Ca' Foscari
University
of Venice

Master's Degree in

Data Analytics for Business and Society

Final Thesis

Predicting Stock Returns of Italian Utility Industries Using Random Forest Algorithms

Supervisor
Ch. Prof. Marco Corazza

Graduand
Marco Bruttocao
Matriculation Number 858067

Academic Year
2021 / 2022

Abstract

Automated stock trading systems based on machine learning techniques have been an extremely active research area in the last decade among both academics and industry professionals. The final goal is to construct systems that are better than or at least as good as their human counterparts in recognizing investment opportunities that could lead to significant profit. Several algorithms have been evaluated in performing this task, such as artificial neural networks, support vector machines, decision trees, genetic algorithms, and a wide variety of hybrid approaches (Rouf, et al., 2021). Drawing on the scientific literature on automated stock trading systems based on machine learning techniques, this thesis aims to evaluate the ability of random forest algorithms in predicting whether the expected return of an investment in stocks will be positive or negative at the end of a hypothetical five-day trading window. Specifically, the final goal is to build an effective trading tool calibrated for the Italian utility and energy sectors. Therefore, the stocks taken into consideration include all the utility and energy companies listed in the FTSE MIB index providing oil, gas, and electricity. The time frame from which the data were extracted ranges from January 1st, 2016, to December 31st, 2021. One of the most widely used approaches is to take advantage of technical analysis to build an exhaustive set of predictors and enhance the model predictive capabilities. In the first instance, several technical indicators are generated based on the historical data of stock trading price and volume, later the overall performance of the model, and consequently of the trading system are evaluated. Finally, further steps were taken in the direction of providing the algorithm with not only market information, but also general Italian economy health indicators release dates, holidays, specific weekdays, and climate change awareness events dates. Final results are promising, meaning that the automated trading system is able to improve the profitability of investments in the Italian utility sector. Specifically, at the end of the testing period, even the simpler trading system provides superior results when compared to traditional buy-and-hold strategies while the “enhanced” trading system is able to achieve the excellent result of almost doubling the outcomes with respect to the same strategies.

Contents

List of Figures

List of Tables

List of Equations

Introduction

As long as stock markets continuously evolve and become increasingly interactive, forecasting of financial performance and comprehension of trading patterns play a crucial role in achieving profitable investments. In such an extremely dynamic and complex environment traders and investors are constantly looking for tools that are able to maximize efficiency and at the same time minimize risk. There is an increasing need to find scalable and integrated solutions that can rapidly adapt to every market shift, and automated trading systems are increasingly developed to serve this purpose. These systems consent to respond immediately to changing market conditions since they are able to open and close trading positions as soon as predefined trading criteria are met. Moreover, automated trading systems do not suffer the biases that individuals tend to show, such as overconfidence, negativity bias, and many others. On the other hand, it is important to remember that automated trading systems can also have pitfalls such as the lack of the judgmental aspect that is typical of human beings. The aim of this thesis is to build an automated trading system that is able to provide consistent positive outcomes and that can be employed to develop a trading strategy, whether it is deployed as such or used to support traders and investors in their choices. In doing that, random forest algorithms are explored evaluating their ability in predicting whether the expected return of an investment in stocks will be positive or negative at the end of a hypothetical trading window when based simply on technical indicators and a selection of additional macroeconomic indicators.

Chapter I: Theoretical Background

Chapter I addresses the necessity of providing a comprehensive overview of the methods and techniques at the basis of the development of an automated trading system based on technical analysis and random forest algorithms. As a starting point, the first section summarizes the main challenges that arise when facing the forecasting problem in the financial markets, presenting several approaches, and outlining their main strengths and pitfalls. Thanks to this preliminary introduction it is possible to deeply understand what challenges have to be faced when developing an intelligent system that is successful in exploiting financial historical data and patterns. The second section deepens the concept of technical analysis, as the trading system that is implemented in this essay will be based on technical indicators to forecast the direction of the stock prices. A historical overview of this trading discipline will be provided, starting from the foundations up to recent developments. The focus of the third section will be on providing the basic concepts that are at the base of the random forest algorithms implemented in this work, giving an overview of the inner working of decision trees and supervised learning methods in general. Section four provides a schematic overview on the cross-validation resampling technique as it will be employed multiple times to find the optimal set of model hyperparameters and to assess the predictive performance in both training and testing phases. Finally, section five describes the intuitions behind the automated trading system and its main properties, illustrating in detail problem-specific challenges and proposed solutions.

1.1 Financial Markets Forecasting: an Overview

Analyzing stock market behavior is undoubtedly extremely challenging because of the market dynamic, nonstationary, noisy, and chaotic nature (Y. Abu-Mostafa, 1996). As a matter of fact, stock markets are affected by many highly interrelated factors that include economic, political, psychological, and company-specific variables (Zhong, 2017). Two operational techniques that have been developed to make decisions in financial markets are technical analysis and fundamental analysis. The basic assumption of technical theories is that history tends to repeat itself, in other terms, past patterns of price behavior will tend to recur in the future (Fama E. F., 1965).

Therefore, these techniques attempt to use knowledge of the past behavior of a price series to predict its future behavior. An in-depth overview of the main instruments deployed in technical analysis will be given in section 1.2, while Chapter III describes the actual indicators considered in this investigation, their mathematical formulations, and the intuition behind their implementation. On the other hand, fundamental analysis is mainly based on three essential aspects (Hu, 2015). The first aspect involves an accurate analysis of the indicators that may affect the future profit and performance of a company, the second aspect regards an in-depth industry analysis to estimate the value of the company based on the overall industry and the third aspect consists of careful company analysis in order to evaluate its current operation and financial status to estimate its internal value. The final aim of fundamental analysis is to define an evaluation that is comparable with a security current price to determine whether the security is undervalued or overvalued, developing investment strategies accordingly. Several theories regarding stock markets have been developed over the years, with the aim of explaining the nature of stock markets or stating whether and by what extent the markets can be beaten i.e., make investment returns that outperform the overall market average. The most popular and debated theory is the Efficient Market Hypothesis (EMH) which states that at any point in time, the market price of the stock incorporates all information about that stock (Fama E. , 1970) (C. Jensen, 1978). A direct implication of EMH is that there is no possibility to beat the market consistently since market prices only react to new information. In contrast with this theory, many financial economists and statisticians believe that stock prices are at least partially predictable, emphasizing psychological and behavioral elements of stock-price determination and volatility (Peter F. Christoffersen, 2006). According to this tendency, it can be stated that future stock prices are somewhat predictable based on past stock price patterns as well as certain fundamental valuation metrics (Suryoday Basak, 2019). Lastly, recent advancements in stock analysis and prediction techniques can be traced back into four categories, as summarized in Figure 1. These categories are respectively statistical methods, pattern recognition, machine learning (ML), and sentiment analysis (Dev Shah, 2019). Statistical methods are based on the idea that chronological collection of observations such as daily prices of stocks can be identified as time series. As a direct consequence, a wide range of models as Auto-Regressive Moving Average (ARMA) and Auto-Regressive Integrated Moving Average (ARIMA) can be implemented for forecasting purposes, due to their use of time series as input variables.

Machine Learning (ML) is a branch of artificial intelligence (AI) that aims to use data and algorithms to imitate the way that humans learn. According to the UC Berkeley School of Information (Berkeley, 2020), the typical supervised machine learning algorithm can be decomposed into three main components:

- A decision process: it is the first step and consists of retrieving and preparing the data to recognize what kind of patterns the algorithm may find, producing the first "guesses".

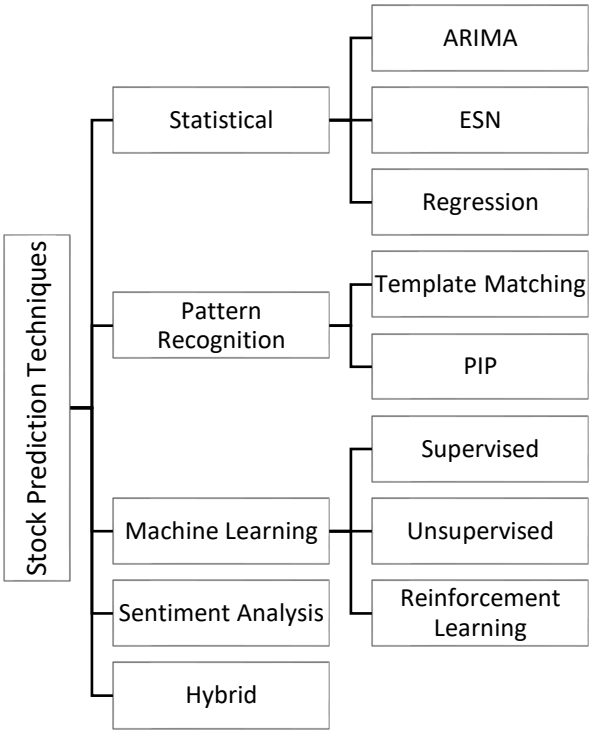
- An error function: it is a method that measures the goodness of the "guesses" by comparing them to known examples (in the case of their availability).

- An updating or optimization process: it is the last step in which the algorithm becomes aware of the wrong "guesses" and updates the decision process to provide better final decisions.

Moreover, ML methods can be grouped into several families that are supervised learning methods, unsupervised learning, semi-supervised learning, and reinforcement learning. The distinctive feature of supervised learning is to provide the machine learning algorithm with a full set of labeled data. With the term labeled it is meant that each example in the training dataset is assigned a tag with the answer the algorithm should come up with. Classification and regression problems are typical examples of supervised learning. On the other hand, in unsupervised learning problems, the models are built on datasets without explicit instructions on what to do with it, and observations are evaluated without providing any kind of label. Unsupervised learning examples are clustering, anomaly detection, and association problems. Semi-supervised learning lies in between supervised and unsupervised learning, meaning that the dataset contains labeled and unlabeled data. This method is useful when extracting important features from the data is difficult, and labeling examples is a time-intensive task. Finally, reinforcement learning is based on the idea that the agent learns an optimal, or nearly optimal, policy that maximizes a predefined "reward function". Those algorithms enable an agent to learn by trial and error using feedback from its own actions and experiences through an action-reward feedback loop.

Several algorithms have been used in stock price direction prediction ranging from simpler techniques such as the single decision tree, discriminant analysis, and naïve Bayes that have been later replaced by better-performing algorithms such as Random Forest, logistic regression, and neural networks (Ballings, Van den Poel, Hespeels, & Gryp, 2015). Pattern recognition takes advantage of machine learning algorithms to automatically recognize patterns and regularities in data. In stock market applications some important families of patterns are continuation and reversal, the first seeking for trend conservation while the second looking for patterns that anticipate changes in direction. Finally, sentiment analysis aims to provide psychology-based indicators to provide information that may precede market fluctuations assuming that stock prices are being influenced by emotion rather than rational decision making. Common techniques in this field involve analyzing social media platforms as well as textual information with natural language processing algorithms. The automated trading system proposed in this research belongs to the supervised machine learning family and takes advantage of technical analysis and further environmental information to predict the direction of the stock market.

Figure 1 – Taxonomy of Stock Prediction Techniques



1.2 Technical Analysis

Technical analysis is considered by many to date back to the 1800s originated with the work of Charles Dow. Technical analysis techniques for discovering hidden relations in stock returns range from extremely simple to quite elaborate, allowing analysts to make profits from changes in the psychology of the market (Pring, 2014):

> "*The technical approach to investment is essentially a reflection of the idea that prices move in trends which are determined by the changing attitudes of investors toward a variety of economic, monetary, political, and psychological forces. Since the technical approach is based on the theory that the price is a reflection of mass psychology ("the crowd") in action, it attempts to forecast future price movements on the assumption that crowd psychology moves between panic, fear, and pessimism on one hand and confidence, excessive optimism, and greed on the other.*"

Nowadays, three general assumptions are generally accepted by practitioners and constitute the foundation of every further construct (Murphy, 1999):

1. Market Action Discounts Everything: this assumption implies that anything that may affect the price fundamentally, politically, or psychologically, is reflected in the price of that market. As a result of this assumption, the study of price action is all that is required to obtain information on future trends.

