KNN	K-Neighbours	12	55.45978 (s)
MLP	Hidden Units	3	2.492273 (m)
RF	Randomly Selected Predictors	5	29.19552 (m)

Source: Own work.

Figure 12: Changes in the AUC's as a Change in the Value of Tuning Parameters



Note. From the top-left to bottom-right: SVM, KNN, MLP and RF. The LOG model does not contain any tuning parameters.

Source: Own work.

3.2 Evaluating the performance of testing models

The next step is then to feed these five best-tuned models with the test set and again evaluate both the class predictions and probabilities using the same tools as above. Therefore, we provide a table summarizing all the explained class predictions and probabilities metrics for all the five best-trained models below. Unsurprisingly, the best model in the testing phase remains the MLP with the highest values for all metrics, except specificity, which is slightly lower than the LOG counterpart. Our final choice for modelling directional changes of the log-relative returns on the closing price of the S&P 500 Index is the multi-layer-perceptron algorithm with five hidden nodes.

Table 5: Class Predictions and Probabilities Metrics for the Best Testing Models

Metric - Model			
	LOG	SVM	KNN
Accuracy	0.9107	0.907	0.9025
95% CI ACC	(0.8993, 0.9211)	(0.8954, 0.9177)	(0.8908, 0.9135)
NIR	0.5297	0.5297	0.5297
P-Value (ACC > NIR)	< 0.05	< 0.05	< 0.05
Kappa	0.8206	0.8132	0.8042
Sensitivity	0.9019	0.8972	0.8893
Specificity	0.9185	0.9157	0.9143
Prevalence	0.4703	0.4703	0.4703
Balanced Accuracy	0.9102	0.9064	0.9018
AUC	0.9659	0.9649	0.9665
95% CI AUC	(0.9597, 0.9721)	(0.9586, 0.9713)	(0.9605, 0.9724)

	MLP	RF
Accuracy	0.9155	0.9107
95% CI ACC	(0.9044, 0.9257)	(0.8993, 0.9211)
NIR	0.5297	0.5297
P-Value (ACC > NIR)	< 0.05	< 0.05
Kappa	0.8308	0.8207
Sensitivity	0.9325	0.9058
Specificity	0.9003	0.915
Prevalence	0.4703	0.4703
Balanced Accuracy	0.9164	0.9104
AUC	0.9741	0.9742
95% CI AUC	(0.9692, 0.9789)	(0.9695, 0.979)

Source: Own work.

Note. NIR stands for no-information rate and ACC is short for accuracy.

3.3 Evaluating the performance of the chosen model on sequential data

As a final test of performance, we retrieve again the original S&P500 Index data attributes, compute the indicators, and pass the augmented dataset through the same transformations as in section 3, yet this time, we use data from the period 01.01.2021-30.04.2021. For this task, we are using the model that best performed on the previous test set, the MLP algorithm and analysing its performance using class predictions metrics. We can notice an overall decrease in performance compared to the training and test sets, yet we consider the results to be over the expectations.

Table 6: Class Predictions Metrics for the MLP Model for the Selected Period

Metric - Value	
Accuracy	0.7901
95% CI	(0.6854, 0.8727)
No Information Rate	0.5556
P-Value [Acc > NIR]	< 0.05
Kappa	0.5738
Sensitivity	0.7500
Specificity	0.8222
Prevalence	0.4444
Balanced Accuracy	0.7861
AUC	0.8883

Source: Own work.