2. Prices Move in Trends: The purpose of charting the price action of a market is to identify trends in the early stages of its development. Moreover, as a general rule of thumb, a trend motion is more likely to continue than to reverse.

3. History Repeats Itself: Chart patterns that have been previously identified and categorized may reveal the bullish or bearish market trends. Supposing that these patterns have given useful trading signals in the past, it can be fairly assumed that they will continue to provide those signals also in the future.

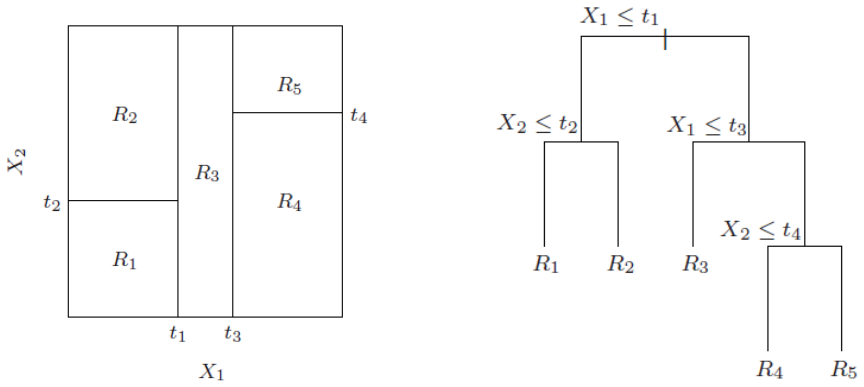
From a practical point of view, major brokerage firms are used to publish technical commentary on the market and several advisory services are based on technical analysis. Moreover, a wide range of patterns and signals have been developed by researchers to support trading, many indicators are focused on identifying the actual market trend, while others are built to determine the strength of a trend and the likelihood of its continuation.

Finally, indicators are usually classified based on their main characteristics and purposes. The main families of indicators consist of price trends, volume and momentum indicators, oscillators, and moving averages. A comprehensive description of the technical indicators implemented to build the automated trading system will be given in Chapter III since it is necessary to provide an exhaustive motivation for their use and consequently their mathematical formulation.

1.3 CART and Random Forest Algorithms

In the framework of supervised learning, decision trees can be defined as hierarchical models where the local region is identified in a sequence of recursive splits. Decision trees are composed of internal decision nodes, in which is implemented a test function with discrete outcomes intended for labeling the branches and terminal leaves. The basic training principle of decision trees is the recursive partitioning of the feature space using a tree structure, where each child node is split until nodes that contain samples of a single class are achieved or different conditions are met, as shown in Figure 2. A node that only contains a single class is homogeneous and hence it is considered entirely pure. In the proposed example (Trevor Hastie, 2009) the first split i.e., the root node, is performed according to the condition $X_1 \leq t_1$. If the observed value lies below this threshold, the root left child node is reached, on the other hand, if it lies above the threshold the root right child node is reached. Following this condition, the feature space is divided into two partitions: $L_X = \{X|X_1 \leq t_1\}$ and $R_X = \{X|X_1 > t_1\}$; this is called binary split. Then this process is repeated until no further splits are performed.

Figure 2 – Feature Space Partitioning



The goodness of split when dealing with classification trees is given by an impurity measure, defining as pure a split in which for all branches, all the instances choosing a branch belong to the same class. A common function to measure impurity is entropy, but also other functions can be used such as misclassification error.

The application of classification and regression trees (CART) along with random forest algorithms in the field of financial market prediction has been deeply explored by several scholars and researchers in the past two decades (Muh-Cherng Wu, 2006) (Ash Booth, 2014) (Alya Al Nasseri, 2015) (Mehar Vijha, 2020) (Pavan Kumar Illa, 2022). It is interesting to notice that those machine learning algorithms have proved to improve efficiencies by 60-86 percent when compared to the past methods such as ARMA or ARIMA when applied to time series analysis (Li, 2017). Classification and Regression Trees algorithm (Leo Breiman, 1984) consists of an algorithm for building a decision tree based on Gini's impurity index as a splitting criterion. Gini impurity represents the likelihood that a randomly selected example would be incorrectly classified by a specific node, and it can be computed as shown in Equation 1:

$$H_{(Q_m)} = \sum_k p_{mk}(1 - p_{mk}), \tag{1}$$

where:

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I_{(y=k)}$$

is the proportion of class k observations in node m. The Gini's index ranges from 0 to 1, where 0 or 1 indicate that all the elements belong to a certain class, or only one class exists, while a value of $p_{mk} = 0.5$ indicates that all the elements are randomly distributed across the various classes. The pseudocode of the CART algorithm is given in Figure 3. Pseudocode can be defined as a technique used to describe the distinct steps of an algorithm in an easy-to-understand form. Moreover, even if it is a syntax-free description, it provides a full description of the algorithm logic.

Figure 3 – CART Algorithm Pseudocode

```
GenerateTree(χ)
    If NodeEntropy(χ)<θ_I
        Create leaf labeled by majority class in χ
        Return
    i ← SplitAttribute(χ)
    For each branch of x_i
        Find χ_i falling in branch
        GenerateTree(χ_i)

SplitAttribute(χ)
    MinEnt ← MAX
    For all attributes i = 1, ..., d
        If x_i is discrete with n values
            Split χ into χ_1, ..., χ_n by x_i
            e ← SplitEntropy(χ_1, ..., χ_n)
            If e < MinEnt, MinEnt← e; bestf ← i
        Else x_i is numeric
            For all possible splits
                Split χ into χ_1, ..., χ_n on x_i
                e ← SplitEntropy(χ_1, ..., χ_n)
                If e < MinEnt, MinEnt← e; bestf ← i
    Return bestf
```

Source: Introduction to Machine Learning (4th ed.)

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently from the feature space and with the same distribution for all trees in the forest (Breiman, Random Forests, 2001). The main idea consists in having a large number of uncorrelated models i.e., several decision trees, which combined predictions are able to outperform the prediction of any of the individual constituent predictors models i.e., a single decision tree. Random forests ensure that the behavior of each tree is few correlated with the behavior of any of the other trees taking advantage of two stratagems: bootstrap aggregation (bagging) and feature randomness.

To deeply understand the first stratagem, a clear definition of bootstrap should be given. Bootstrapping is a resampling method and it consists of sampling with replacement from a single dataset to create many simulated samples that have the same size as the original one. This means that some samples may be represented multiple times in the bootstrap sample while others may not be selected at all. These samples are usually referred to as "out-of-bag" samples. At this point bagging predictors can be defined as a method for generating multiple versions of a predictor and using these to get an aggregated predictor, keeping in mind that the vital element consists of ensuring that all these versions are slightly different one from the other thanks to the bootstrap samples on wich they are constructed (Breiman, Bagging Predictors, 1996). The bagging algorithm consists in mainly three steps:

- Bootstrapping: Generating several subsets of the original training dataset by selecting data points randomly and with replacement.

- Parallel training: Several decision trees are trained in parallel on different bootstrap datasets.

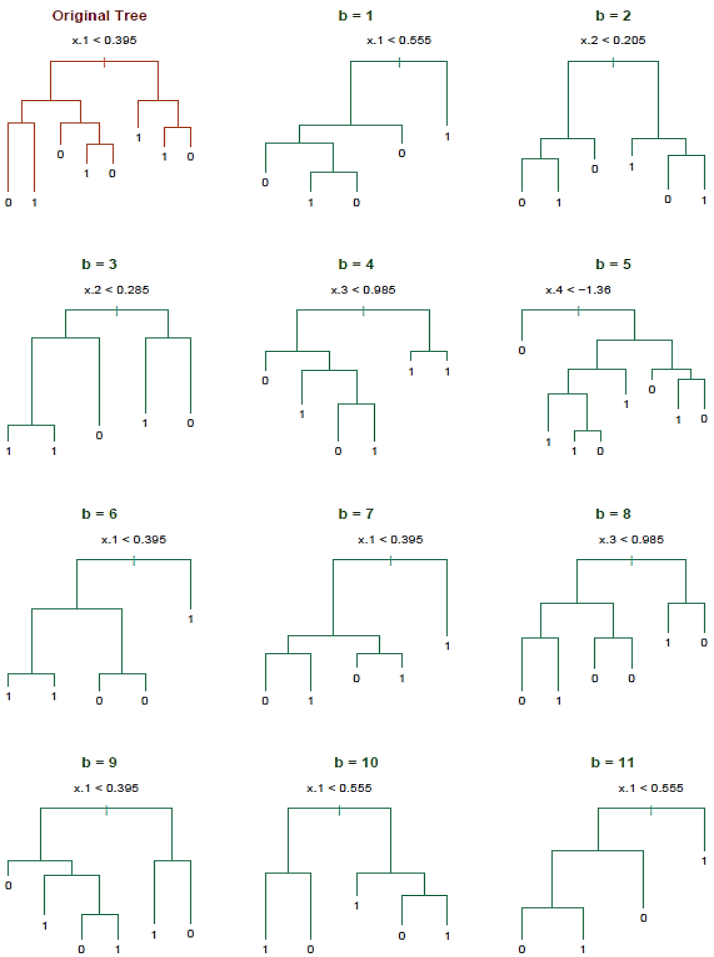
- Aggregation: Finally, depending on the task that can be regression or classification, an average or a majority of the predictions is taken. In the case of regression, an average is computed from all the outputs predicted by the individual classifiers. In the case of classification problems, the class with the highest number of votes is accepted (majority voting).

To clarify this process an example can be useful (Trevor Hastie, 2009). A sample is generated from a standard Gaussian distribution and a response Y is generated according to:

$$Pr(Y = 1|x_1 \leq 0.5) = 0.2, \qquad Pr(Y = 1|x_1 > 0.5) = 0.8\,. \qquad (2)$$

The experiment consists of fitting classification trees to the entire training sample and several bootstrap samples. Figure 4 shows the tree built using all the observations i.e., the "original tree" and eleven trees grown on bootstrap samples. It can be noticed that the trees are all different, with different splitting features i.e., the features that minimize the impurity at each split, and cutpoints i.e., the threshold value chosen to perform the split. This procedure dramatically reduces the variance of unstable procedures like CART, leading to improved prediction.

Figure 4 – Bagging Trees on a Simulated Dataset



Source: The Elements of Statistical Learning (2nd ed.)

Moving to feature randomness, it consists in randomly selecting the features that can be used by each tree among all the possible features in the training phase. In a single decision tree, at each split every possible feature is considered when computing the Gini's impurity index that ensures the best split. In contrast, each tree in a random forest is allowed to consider only a random subset of features. It can be stated that the generalization error of a random forest i.e., the measure of how accurately it predicts outcome values for previously unseen data, decreases with the generalization error of the individual trees, and with the correlation between trees (Breiman, Random Forests, 2001). Increasing the number of features trees can use in the training phase with respect to the maximum number of available features also increases the correlation between the single predictors, consequently increasing the generalization error. The optimal number of features to consider (m) depends on the problem and for this reason, it is treated as a tuning parameter. Recommended default values are $m = p/3$ for regression problems and $m = \sqrt{p}$ for classification problems, where p is the total number of features (Trevor Hastie, 2009). To summarise, in a random forest, trees are not only trained on different sets of data – thanks to bagging – but also use different features to make decisions, ensuring that the principle of the wisdom of the crowd[1] is adopted.

1.4 Cross-Validation Resampling Technique

Whenever any machine learning model is built, it is extremely important to assess if it can be considered stable. In fact, a stable model is able to perform well on unseen data, it is consistent and provides accurate predictions on a wide range of input data. Several techniques can be adopted to test the stability and reliability of a specific machine learning model and one of them is cross-validation. The aim of cross-validation is to provide an efficient way to divide data into segments that can be used to train and test a model on different iterations. The simplest approach consists in dividing the data randomly into two portions, one used to learn or train a model and the other used to validate the model.

[1] The "wisdom of the crowd" phenomenon refers to the finding that the aggregate of a set of proposed solutions from a group of individuals performs better than the majority of individual solutions. Wisdom of the crowd phenomenon might be broadly applicable to problem-solving and decision-making situations that go beyond the estimation of single numbers (Yi, 2012).

This approach may lead to high variability in the results since the performance of the considered model vary depending on the subset of data chosen for train and test. The most widespread approach to reducing variability consists in performing many rounds of cross-validation using many different partitions, and averaging the results. This technique is known as K-fold cross-validation, and consists of the following steps:

1. Considering the K^{th} part of the data, a model is fit to the other $K - 1$ parts.

2. The prediction error of the fitted model is computed when predicting the K^{th} part of the data.

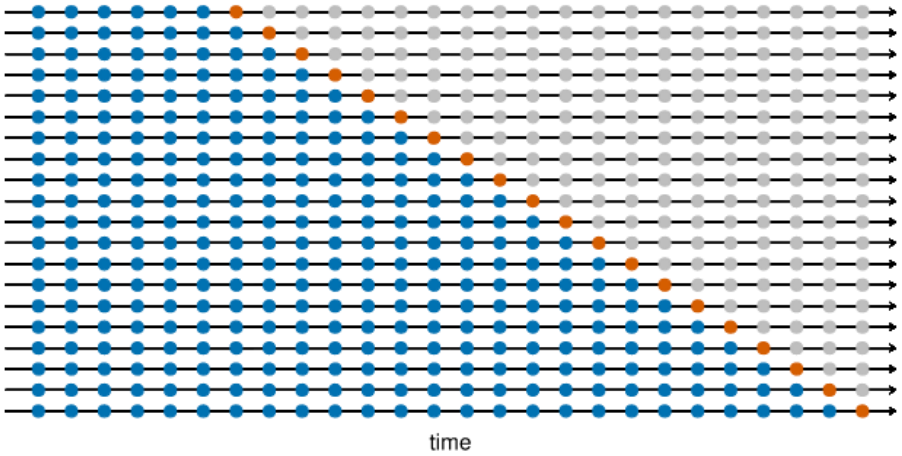
3. The process is repeated for $k = 1, 2, \ldots, K$ and all the estimates of prediction error are finally combined.

The optimal choice of K depends on the application considering the amount of data and the complexity of the model that is tested. Moreover, there is a bias-variance trade-off associated with the choice of K. Typically K-fold cross-validation is performed using $K = 5$ or $K = 10$, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor very high variance[2] (Trevor Hastie, 2009). When considering classification tasks, this process provides suitable results for balanced classification but it is unreliable for imbalance classes since the data is split randomly without taking care of the underlying class imbalance. In such cases, stratified K-fold cross-validation should be adopted. It is an extension of the standard K-fold cross-validation technique specifically used for classification problems and its aim is to maintain the same class ratio throughout the K folds as the ratio in the original dataset.

[2] Bias can be defined as the difference between the average value predicted by the machine learning model and the actual value, while variance is the variability of model prediction for a given data point. The bias-variance tradeoff is a well-known problem when dealing with supervised learning. Ideally, a machine learning model should both accurately capture the regular patterns in its training data and generalize well to unseen data. In reality, it is typically impossible to do both simultaneously. In fact, high variance learning methods tend to overfit noisy or unrepresentative training data, while high bias learning methods tend to produce simpler models that may fail to capture important patterns.

Moreover, when considering time-series data, the cross-validation procedures that have been mentioned before are frequently not applicable. In fact, in the case of underlying time dependence, the observations should not be shuffled before building folds since the order of observations is important. Several time-series that represent the evolution of closing prices of different stocks listed in the FTSE MIB underlie the basis of the automated trading system built in this dissertation. Hence, there is a need to adjust the original cross-validation procedure to accommodate the need of providing the algorithm with a sort of "memory". In practice, several test sets are considered (orange points), each consisting of a single observation while the training set consists of observations that occurred prior to the observation that forms the test set (blue points), as shown in Figure 5. Finally, the overall accuracy is computed by averaging all the estimates of prediction errors as already explained above when describing K-fold cross-validation..

Figure 5 – Time Series Cross-Validation



Source: Forecasting, Principles and Practice (3rd ed.)

1.5 Automated Trading System

The main objective of this thesis consists in building a reliable and effective automated trading system based on random forest algorithms and technical indicators. The scientific literature provides several examples of successful implementations of machine learning techniques and hybrid approaches in building automated trading systems, but this research aims to explore its application to the Italian stock market, specifically in the energy and utility sector. In order to do that, the FTSE MIB index is considered.

FTSE MIB is the benchmark stock market index for the Borsa Italiana (Italian national stock exchange) and tracks the performance of forty leading companies. As of 29 April 2022, the overall net market capitalization was 395.037,00 million euros (FTSE Russell, 2022). A comprehensive list of all the ICB Supersector[3] and a respective number of firms/market capitalization is provided in Table 1. To fulfill the requirement of specificity and restrict application fields, the sectors that have been considered are utilities and energy. These two sectors cover more than 30% of the FTSE MIB total market capitalization, providing a solid base on which the trading system can be built.

Table 1 – FTSE MIB ICB Supersector Breakdown, FTSE Russel Factsheet

Code	ICB* Supersector	Nr. Of Cons.	Net Mkt Cap. (EURm)**	Wgt %
3010	Banks	6	67,635.00	17.12
6510	Utilities	5	67,510.00	17.09
4010	Automobiles and Parts	3	53,742.00	13.60
6010	Energy	4	52,280.00	13.23
5020	Industrial Goods and Services	7	48,081.00	12.17
3030	Insurance	3	30,264.00	7.66
1010	Technology	1	23,222.00	5.88
3020	Financial Services	4	17,434.00	4.41
2010	Health Care	3	12,184.00	3.08
4020	Consumer Products and Services	1	10,797.00	2.73
1510	Telecommunications	2	6,561.00	1.66
4510	Food Beverage and Tobacco	1	5,325.00	1.35
Totals		**40**	**395,037**	**100.00**

** Industry Classification Benchmark*

*** As of 29 April 2022*

The chosen time frame ranges from January 1st, 2016, to December 31st, 2021. According to this time frame, only stocks belonging to utility and energy sectors that have been listed throughout all the periods are considered, as shown in Table 2.

[3] The Industry Classification Benchmark (ICB) is a detailed and comprehensive structure for sector and industry analysis, facilitating the comparison of companies across four levels of classification and national boundaries. The classification system allocates companies to the Subsector whose definition closely describes the nature of its business as determined from the source of its revenue or the source of the majority of its revenue where available (Russell, 2022).

Table 2 – Experimental Framework Stock List

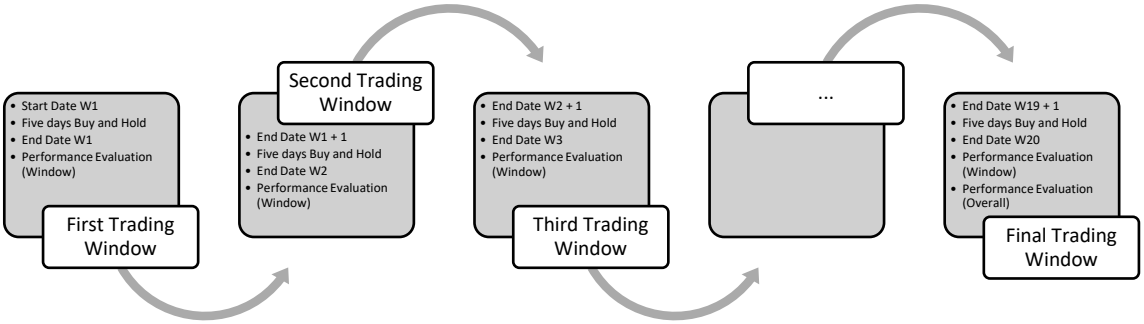
Firm Name	**Ticker***
A2A SpA	A2A.MI
Enel SpA	ENEL.MI
Eni SpA	ENI.MI
Hera SpA	HER.MI
Saipem SpA	SPM.MI
Snam SpA	SRG.MI
Terna - Rete Elettrica Nazionale SpA	TRN.MI

* As it appears on the Yahoo! Finance platform

In order to conduct the experiment, stock data are gathered, and technical indicators are computed on the basis of trading prices and volumes. This procedure will be explained in detail in Chapter II and Chapter III since it constitutes a pillar of the experiment. This dissertation serves a twofold purpose, while the first is undoubtedly building an automated trading system, the second consists in assessing whether additional environmental information can influence its performance. That additional information includes main Italian economic indicators release dates, holidays, and United Nations Climate Change Conference days. In fact, many empirical regularities of financial markets have been documented in the literature considering events such as turn-of-the-month, weekend effects, and exchange holidays (Ash Booth, 2014). As an example, numerous studies confirm the hypothesis that asset prices tend to be lower on Mondays than on the preceding Fridays (Rogalski, 1984), (French, 1980). To explore the possible effects of that information, two separate experiments are defined. The first trading system is based only on technical indicators while the second is based on the same technical indicators and the additional environmental data. The random forest algorithm at the heart of those systems is trained on the first five years of the time frame considered, while the last year will be used to evaluate its performance in several trading windows. The inner working of both the trading systems can be resumed as follows. Given a particular day, the algorithm should predict for each stock if the expected return will be positive after five days and provide the likelihood of this occurrence expressed as a probability. After this first step, a hypothetical portfolio is built. This portfolio contains only stocks that are predicted to have positive returns with an expected probability higher than a predefined threshold.

A buy signal is provided, the selected stocks are held for five days and finally, the position is closed. The total return is then computed and stored for performance comparison and evaluation. This process is repeated twenty times following the pattern represented in Figure 6. It is important to underline that no additional transactional costs or fees are considered when opening and closing positions, computing returns, and comparing the financial performances of the different portfolios. An in-depth explanation of the implementation of the trading system will be provided in Chapter III.

Figure 6 – Consecutive Trading Windows Test Framework



As it can be noticed, the algorithm does not predict the actual return but the expected direction of the returns (positive or negative). Forecasting techniques may generate some out-of-scale values which turn out to be outliers. As an example, forecasting the price of a stock as zero is unlikely in the market transactions. On the contrary, the classification model provides a probabilistic view of the predictive analysis, and it plays a safer role as it predicts the direction of the trend using the likelihood of the situation and hence providing more trustworthy results (Suryoday Basak, 2019).

Chapter II: Dataset

Chapter II provides a complete description of all the technical steps and procedures that have been taken in retrieving, cleaning, merging, and managing the experiment essential data. One of the key factors that influence the overall performance of every machine learning algorithm is data quality; incorrect or poor-quality input will always produce inaccurate output. For this reason, this first step has been accomplished with the utmost attention and extreme care, implementing ad hoc Python[4] functions and scripts to minimize the risk of data redundancies and inconsistencies. It is useful to highlight that the only tool used in the data manipulation process was the pandas data analysis library (The pandas development team, 2020) (McKinney, 2010), while time-series data of stocks have been retrieved through the Yahoo! Finance's API[5]. Pandas is a widely known and reliable software library written for data manipulation and analysis in Python. It has been chosen since it is considered one of the best data analysis and manipulation tools available at this time, it provides a conspicuous selection of time-series functionalities, and it is powerful and easy to use. Concerning Yahoo! Finance's API, it is an open-source tool that uses Yahoo's publicly available APIs and is intended for research and educational purposes.