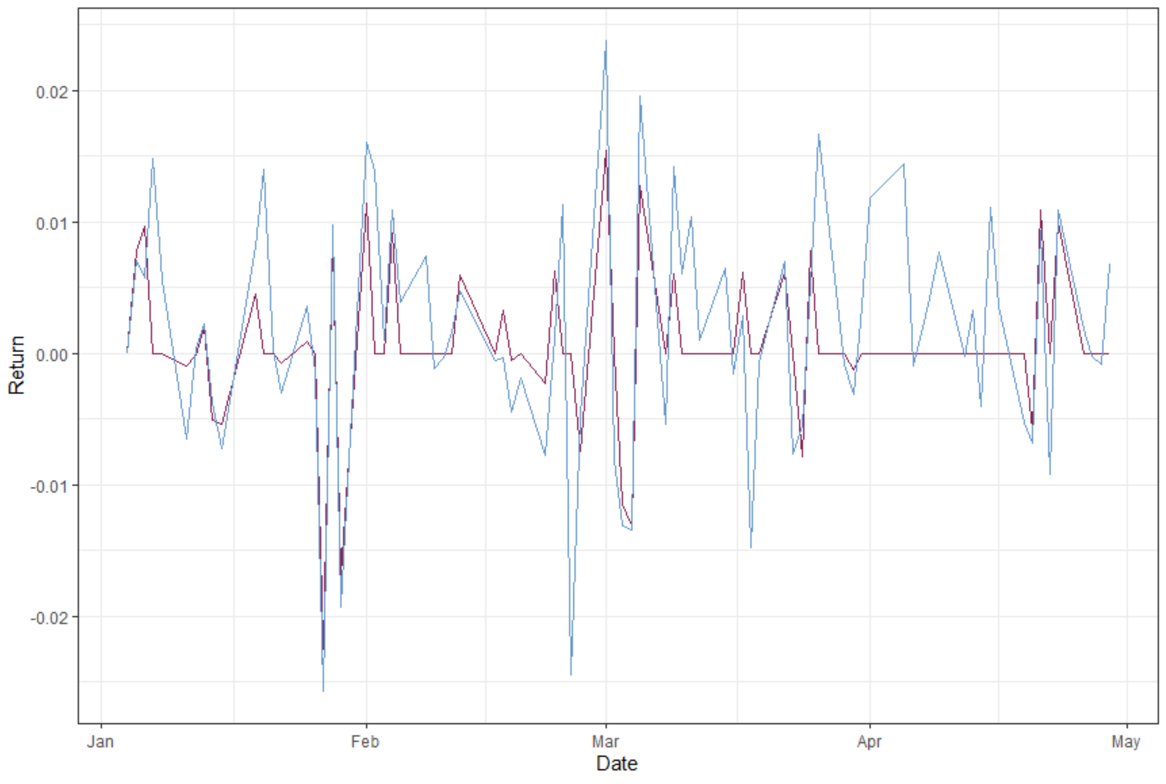
4 TRADING STRATEGY

Finally, we develop a simple trading strategy for the period 01.01.2021-30.04.2021, which is based on the generated class predictions. In a real-world application, the strategy would be executed at the beginning and the end of a market day. At the beginning of the day, predictions for the directional change are retrieved and if the index's value is projected to rise at the end of the day, we buy the security immediately and then sell it at the end of the day. On the other hand, if the value is projected to be negative, we do not enter the day trade.

As with every trade on a market, we must consider several factors before we generate a number representing our net return, which would then be compared to the net return of a passive strategy. First, we must choose an asset that represents the S&P500 Index. A reasonable choice for replicating the index would be to buy an exchange-traded fund (ETF).

This financial security in this particular context tracks the index and can be bought or sold on a stock exchange as any other regular stocks. For simplicity, we assume that our chosen ETF can perfectly replicate the value of the S&P500 Index. Second, we must consider transaction costs. These are the expenses in making any economic trade, either buying or selling when participating in a market. In a financial context, these costs include brokers' commissions and spreads, which are the differences between the dealer's buying and selling price. Additionally, we must consider the endogenous cost of an ETF, which is the expense ratio (Gastineau, 2008). By looking at the ratios of the seven most traded ETF's on the S&P500 Index, we assume that our chosen ETF has an expense ratio of 0.05 including provision costs (Investopedia, 2015). Therefore, transaction costs are of key importance for investors and traders, as they can represent a significant burden on the return on investment. Finally, we assume that the chosen ETF is sufficiently liquid for daily trading. The passive trading, or better, investment strategy consists of entering the market at the beginning of the period and exiting it at the end. As a benchmark value, we trade 10.000 USD. The net return of our simple trading strategy based on the generated predictions for the abovementioned period is 4.26%, which is significantly lower than the return of the passive investment strategy, which amounts to 13.80%. We can observe the evolution of the returns for both strategies below.