2.1 Time Series Data

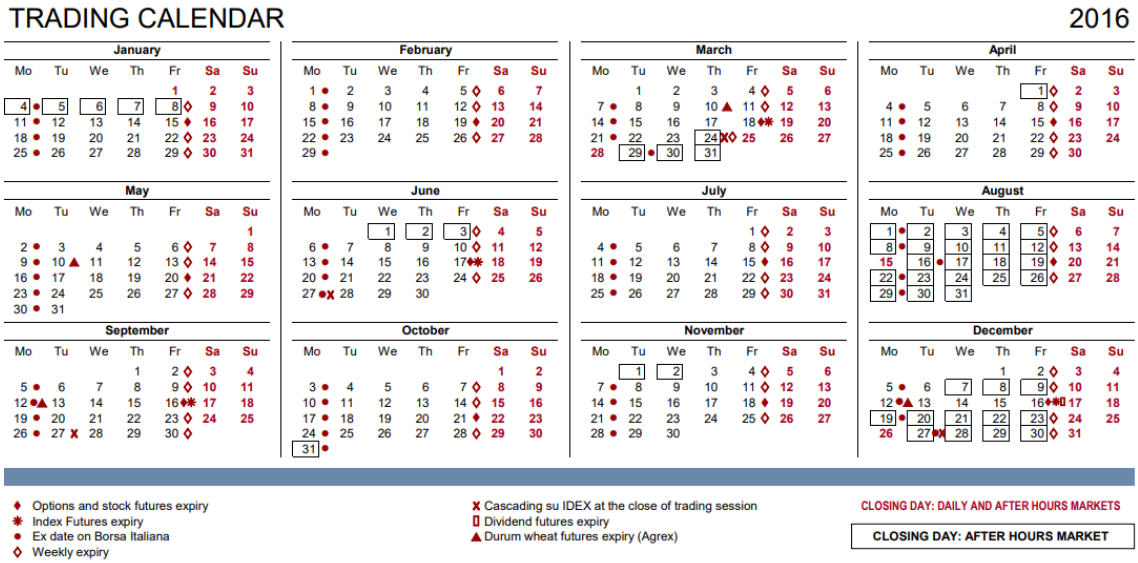
As previously stated, stocks that are considered in this experiment belong to the Italian utility and energy sectors, and the period of interest ranges from January 1st, 2016, to December 31st, 2021. These two preconditions consent to reducing the computational costs and training times, managing however to provide the machine learning algorithm with an appropriate amount of data. Moreover, stock prices and transaction volumes data are collected on a daily basis, considering the final goal of this research of providing an automated tool that deals with five days trading windows.

[4] Python is an open-source, interpreted, high-level programming language. It is one of the most popular languages used by data scientists for various data science projects and applications thanks to dedicated mathematics, statistics, and scientific libraries and functions.

[5] yfinance v. 0.1.70: https://pypi.org/project/yfinance/

It is important to notice that Yahoo! Finance's API needs three parameters to properly gather the time series data. Those parameters consist of a starting date, an ending date, and a trading calendar that indicates the closing days of the market. While setting the first two parameters is straightforward, creating a custom calendar that included the closing day of the market required additional resources. The Borsa Italiana website (Trading Calendar Archive, 2022) provides yearly trading calendars in pdf format, starting from 2006 (an example of a pdf file can be found in Figure 7).

Figure 7 – Borsa Italiana Trading Calendar



Taking as a reference the trading calendars from 2016 to 2021, all the days that were not labeled as 'Closing Day: Daily and After Hours markets' were included in the list of actual trading days. This list was finally used as the trading calendar parameter of Yahoo! Finance's API. Table 3 provides an example of the information gathered for each stock (the example is limited to a single stock, in January 2016. The whole data frame contains more than 10 thousand observations):

- *Date*: trading day date.

- *Ticker*: arrangements of symbols or characters representing the specific company as it appears on the Yahoo! Finance platform.

- *Open*: first transacted price after the market opens for normal trading.

- *High*: highest transacted price hit during the condsidered trading day.

- *Low*: lowest transacted price observed during the condsidered trading day.

- *Close*: last transacted price before the market closes for normal trading.

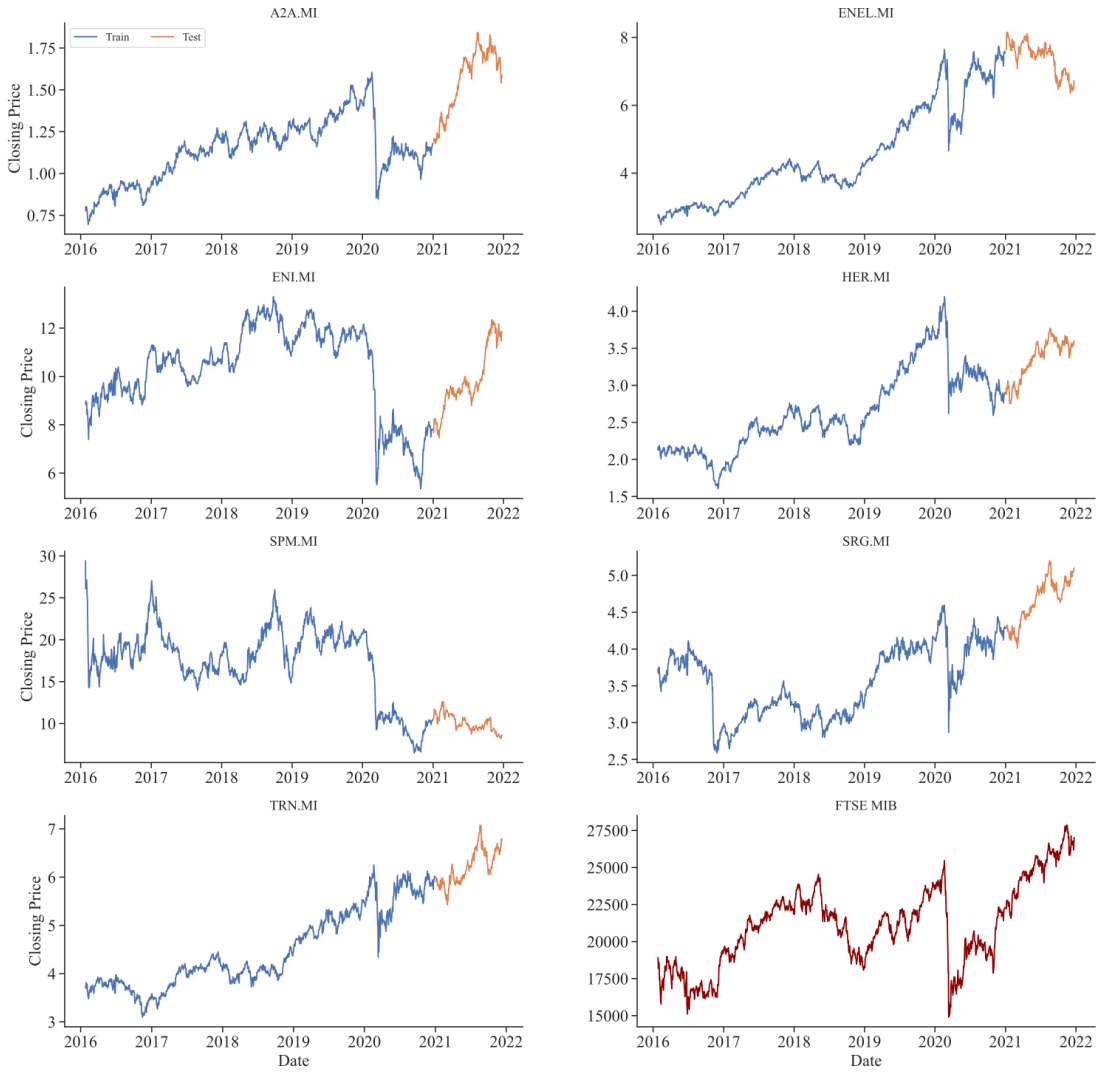
- *Volume*: number of shares of security traded during the considered trading day.

Table 3 – a2a S.p.A. Stock Data, January 2016 Draw

Date	Ticker	Open	High	Low	Close	Volume
1/4/2016	A2A.MI	0.9542	0.9588	0.9373	0.9404	12496412
1/5/2016	A2A.MI	0.948	0.9549	0.935	0.9511	9773848
1/6/2016	A2A.MI	0.9465	0.9596	0.9411	0.945	12137840
1/7/2016	A2A.MI	0.9334	0.9573	0.9181	0.9549	17878384
1/8/2016	A2A.MI	0.9557	0.9657	0.9473	0.9488	14424671
1/11/2016	A2A.MI	0.9526	0.9634	0.9404	0.9526	16012506
1/12/2016	A2A.MI	0.9526	0.9534	0.895	0.9042	38278597
1/13/2016	A2A.MI	0.9219	0.9304	0.885	0.8943	38161268
1/14/2016	A2A.MI	0.8843	0.8896	0.8697	0.8704	25217827
1/15/2016	A2A.MI	0.8651	0.872	0.8436	0.8612	20817266
1/18/2016	A2A.MI	0.8635	0.8666	0.8251	0.8336	17990465
1/19/2016	A2A.MI	0.8436	0.8482	0.8113	0.8428	22502234
1/20/2016	A2A.MI	0.8266	0.8328	0.8051	0.8144	29796865
1/21/2016	A2A.MI	0.8136	0.8412	0.8021	0.8228	22968792
1/22/2016	A2A.MI	0.8336	0.8366	0.8144	0.8274	23199550
1/25/2016	A2A.MI	0.8274	0.832	0.7936	0.8128	29554822
1/26/2016	A2A.MI	0.8067	0.8374	0.7998	0.8336	24279473
1/27/2016	A2A.MI	0.8366	0.8474	0.8205	0.8474	24092591
1/28/2016	A2A.MI	0.8474	0.8551	0.8128	0.8197	20418854
1/29/2016	A2A.MI	0.8366	0.8497	0.8282	0.8459	19932484

Figure 8 graphically summarizes the daily closing prices of each stock being part of the experiment and the last plot (highlighted in red) can be taken as a general reference since represents the FTSE MIB index daily closing price. Moreover, it can be noticed that each time series has been split into two sections. The blue section – labeled as Train – will be used to properly train the random forest algorithm and covers the period that ranges from the beginning of 2016 to the end of 2020. On the other hand the orange section – labeled as Test – will be used to extract several different trading windows from which the overall performance of the automated trading system is evaluated in the year 2021 (in-depth explanation is provided in Chapter III, Method, and Chapter IV, Results).