Figure 13: Relative Returns for Both Strategies Compared



Note. The active trading strategy is in red, the passive one is in blue. Displayed are the daily returns for both strategies.

Source: Own work.

CONCLUSION AND RECOMMENDATIONS

If we were asked whether is it possible to enhance the use of technical analysis with the help of a branch of artificial intelligence, then our answer would be positive. In this work, we see that it is possible to take the information given by financial technical indicators and feed it to modern learning models to forecast the direction of a financial asset class. Based on the real-world application set, we deem the results of our work satisfactory, especially when we would consider this work to be a basis upon which to build more sophisticated forecasting models and trading strategies.

Recalling the efficient market hypothesis, we dare not to take a position in its favour and neither against it. The reasons for the latter, lie in several factors, which can also be interpreted as recommendations for further research and development. First, we could experiment with forecasting models on even broader time horizons, meaning we would change the data partitioning methods. Second, we could spend additional resources on finding more informative technical indicators or more informative versions of them, as well as further explore the existing literature and hopefully land on a particular model, which significantly outperforms its peers. We could also try to change the basic underlying assumptions of log-normality and the theoretical consequent stationarity property and perhaps explore models that work with non-stationary data. Moreover, regarding the chosen trading strategy, we could create a slightly more sophisticated one, which includes the information of the predicted downward directional changes and capitalize on it by using supplementary financial instruments such as put options, for which we believe they can significantly improve the net return of our daily trading strategy. Finally, although it would be out of the scope of our thesis, we could also try to expand it and include other types of information, for example, ratios from fundamental analysis and macroeconomic indicators, and test whether we can consistently outperform the market.

REFERENCES

1. Abu-Mostafa, Y., & Atiya, A.F. (2004). Introduction to financial forecasting. *Applied Intelligence*, *6*, 205-213. Retrieved October 10, 2020 from https://doi.org/10.1007/BF00126626

2. Altman, D., Bland, J. (1994). Diagnostic Tests 3: Receiver Operating Characteristic Plots. *British Medical Journal*, *309*(6948), 188. doi: 10.1136/bmj.309.6948.188

3. Bender, E. A., Williamson, S. G. (2010). *Lists, Decision and Graphs: With an Introduction to Probability*. Retrieved October 10, 2020 from https://cseweb.ucsd.edu/~gill/BWLectSite/Resources/LDGbookCOV.pdf

4. Biau, G., & Scornet, E. (2016). A random forest guided tour. *TEST*, *25*(2), 197–227. doi:10.1007/s11749-016-0481-7

5. Blessie, E. C., & Karthikeyan, E. (2012). Sigmis: A Feature Selection Algorithm Using Correlation Based Method. *Journal of Algorithms & Computational Technology*, *6*(3), 385–394. doi:10.1260/1748-3018.6.3.385

6. Box, G. E. P., & Cox, D. R. (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, *26*(2), 211–243. doi:10.1111/j.2517-6161.1964.tb00553.x

7. Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, *14*(6), 1506–1518. doi: 10.1109/TNN.2003.820556

8. Chambers, J., Cleveland, W., Kleiner, B., & Tukey, P. (2017). *Graphical Methods for Data Analysis* (1st ed.) Boca Raton: Chapman and Hall. https://doi.org/10.1201/9781351072304

9. Chang, P.C., Fan, C. Y., & Lin, J. L. (2011). Trend discovery in financial time series data using a case based fuzzy decision tree. *Expert Systems with Applications*, *38*(5), 6070–6080. doi:10.1016/j.eswa.2010.11.006

10. Chen, Y., & Hao, Y. (2017). A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, *80*, 340–355. doi:10.1016/j.eswa.2017.02.044

11. Chi-Jie, L., Tian-Shyug, L., & Chih-Chou, C. (2009). Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, *47*(2), 115-125. https://doi.org/10.1016/j.dss.2009.02.001