Figure 8 – Stocks Time Series, Train and Test Partitions

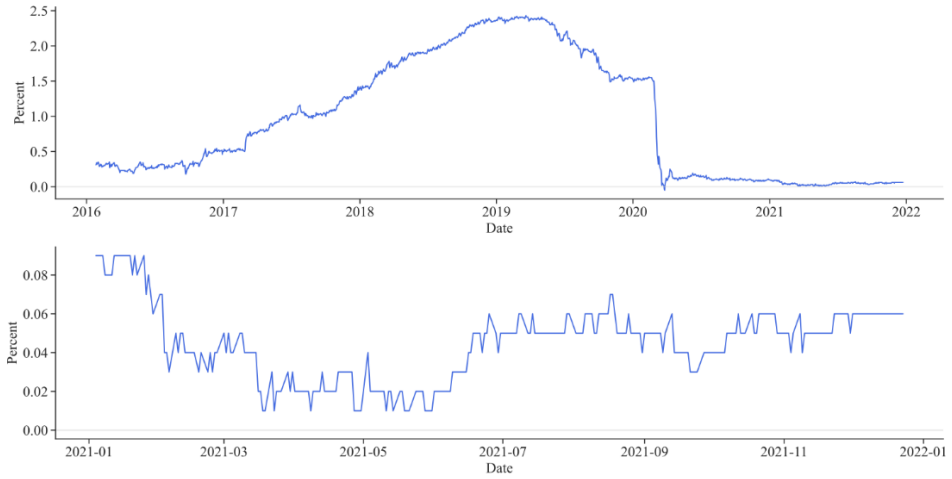


The following step consists of creating an additional data frame containing the three-month U.S. Treasury bill secondary market rate on a business-daily frequency gathered from the FRED Economic Data Online Platform (Board of Governors of the Federal Reserve System, 2022). The above-mentioned time series will be essential in the financial performance evaluation process of the automated trading system since the chosen evaluation metrics – Sharpe Ratios and Sortino Ratios – rely on the rate of return on a risk-free investment for comparison. These two ratios are commonly used by practitioners to determine the risk-adjusted return on investment. The higher the ratios, the greater the investment return relative to the amount of risk taken, and thus, the better the investment (additional details and mathematical formulations will be provided in Chapter III, Method, and Chapter IV, Results).

Scholars and practitioners commonly agree on the fact that in practice a truly risk-free rate does not exist since even the safest investments carry a minimal risk component. The three-month U.S. Treasury bill secondary market rate (DTB3) can be used as a useful proxy due to the fact that the market considers an extremely low chance of the U.S. government defaulting on its obligations. This choice is based on the idea that a hypothetical investor has the opportunity to safely invest abroad his capital considering the Italian government's risk of default as not negligible, negatively affecting its treasury bill interest rates. Since the data is available daily, the natural choice is to maintain the days that were considered actual trading days when looking at the Borsa Italiana trading calendar. This procedure permits to perfectly align the data frame containing the stocks data with the data frame containing the three-month treasury bill secondary market rate. The DTB3 time series ranging from January 2016 to December 2021 is graphically represented in Figure 9 upper graph. Notice that since there is no need to compute Sharpe and Sortino ratios in the training phase of the algorithm[6], the actual DTB3 time series that will be used for comparison begins in January 2021 according to the testing phase period as represented in Figure 9 lower graph.

[6] Sharpe and Sortino's ratios are used to compare the financial performance of the different portfolios once the algorithm makes its prediction. In contrast, in the training phase, the algorithm is evaluated on its predictive capability through different metrics (Accuracy, Precision, Recall, etc.).

Figure 9 – Three-Month Treasury Bill Secondary Market Rate



2.2 Data Integration

Keeping in mind that the final objective of this dissertation is not only to provide a valid and reliable automated trading tool but also to evaluate the potential impact of providing the random forest algorithm with additional environmental information, additional data must be integrated into the main dataset. This supplementary information consists of an exhaustive set of release dates regarding the main economic indicators that may help to judge the overall health of the Italian economy and a set of dates of interest as Italian holidays recognized throughout the territory in which the trading market are active. The intuition is to provide the machine learning algorithm with a collection of 0-1 binary features stating if on a particular day an event occurred (label: 1) or not (label: 0).

2.2.1 Economic Indicators

The purpose of economic indicators is to give an overview of the health state of an economy. For practitioners, economic indicators provide useful information that can lead to discovering new opportunities. Economic indicator announcements are of the utmost importance since markets can move dramatically around each release as traders and investors adjust their portfolios. Data regarding the Italian economic indicators have been retrieved from the Trading Economics website (Trading Economics, 2022).

Trading Economics provides accurate historical data and forecasts for 196 countries and more than twenty million economic indicators, exchange rates, stock market indexes, government bond yields, and commodity prices. Economic indicators are mainly grouped into leading indicators and lagging indicators. The first group is generally used to predict future trends of an economy, while the second trail the economy and is released after economic activity occurs. The economic indicators that have been considered to enrich the original time series are the following:

- Business Confidence (leading indicator): This indicator is based on surveys concerning production, orders, and stocks of finished goods in the industry sector and provides information on future trends. It is usually used to monitor output growth and to anticipate turning points in economic activity.

- Consumer Confidence (leading indicator): This indicator is based on surveys on the expected financial situation, sentiment about the general economic situation, unemployment, and capability of savings of households. It provides an indication of households' future consumption and savings.

- Composite PMI (leading indicator): The composite Purchasing Managers' Index (PMI) consists of an economic health indicator for the manufacturing and service sectors. It is based on monthly surveys of different companies and provides information about current business conditions.

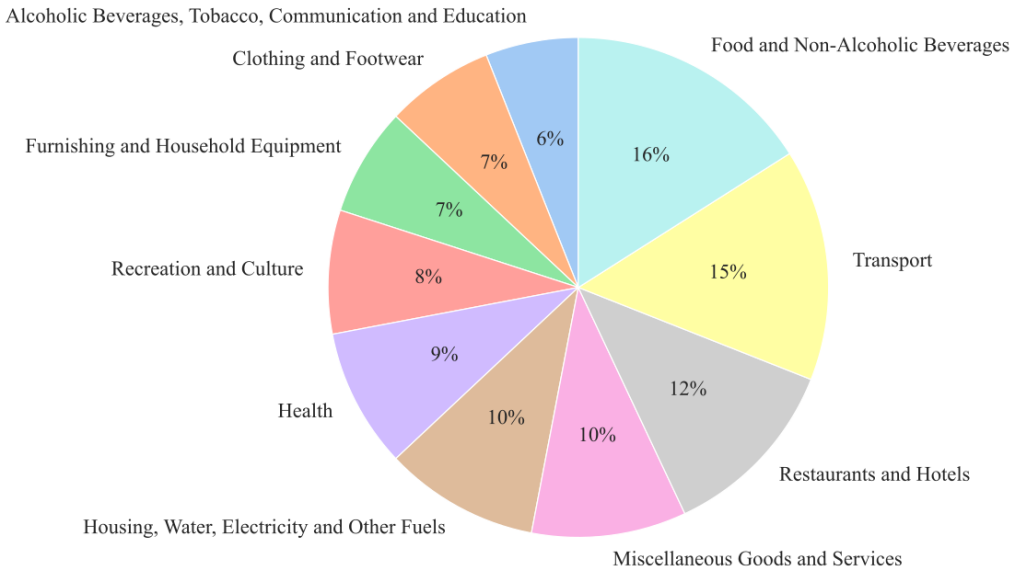
- Services PMI (leading indicator): The services Purchasing Managers' Index (PMI) is an indicator of economic health specifically built on the service sector. It provides information about current business conditions tracking variables such as sales, employment, inventories, and prices.

- PPI (MoM, lagging indicator): Month on Month Producer Price Index measures the variation in the prices of goods sold by industrial producers within the domestic market with respect to the previous month. The index covers products considered most representative of the goods sold in a country by industrial enterprises resident in the country.

- PPI (YoY, lagging indicator): Year over Year Producer Price Index is similar to PPI (MoM) with the only difference of comparing the results with the ones achieved the previous year.

- CPI (MoM, lagging indicator): The Consumer Price Index measures the weighted average of prices of a basket of consumer goods and services. It is calculated by considering price changes for each item in the predetermined basket of goods with respect to the prices recorded the previous month and averaging them.

Figure 10 – Italian Consumer Price Index Composition



Data Source: Trading Economics

- CPI (YoY, lagging indicator): Year over Year Consumer Price Index is similar to CPI (MoM), but results are compared with the ones achieved the previous year.

- GDP (QoQ, leading indicator): Quarter on Quarter Gross Domestic Product is the total monetary value of all the finished goods and services produced within a country with respect to the same value recorded the previous quarter. It is a broad measure of overall domestic production and an indicator of general economic health.

- GDP (YoY, leading indicator): Year over Year Gross Domestic Product is similar to GDP (QoQ), but results are compared with the ones achieved the previous year.

2.2.2 Italian Holidays and Additional Information

Table 4 summarizes further information that may potentially influence the trading sessions. As an example, some Italian holidays do not implicate a closing day in the markets managed by Borsa Italiana, but it is also true that those trading sessions are somewhat conditioned by the non-working day atmosphere. Moreover, Mondays and Fridays may also reflect particular conditions since the firsts discount events occurred during the weekend and the seconds prepare for the trading stop. Since the experiment considers stocks belonging to the Italian utility and energy sectors also United Nations Climate Change Conference days have been included as possible factors that may influence trading volumes and prices. All this additional information has been included in the dataset as 0-1 binary features stating if on a particular day an event occurred or not.

Table 4 – Additional Information Breakdown

	Event	**Date**
Italian Holidays	Epiphany	6th January
	Liberation Day	25th April
	Republic Day	2nd June
	All Saints' Day	1st November
	Feast of the Immaculate Conception	8th December
United Nations Climate Change Conferences	COP 26 Glasgow, UK	31 October to 12 November 2021
	COP 25 Madrid, Spain	2–13 December 2019
	COP 24 Katowice, Poland	2–15 December 2018
	COP 23 Bonn, Germany	6 November to 17 November 2017
	COP 22 Marrakech, Morocco	7 November to 18 November 2016
Weekdays	Each Monday and Friday that is also a trading day	(January 2016 - December 2021)

Chapter III: Method

Chapter III constitutes the core of this essay since it describes in detail all the steps that have been taken to effectively build the automated trading system. Section 3.1 provides the intuitions behind the systems and the definition of the trading rules, with particular attention to the target variable definition. The target variable consists of the feature that the machine learning algorithm should be able to predict after the training phase. In this particular case, the target variable is a 0-1 binary feature that indicates the direction of the market. This feature must be computed by applying several transformations and alignments starting from each stock adjusted closing price. This step is crucial since even the smallest inaccuracy inevitably leads to wrong or inconsistent model outputs. Section 3.2 provides an exhaustive explanation of the technical indicators used as features in training the random forest algorithm, including mathematical formulations and the intuitions behind its implementation. Only a selection of technical indicators will be considered, even if much more could have been included. This selection is made based on a review of the most widely used indicators by practitioners and scholars, considering previous studies on their predictive capacity. Finally, the random forest algorithm implementation will be treated in section 3.3. Since the whole experiment is performed via Python code, the library that has been chosen to implement the algorithm and define its hyperparameters is scikit-learn, version 1.1.1 (Fabian Pedregosa, 2011). An in-depth explanation of each hyperparameter will be given and the whole training process will be described, providing a step-by-step overview. Finally, at the end of this chapter, the comprehensive set of metrics used in the model evaluation process is provided, with particular attention to the implications in the selection of each metric.