12. Cohen, J. (1960). A Coefficient of Agreement for Nominal Data. *Educational and Psychological Measurement*, *20*, 37–46. https://doi.org/10.1177/001316446002000104

13. Colby, R. W. (2002). *The Encyclopedia of Technical Market Indicators (2nd ed.)*. New York: McGraw-Hill.

14. Cristianini, N., & Ricci, E. (2008). Support Vector Machines. *Encyclopedia of Algorithms,* 928–932. doi:10.1007/978-0-387-30162-4415

15. Curtis, F. E., & Scheinberg, K. (2017). Optimization Methods for Supervised Machine Learning: From Linear Models to Deep Learning. *The Operations Research Revolution,* 89–113. doi:10.1287/educ.2017.0168

16. Cybenko, G. (1989). Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, *2* (4), 303-314. https://doi.org/10.1007/BF02551274

17. Di Lorenzo, R. (2012). *How to Make Money by Fast Trading. Perspectives in Business Culture* (1st ed.). Italy: Springer.

18. Dobson, A. J. (2002). *An Introduction To Generalized Linear Models* (2nd ed.). New York: Chapman & Hall/CR.

19. Dohoo, I. R., Ducrot, C., Fourichon, C., Donald, A., & Hurnik, D. (1997). An overview of techniques for dealing with large numbers of independent variables in epidemiologic studies. *Preventive Veterinary Medicine*, *29*(3), 221–239. doi:10.1016/s0167-5877(96)01074-4

20. Dwyer, W. D. (2005). *Examples Of Overfitting Encountered When Building Private Firm Default Prediction Models*. Retrieved December 18, 2020 from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.4174&rep=rep1&type=pdf

21. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognitions Letters, 27*(8), 861–874. doi:10.1016/j.patrec.2005.10.010

22. Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669. doi:10.1016/j.ejor.2017.11.054

23. Garman, M., B., & Klass, M., T. (1980). On the Estimation of Security Price Volatilites from Historical Data. *The Journal of Business*, *53*(1), 66-78.

24. Gastineau, G. L. (2008). *Exchange-Traded Funds. Handbook of Finance* (1st ed.). New York: Wiley.

25. Gorunescu, F. (2011). Classification Performance Evaluation. *Data Mining*, 319–330. doi:10.1007/978-3-642-19721-5_6

26. Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN Model-Based Approach in Classification. *Lecture Notes in Computer Science*, 986–996. doi:10.1007/978-3-540-39964-3_62

27. H. M. Hassan, A. M. Negm, M. Zahran, & O. Saavedra. (2015). *Assessment of Artificial Neural Network for Bathymetry Estimation Using High Resolution Satellite Imagery in Shallow Lakes: Case Study El Burullus Lake*. Retrieved February 16, 2021 from https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network_fig4_303875065

28. Hill, A. (2019). Finding Consistent Trends with Strong Momentum: RSI for Trend-Following and Momentum Strategies. *SSRN*. Retrieved February 18, 2021 from http://dx.doi.org/10.2139/ssrn.3412429

29. Ho, T. K. (1998). *The random subspace method for constructing decision forests*. doi:10.1109/34.709601

30. Hornik, K. (1991). Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, *4*(2), 251–257. doi:10.1016/0893-6080(91)90009-T

31. Ihaka, R., & Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, *5*(3), 299–314. doi:10.1080/10618600.1996.10474713

32. Investopedia. (2015). *The Top 7 ETFs For Day Trading*. Retrieved April 18, 2021 from https://www.investopedia.com/articles/investing/110315/top-7-etfs-day-trading.asp

33. Irizarry, R. (2019). *Introduction to Data Science: Data Analysis and Prediction Algorithms with R* (1st ed.). London: Chapman & Hall/CRC.

34. James, F. E. (1968). Monthly Moving Averages–An Effective Investment Tool? *The Journal of Financial and Quantitative Analysis*, *3*(3), 315. doi:10.2307/2329816

35. Jiao, Y., & Jakubowicz, J. (2017). Predicting stock movement direction with machine learning: An extensive study on S&P 500 stocks. *2017 IEEE International Conference on Big Data (Big Data)*, 4705–4713. doi:10.1109/BigData.2017.8258518

36. Kiseon Kim, & Shevlyakov, G. (2008). Why Gaussianity? *IEEE Signal Processing Magazine*, *25*(2), 102–113. doi:10.1109/msp.2007.913700

37. Klinker, F. (2010). Exponential moving average versus moving exponential average. *Mathematische Semesterberichte*, *58*(1), 97–107. doi:10.1007/s00591-010-0080-8

38. Krollner, B., Vanstone, B., & Finnie, G. (2010). Financial time series forecasting with machine learning techniques: A survey. *In Proceedings of the 18th European Symposium on Artificial Neural Networks (ESANN2010): Computational Intelligence and Machine Learning*, 25-30. Retrieved November 10, 2020 from https://pure.bond.edu.au/ws/files/27498056/Financial_time_series_forecasting_wit h_machine_learning_techniques.pdf

39. Kuhn, M., Johnson, K. (2013). Applied predictive modeling (1st ed.). New York: Springer.

40. Kumar, R., & Indrayan, A. (2011). Receiver operating characteristic (ROC) curve for medical researchers. *Indian Pediatrics*, *48*(4), 277–287. doi:10.1007/s13312-011-0055-4

41. Loong, T.-W. (2003). Understanding sensitivity and specificity with the right side of the brain. *BMJ, 327*(7417), 716–719. doi:10.1136/bmj.327.7417.716

42. Malkiel, B. G. (1999). *A random walk down Wall Street: Including a life-cycle guide to personal investing* (1st ed.). New York: Norton.

43. Mansfield, E. R., & Helms, B. P. (1982). Detecting Multicollinearity. *The American Statistician*, *36*(3), 158. doi:10.2307/2683167

44. Mishra, P., Pandey, C. M., Singh, U., Gupta, A., Sahu, C., & Keshri, A. (2019). Descriptive statistics and normality tests for statistical data. *Annals of cardiac anaesthesia*, *22*(1), 67–72. Retrieved October 19, 2020 from https://doi.org/10.4103/aca.ACA_157_18

45. Murtagh, F. (1991). *Multilayer perceptrons for classification and regression*. doi:10.1016/0925-2312(91)90023-5

46. Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the Equity Risk Premium: The Role of Technical Indicators. *Management Science*, *60*(7), 1772–1791. doi:10.1287/mnsc.2013.1838

47. Nornadiah, M. R. &, Bee, Y. (2011). Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling Tests. *Journal of Statistical Modeling Analytics*, (2). Retrieved March 26, 2021 from

https://www.researchgate.net/publication/267205556_Power_Comparisons_of_Shapiro-Wilk_Kolmogorov-Smirnov_Lilliefors_and_Anderson-Darling_Tests

48. Proietti, T., & Lütkepohl, H. (2013). Does the Box–Cox transformation help in forecasting macroeconomic time series? *International Journal of Forecasting*, *29*(1), 88–99. doi:10.1016/j.ijforecast.2012.06.001

49. Rogers, L. C. G., Satchell, S., & Yoon, Y. (1994). Estimating the Volatility of Stock Prices: A Comparison of Methods That Use High and Low Prices. *Applied Financial Economics*, *4*(3), 241-247. doi: 10.1080/758526905

50. S. Zhang, X. Li, M. Zong, X. Zhu, & D. Cheng. (2016). *Learning k for kNN Classification*. Retrieved December 18, 2020 from https://dl.acm.org/doi/pdf/10.1145/2990508

51. Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing,* 106181. doi:10.1016/j.asoc.2020.106181

52. Shynkevich, Y., McGinnity, T. M., Coleman, S. A., Belatreche, A., & Li, Y. (2017). Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*, *264*(1), 71–88. doi:10.1016/j.neucom.2016.11.095

53. Silva, A., Neves, R., & Horta, N. (2015). A hybrid approach to portfolio composition based on fundamental and technical indicators. *Expert Systems with Applications*, *42*(4), 2036–2048. doi:10.1016/j.eswa.2014.09.050