3.1 Trading Rules

As anticipated in Chapter I, the automated trading system provides an indication of the direction of the market, expressed as a positive or negative expected return considering a five-day trading window. Specifically, the set of technical indicators – or technical indicators comprehensive of additional information – is observed at time $t-5$ and it is used to predict the direction of the stocks movement 5 days later, at time t. From a practical point of view, if the system predicts a positive return of a certain stock after 5 days it will buy at the next day opening price, hold the position for five trading days, and sell at the closing price 5 days later. The first step in implementing the system trading rules consists in computing the above-mentioned rate of return at time t, since this information is not included in the original dataset. Therefore, a new feature is added to the data frame displaying the values obtained considering Equation 3:

$$r_{i,t} = \frac{Closing\ Price_t - Opening\ Price_{t-5}}{Opening\ Price_{t-5}}. \tag{3}$$

Once this new feature is added, the intuition is to move from a regression problem to a classification problem creating an additional binary feature following Equation 4:

$$target_i = \begin{cases} 1 & \text{if } sign(r_i) \text{ is positive;} \\ 0 & \text{if } sign(r_i) \text{ is negative.} \end{cases} \tag{4}$$

Table 5 provides a sample of the target direction feature implementation, while Equation 5 provides the actual computations of the first Target (%), as an example.

$$\text{Target}(\%)_{2016-01-26} = \frac{\text{Close}_{2016-02-01} - \text{Open}_{2016-01-27}}{\text{Open}_{2016-01-27}} = \tag{5}$$

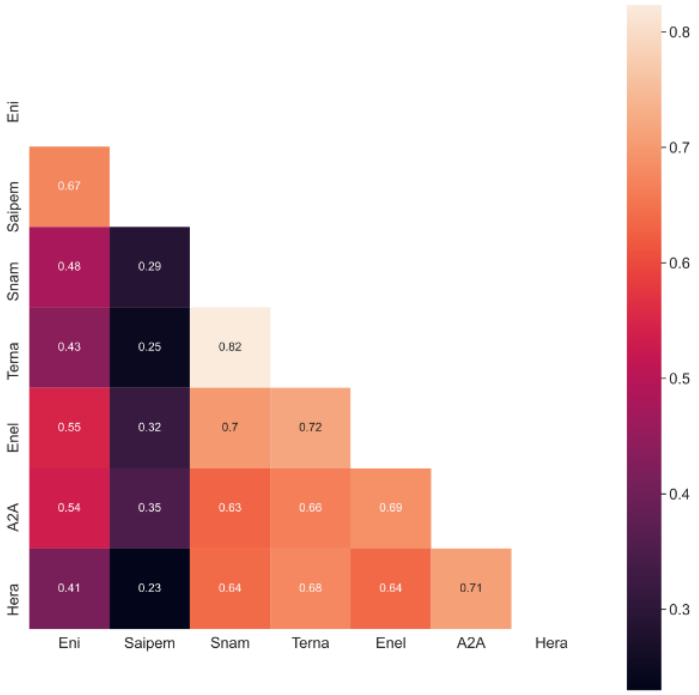
$$= \frac{0.846624 - 0.836637}{0.836637} = 1.1937\ \%.$$

Table 5 – Target Direction Example, A2A SpA

Date	Ticker	Open	Close	Close_Shifted	Target (%)	Direction
2016-01-26	A2A.MI	0.806674	0.833564	0.846624	1.193763	1
2016-01-27	A2A.MI	0.836637	0.847392	0.830491	-1.994569	0
2016-01-28	A2A.MI	0.847392	0.819735	0.824345	-1.469247	0
2016-01-29	A2A.MI	0.836637	0.845856	0.818967	-2.825894	0
2016-02-01	A2A.MI	0.842783	0.846624	0.790541	-6.369421	0
2016-02-02	A2A.MI	0.844319	0.830491	0.752128	-9.935614	0
2016-02-03	A2A.MI	0.835100	0.824345	0.734458	-11.645099	0
2016-02-04	A2A.MI	0.831259	0.818967	0.744061	-8.632074	0
2016-02-05	A2A.MI	0.814357	0.790541	0.750207	-4.917224	0
2016-02-08	A2A.MI	0.789004	0.752128	0.754433	0.101929	1
2016-02-09	A2A.MI	0.753665	0.734458	0.770566	3.884013	1
2016-02-10	A2A.MI	0.741756	0.744061	0.759042	2.916660	1

Moreover, since all the stocks belong to the same industry, returns are likely to be highly correlated, as shown in Figure 11. Pearson correlation coefficient is computed to measure the linear correlation among each pair of stock returns. Considering that the Pearson correlation coefficient can take values that range between -1 and 1, where -1 implies that all data points lie on a line for which Y decreases as X increases and vice versa, positive correlations among the returns of all the stocks is present, where the lowest coefficients belong to Saipem SpA.

Figure 11 – Correlation Matrix, Daily Stock Returns



Following the principles of modern portfolio theory, assets selected for a hypothetical portfolio should not be highly correlated to achieve diversification and reduce the risk of losses. Since in this framework it is possible to perform only low diversification based on different degrees of correlation, an alternative strategy is implemented. The automated trading system chooses the asset to include in the portfolio based on the probability that its predicted return will be positive in the five-day trading window. Specifically, if the computed probability is greater or equal to a predetermined threshold, the asset will be included in the portfolio while it will be excluded otherwise. The actual choice of the threshold value is part of the training phase of the algorithm and will be treated in section 3.3. Since the implemented algorithm consists of a random forest, the predicted class probabilities are computed as the mean predicted class probabilities of the trees in the forest given that the class probability of a single tree is the fraction of samples of the same class in a leaf (Fabian Pedregosa, 2011). In other words, the computed probability is the proportion of trees that predict class c when observation x_i is observed by the classifier. Once all the promising stocks are selected, the overall portfolio return is computed at the end of each trading window thus it is possible to compare its performance with pre-determined benchmarks. The return of the portfolio is computed as shown in Equation 6, assuming that each asset i in the portfolio has return r_i and weight w_i over the total portfolio:

$$r_p = \sum_{i=1}^{n} r_i w_i, \tag{6}$$

where:

$$w_i = \frac{\text{stock value}}{\text{total portfolio value}}.$$

It is important to notice that short selling is not allowed in this framework, meaning that the weights are assumed to be positive. Moreover, the sum of the weights must be exactly equal to 1 i.e., the so-called budget constraint, since all the available resources are invested in each trading window, equally distributing the weights among all the included assets.

Finally, Table 6 and Table 7 resume the occurrences of the binary target for each stock, both for the train set and the test set. This information is crucial to define if the dataset can be defined as balanced i.e., the target class has an even distribution of observations. In fact, especially in assessing the model training performance, the use of common metrics such as the accuracy score in imbalanced data sets can lead to sub-optimal classification models and produce misleading conclusions. As expected, given the stochastic nature of returns, positive market direction labels (1) are approximately the same as negative market direction labels (0). This implies that evaluation metrics such as accuracy, precision, recall, and F-scores can be safely employed in the model testing phase.

Table 6 – Target Distribution, Train Set

Ticker	Target Direction	Occurrences	% Occurrences
A2A.MI	0	561	45.35
	1	676	54.65
ENEL.MI	0	517	41.79
	1	720	58.21
ENI.MI	0	617	49.96
	1	618	50.04
HER.MI	0	570	46.61
	1	653	53.39
SPM.MI	0	649	52.25
	1	593	47.75
SRG.MI	0	574	47.17
	1	643	52.83
TRN.MI	0	547	44.44
	1	684	55.56

Table 7 – Target Distribution, Test Set

Ticker	Target Direction	Occurrences	% Occurrences
A2A.MI	0	107	43.15
	1	141	56.85
ENEL.MI	0	133	53.2
	1	117	46.8
ENI.MI	0	92	36.65
	1	159	63.35
HER.MI	0	114	45.78
	1	135	54.22
SPM.MI	0	128	51.41
	1	121	48.59
SRG.MI	0	100	40.49
	1	147	59.51
TRN.MI	0	100	41.49
	1	141	58.51

3.2 Technical Indicators

As already anticipated in Chapter I, a wide range of indicators is currently employed by practitioners in implementing trading strategies. It must be specified that an indicator as such is not assimilable to a trading strategy and the use of different types of indicators is typically recommended since they complement each other. Hereafter, a complete list of all the technical indicators considered in this essay is provided, including mathematical formulation and the intuition behind their implementation. As a side note, it must be noticed that all the graphical representations and computations have been executed taking as a reference the A2A SpA stock for simplification. Comparable results are achieved when considering all the other time series.

Exponential Moving Average

The aim of exponential moving average (EMA) is to establish the direction in which the price of a security is moving based on past prices, placing greater weight on the most recent data points. For this reason, it can be said that the exponentially weighted moving average is influenced more significantly by recent price changes than a simple moving average, which applies an equal weight to all observations in the period. The mathematical formulation of EMA is shown in Equation 7, while Figure 12 provides its graphical representation.

$$EMA = \begin{cases} Y_0\,, & t = 0; \\ \alpha Y_t + (1-\alpha)\cdot S_{t-1}, & t > 0. \end{cases} \tag{7}$$

where:

$Y_t =$ Value at Period t ;

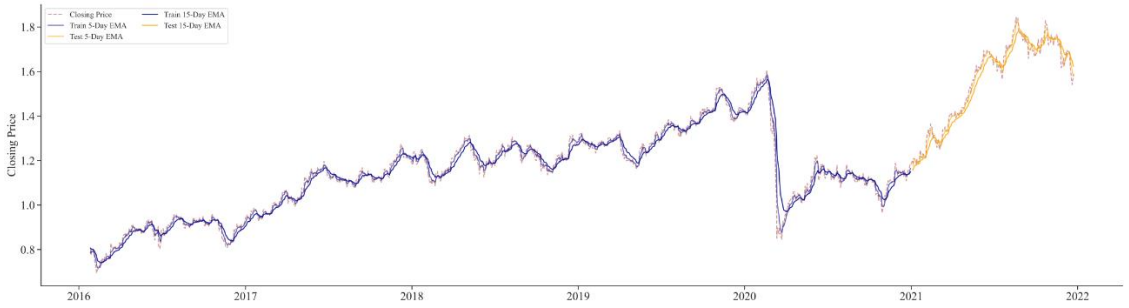
$S_t =$ EMA Value at period t ;

α = Degree of Weighting Decrease.

In this experiment, α has been set to 0.2, and two different EMA are computed, the first covering a 5-day time frame while the second covering a 15-day window. The length of a moving average determines its sensibility to price changes, the longer the time span, the less sensitive the moving average will be, keeping in mind that shorter moving averages are typically used for short-term trading. For those reasons, two short moving averages are chosen with different time windows, to provide different insights into the final model.

It should be highlighted that alpha lies in the $[0, 1]$ range. Higher value of this parameter discounts older observations faster, increasing the importance of recent observations. This means that, since in this experiment alpha is 0.2, 5-day EMA and 15-day EMA weights more past observations than recent observations.

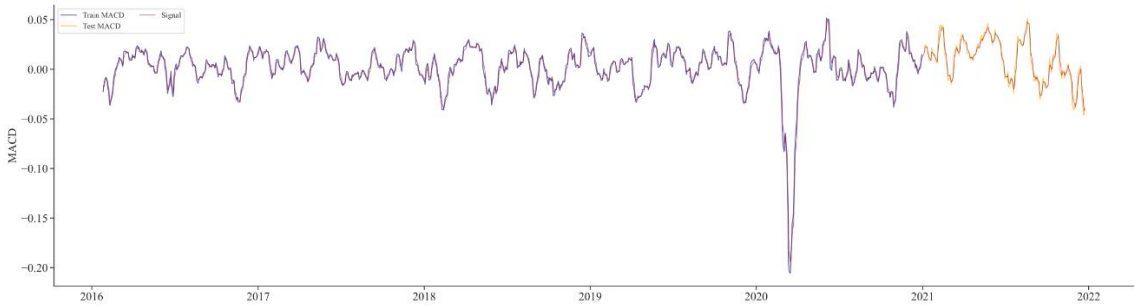
Figure 12 – Exponential Moving Average



Moving Average Convergence Divergence

Moving average convergence divergence (MACD) is constructed on the idea of highlighting the relationship between two moving averages of a security price. Usually, the MACD is calculated by subtracting the 26-day EMA from the 12-day EMA. After that, a 9-day EMA of the MACD i.e., the signal line, is usually plotted versus the MACD line providing buy and sell signals. Specifically, a buy signal is provided if the MACD crosses above its signal line and a sell signal is provided if the MACD crosses below the signal line. Since this specific automatic trading system works on a five-day trading window, the MACD has been computed by subtracting the 15-day EMA from the 5-day EMA (3-day EMA signal line), as already explained when computing EMA independent indicators. The resulting indicator is shown in Figure 13.

Figure 13 – Moving Average Convergence/Divergence



Bollinger Bands

Bollinger bands belong to the family of volatility indicators and consist of a set of three curves that overlap the price chart. These curves are a simple moving average (SMA) of the security price, an upper band, and a lower band. The upper and lower band are usually two standard deviations away from the SMA. A common interpretation of the Bollinger bands consists in considering the market overbought[7] when prices move to the upper band and oversold[8] when the prices move to the lower band, thus providing sell or buy indications. The Bollinger bands can be mathematically represented as shown in Equation 8:

$$SMA(P)_k = \frac{p_{n-k+1} + p_{n-k+2} + \cdots + p_n}{k} = \frac{1}{k}\sum_{i=n-k+1}^{n} p_i\,;$$
$$UPPER = SMA(P)_k + m * \sigma(P)_k; \tag{8}$$
$$LOWER = SMA(P)_k - m * \sigma(P)_k.$$

where:

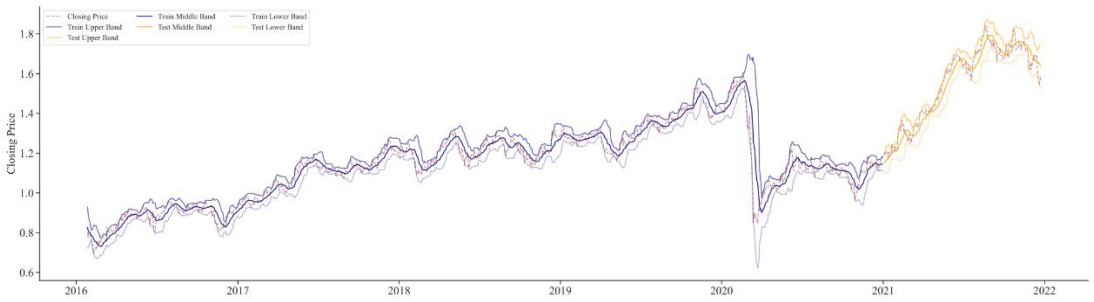
$k =$ Period;
$P =$ Time series get by (High + Low + Close)/3 ;
$p =$ Data Point of P;
$n =$ Number of Entries;
$m =$ Number of Standard Deviations;
$\sigma(P)_k =$ Standard Deviation Over the Last k Periods.

[7] A security is considered to be overbought when its actual trading price is believed to be at a level above its intrinsic or fair value. The intrinsic value can be defined as the present value of all expected future cash flows, discounted at the appropriate discount rate. Generally, it is expected that the market will correct the price with a downward swing in the near future.

[8] A security is considered to be oversold when its actual trading price is believed to be at a level below its intrinsic or fair value. Generally, it is expected that the market will correct the price with an upward swing in the near future.

Since the automatic trading system works on a five-day trading window, the moving average period k has been set to 15 days instead of the usual 20 days period to accommodate for the short-term prediction window. With respect to the number of standard deviations for the upper and lower bands, it has been set to 2 as a common practice. Figure 14 shows the graphical representation of the Bollinger bands on top of the A2A stock closing price.

Figure 14 – Bollinger Bands



Percentage Price Oscillator

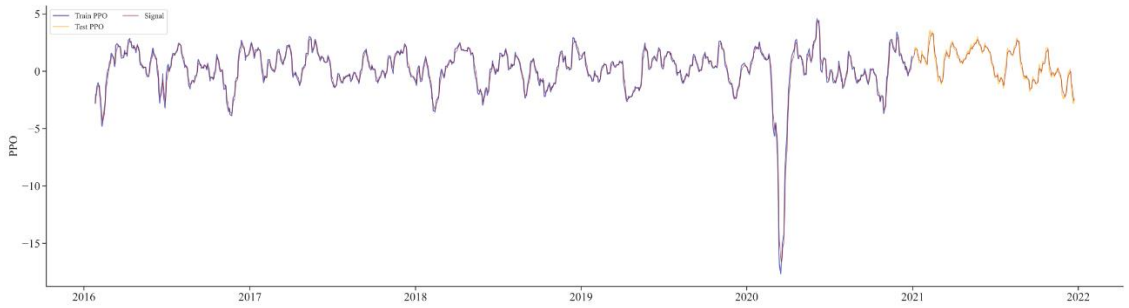
The percentage price oscillator[9] (PPO) is a technical momentum indicator that is to some extent similar to MACD but the relationship between two moving averages is expressed in percentage terms, as shown in Equation 9:

$$PPO = \frac{\text{12-period EMA} - \text{26-period EMA}}{\text{26-period EMA}} \cdot 100\,. \tag{9}$$

As for MACD, the moving averages that are usually considered are a 26-day and 12-day EMA. As already suggested above, the EMA windows have been set to 5 and 15 days, with a signal EMA of 3 days. Figure 15 provides a visual representation of this oscillator.

[9] An oscillator consists of a momentum indicator, whose fluctuations are bounded between an upper band and lower band. Market momentum refers to the ability of a market of maintaining a continuous increase or decrease in price within a certain period. Usually, oscillators provide overbought or oversold signals.

Figure 15 – Percentage Price Oscillator



Market Momentum (MOM)

Market momentum indicates the speed at which the price of a security is changing, as shown in Figure 16. Practitioners usually use momentum indicators to develop strategies that consist in opening long positions[10] when the selected indicator is rising and opening short positions[11] after it has peaked. Technicians typically use a 10-day time frame when measuring market momentum, following Equation 10.

$$MOM = P_t - P_{t-x}, \tag{10}$$

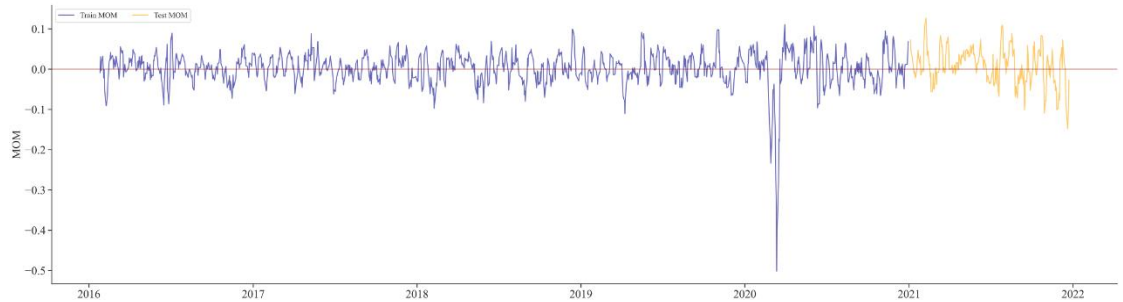
where:

P_t = Closing Price at time t ;

P_{t-x} = Closing Price at time $t - x$.

Since the automatic trading system implemented in this experiment works on a 5-day trading window, the same time frame has been set as x in computing the momentum indicator.

Figure 16 – Market Momentum



[10] Opening a long position consists in buying units of a given instrument.

[11] Opening a short position consists in selling units of an instrument at the current price.

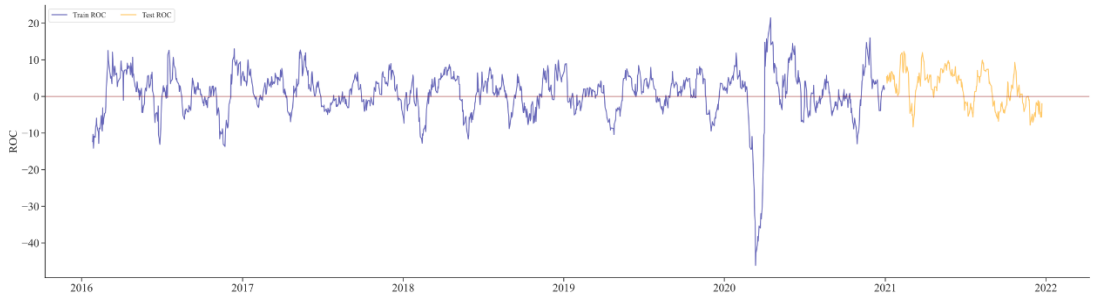
Rate-of-Change (ROC)

Rate-of-change indicator is a momentum indicator and provides the same information of the market momentum, but it is expressed in percentage terms. Rate of Change can be computed following Equation 11:

$$ROC = \frac{P_t - P_{t-x}}{P_{t-x}} \cdot 100\,. \tag{11}$$

In order to guarantee variability in the features provided to the random forest algorithm, the time span used to compute ROC has been set to 15 days. The resulting indicator is graphically represented in Figure 17.

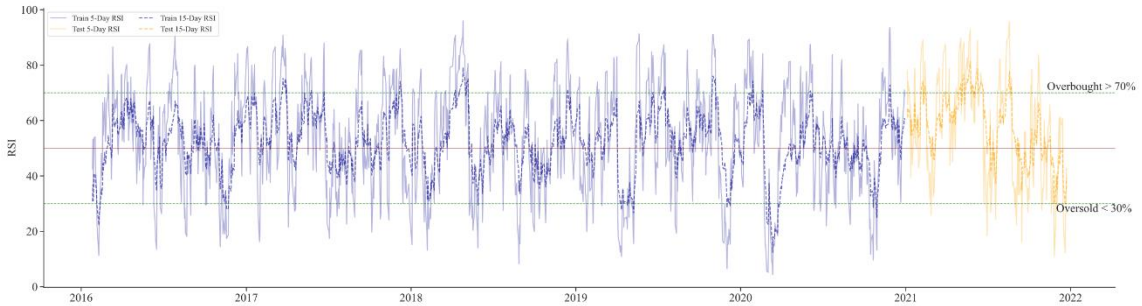
Figure 17 – Rate of Change



Relative Strength Index (RSI)

Relative strength index belongs to the family of momentum indicators and it is displayed as an oscillator ranging from 0 to 100, as shown in Figure 18.

Figure 18 – Relative Strength Index



The usual interpretation given by practitioners consists in considering an asset overbought (overvalued) when the RSI is above 70% and oversold (undervalued) when it is below 30%. Usually, long positions are opened when an asset starts to recover from oversold and vice versa. Equation 12 and Equation 13 provide the necessary steps to compute the relative strength index since the final result derives from a two-part calculation. Those steps include the assessment of average gains and losses over a specified time window. Usually, technicians set this window to 14-day but in this experiment two different time frames have been chosen, computing the first RSI on a 5-day window and the second RSI on a 15-day window.