54. Timmermann, A., Granger, C. W. J. (2004). *Efficient market hypothesis and forecasting*. doi:10.1016/s0169-2070(03)00012-8

55. Ţiţan, A. G. (2015). The Efficient Market Hypothesis: Review of Specialized Literature and Empirical Research. *Procedia Economics and Finance, 32,* 442–449. doi:10.1016/S2212-5671(15)01416-1

56. Wang, L. (2005). *Support Vector Machines: Theory and Applications* (1st ed.). New York: Springer.

57. Wilder, J. W. (1978). *New Concepts in Technical Trading Systems* (1st ed.). Winston-Salem: Hunter Publishing Company.

58. Witten, H. I., Eibe, F., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Burlington: Morgan Kaufmann.

59. Yeo, I., Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*, (4), 954–959. https://doi.org/10.1093/biomet/87.4.954

60. Yıldırım, D.C., Toroslu, I.H. & Fiore, U. (2021). Forecasting directional movement of Forex data using LSTM with technical and macroeconomic indicators. *Financial Innovation*, *7* (1). Retrieved November 11, 2020 from https://doi.org/10.1186/s40854-020-00220-2

61. Zheng, A., & Casari, A. (2018). Feature Engineering for Machine Learning (1st ed.). Newton: O'Reilly.

62. Zhong, Y., & Enke, D. (2019). Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financial Innovation*, *5*(1), 4–11. doi:10.1186/s40854-019-0138-0

APPENDICES

Appendix 1: Povzetek (Summary in Slovene language)

V tem delu prikazujem, kako lahko uporabimo tehnično analizo v kombinaciji s sodobnimi modeli napovedovanja s področij strojnega in globokega učenja za ustvarjanje dodatnih informacij, ki jih lahko uporabimo v procesu odločanja, kot je, na primer, strategija trgovanja, tako da začetni problem napovedovanja časovnih vrst preoblikujemo v nalogo klasifikacije le teh. Najprej pridobim javno dostopne podatke o denvnih cenah in obsegu indeksa S&P 500, jih razširim z izračunom množice tehničnih kazalnikov in jih uporabim kot vhodne podatke za modele s končnin ciljem ustvarjenja binarne napovedi dnevne spremembe smeri logaritemske donosnosti indeksa. Celoten pristop k modeliranju in vrednotenju napovedi temelji na običajnih praksah strojnega in globokega učenja. Nato, pridobljene napovedi uporabim za testiranje preproste strategije trgovanja in to primerjam s pasivno strategijo "kupi in drži". Ugotovil sem, da je v tem primeru najboljša kombinacija za izboljšanje uporabe tehnične analize združevanje tehničnih kazalnikov z večplastno perceptronsko nevronsko mrežo. Čeprav je neto donos moje preproste dnevne strategije trgovanja nižji od pasivne strategije "kupi in drži", verjamem, da je moj poskus lahko podlaga za natančnejše testiranje omenjenih pristopov in da bo to spodbuda bodoče radovednosti za nadaljnji razvoj.

Appendix 2: Abstract

In this work, we show how we can use technical analysis combined with modern prediction models from the fields of machine and deep learning to generate additional information that can be applied to a decision-making process, such as a trading strategy, by reformulating a time-series prediction problem into a time-series classification task. We retrieve publicly available data about the S&P 500 Index daily prices and volume, expand it by computing a series of technical indicators and use them as inputs to our learning models to generate a binary prediction of the index's daily log-relative-return directional change. The entire forecasting modelling and evaluation approach is based on common machine and deep learning practices. Ultimately, we use the generated predictions to test a simple trading strategy and compare it to a passive buy-and-hold strategy. We find that, in our case, the best combination for enhancing the use of technical analysis is to pair technical indicators with a multi-layer-perceptron neural network. Although the net return of our simple daily trading strategy is lower than a buy-and-hold passive strategy, we believe that our experiment forms the basis for more rigorous testing of the aforementioned approaches and spur the curiosity for further development.