Step 1:

$$RSI_{step\ 1} = 100 - \left[\frac{100}{1 + \frac{\text{Avg. Gain}}{\text{Avg. Loss}}}\right], \tag{12}$$

where:

$$\text{Avg. Gain} = \frac{\%g_1 + \%g_2 + \cdots + \%g_{5(15)}}{5\ (or\ 15)}, \quad g = \text{Gain};$$

$$\text{Avg. Loss} = \frac{\%l_1 + \%l_2 + \cdots + \%l_{5(15)}}{5\ (or\ 15)}, \quad l = Loss.$$

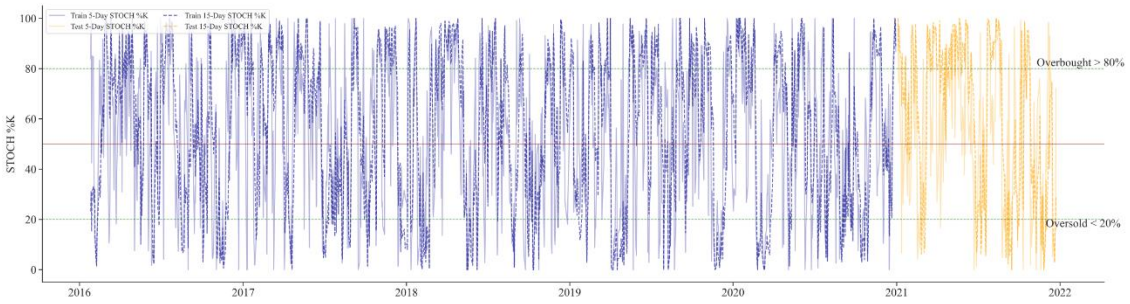
Step 2:

$$RSI_{step\ 2} = 100 - \left[\frac{100}{1 + \frac{(\text{Avg. Gain}_{t-1} \cdot 4(or\ 14)) + \text{Current Gain}}{(\text{Avg. Loss}_{t-1} \cdot 4(or\ 14)) + \text{Current Loss}}}\right]. \tag{13}$$

Stochastic Oscillator %K (STOCH)

According to George C. Lane (Lane, 1984), the developer of the Stochastic Oscillator, its aim is to compare a closing price of a security to a range of its prices over a well-defined time period. Specifically, it is a momentum indicator that ranges from 0 to 100. Usually, practitioners set the time frame to 14 days but following the same reasoning adopted for the other indicators, two different indicators have been computed on a 5-day period and a 15-day period, as shown in Figure 19.

Figure 19 – Stochastic Oscillator %K



The mathematical formulation of this oscillator is provided in Equation 14:

$$\%K = \frac{(P - \mathrm{L5}(or\ 15))}{(\mathrm{H5}(or\ 15) - \mathrm{L5}(or\ 15))} \cdot 100\ . \tag{14}$$

where:

$\mathrm{P} =$ Current Closing Price;

$\mathrm{L5}(or\ 15) =$ The lowest price hit in the past 5 (or 15) days;

$\mathrm{H5}(or\ 15) =$ The highest price hit in the past 5 (or 15) days.

Generally, values over 80 are considered in the overbought range, while values under 20 are considered oversold.

Williams %R (WILLIAMS)

The Williams %R indicator is remarkably similar to the stochastic oscillator, with a slightly different mathematical foundation, as shown in Equation 15:

$$\%R = \frac{H15 - P}{H15 - L15}\ , \tag{15}$$

where:

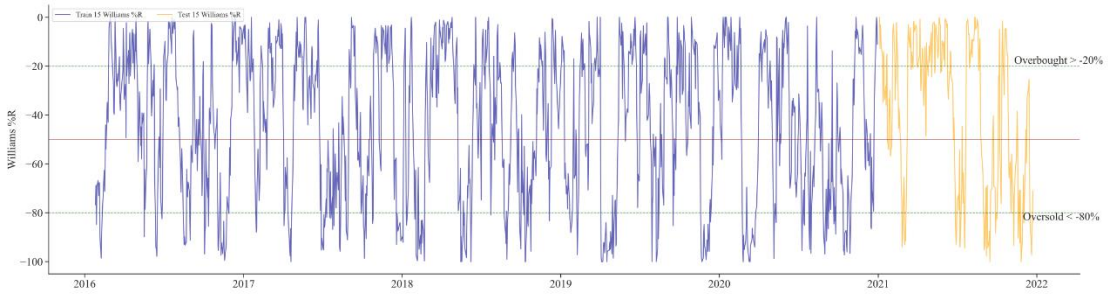
$\mathrm{P} =$ Current Closing Price;

$\mathrm{L15} =$ The lowest price hit in the past 15 days;

$\mathrm{H15} =$ The highest price hit in the past 15 days.

Generally, values under -80 are considered in the oversold range, while values over -20 are considered overbought. The main difference between Williams %R (Figure 20) and the stochastic oscillator consist in the fact that the first represents the closing level versus the highest high for the lookback period, while the second represent the closing level in relation to the lowest low. In other words, the Williams %R is the inverse of the Stochastic Oscillator and has a different scale since it ranges from 0 to -100 instead of $+100$.

Figure 20 – Williams %R



On-Balance Volume (OBV)

On-balance volume momentum indicator aims to predict changes in stock price through volume flow and it consists of a cumulative total of the trading volume (Granville, 1963). Hence, the OBV mathematical formulation is straightforward, as shown in Equation 16:

$$OBV_t = OBV_{t-1} + \begin{cases} volume_t\,, & if\ close_t > close_{t-1}; \\ 0\,, & if\ close_t = close_{t-1}; \\ -volume_t\,, & if\ close_t < close_{t-1}\,. \end{cases} \tag{16}$$

Granville's idea is based on the belief that a large shift in volume without a significant change in the stock price will cause the latter to jump upward or fall downward. Figure 21 provides a visual representation of the on-balance volume indicator.

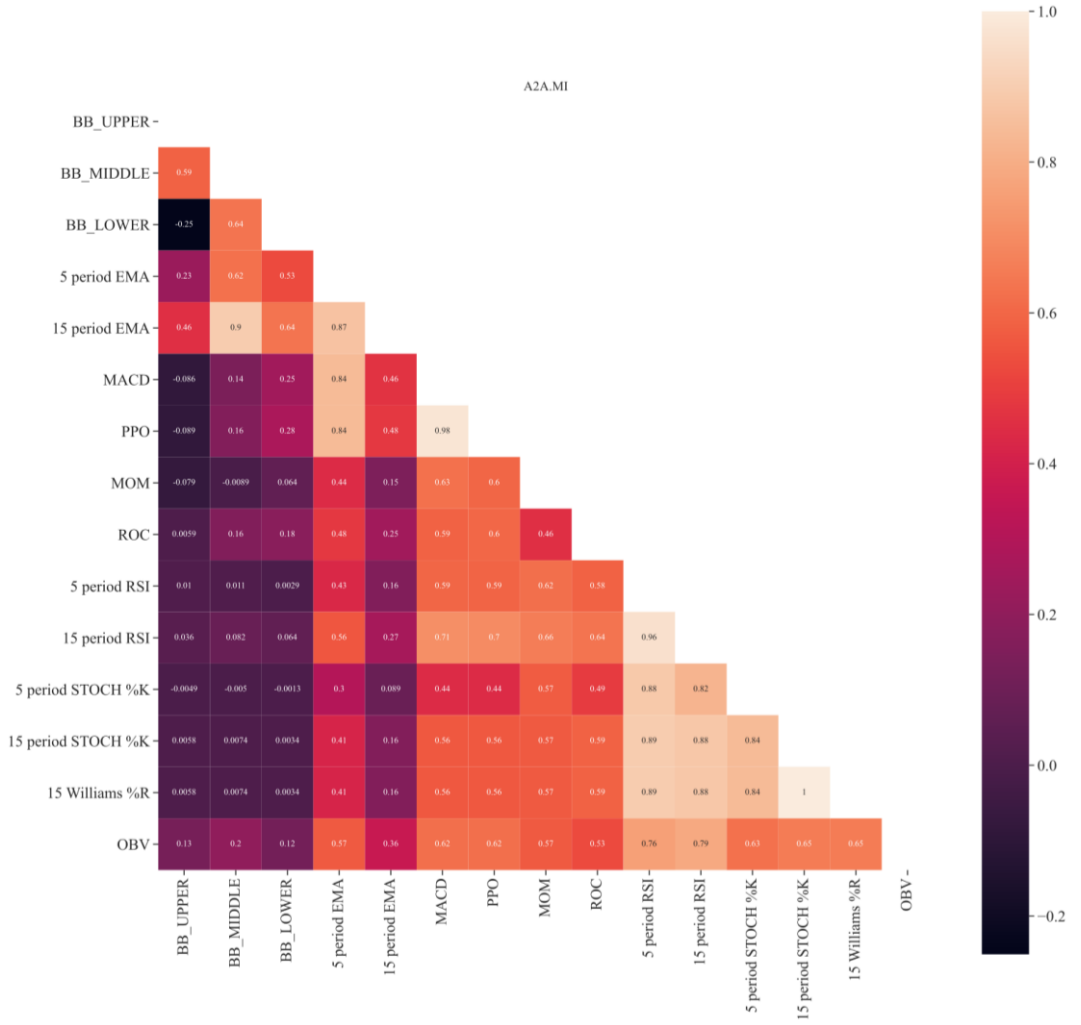
Figure 21 – On Balance Volume

3.2.1 Additional Remarks on Technical Indicators

When adding additional features to a data frame, it is important to consider some important implications such as additional interactions and hidden relations. Specifically, several indicators of the same type that have been computed i.e., momentum indicators or trend indicators, rely on similar inputs and similar processing. In such cases, it may happen that information reverberates, in the sense that different indicators provide the same information thus reducing the overall performance of the machine learning model. This issue is known as multicollinearity and generally occurs when there are high correlations between two or more variables that are used as predictors. The first preventative measure that has been taken in order to mitigate this effect was to compute technical indicators on different time windows, providing the algorithm with different perspectives of the original time series. However, this safeguard may not be sufficient and further investigation is necessary. Thus, each indicator has been treated as a time series and possible correlations are then explored among them. The first step consists in removing the temporal dependence, making the time series stationary through a first-order differencing i.e., computing the differences between consecutive observations. The following step is to compute the Pearson correlation coefficient to measure the linear correlation between each pair of differentiated indicators. Figure 22 provides a visual representation of all the pairwise correlations trough a correlation matrix. Since Pearson correlation coefficient can take values that ranges between -1 and 1, the closer the computed pairwise correlation is to those boundaries, the higher the likelihood of having multicollinearity issues. Notice that the analysis is performed only on indicators relative to A2A stock for simplification, but the same reasoning can be extended to all the other indicators.

As expected, stochastic oscillator %K (STOCH), relative strength index (RSI), and Williams %R (WILLIAMS) have a noticeable pairwise correlation, since they belong to the family of momentum indicators. Moreover, 5-day EMA appears to be highly correlated to the percentage price oscillator (PPO), the moving average convergence divergence (MACD), and the 15-day EMA.

Figure 22 – Correlation Matrix, Technical Indicators



The strategy implemented to choose which feature should be removed involves a recursive feature elimination with cross-validation (Guyon, 2002) mixed with the knowledge acquired from the analysis of the correlation matrix represented in Figure 22.

The recursive feature elimination consists in performing the following steps:

1. Train a random forest classifier.

2. Compute the ranking criterion for all features.

3. Remove the feature with the smallest ranking criterion.

As already anticipated in Chapter I, each tree of a random forest selects the best split for each internal node through the Gini impurity index. Thanks to this index, it can be measured to what extent each feature decreases the impurity of the split. The average over all trees in the forest is the measure of the feature importance. In other terms, feature importance is computed as the mean and standard deviation of accumulation of the impurity decrease within each tree. This feature importance can be finally used as a ranking criterion to prune[12] the features that do not provide consistent benefit in training the classifier. An automatic procedure that recursively eliminates features and returns the optimal number of features can be implemented via cross-validation. This approach consists in randomly dividing the set of observations into five folds of approximately equal size. Then, the first fold is treated as a validation set, and the random forest classifier is fit on the remaining four folds. By doing that, the procedure is repeated five times, making it possible to compare the results of each run and the average is taken as a reference to choose the optimal number of features, as shown in Figure 23. The result of this procedure is reported in Table 8. As it can be seen, the optimal number of features is ten and some of the features that have been automatically removed were previously marked as high possible redundancies.

[12] Pruning can be defined as a data compression technique in machine learning and search algorithms that is implemented to reduce the size of decision trees by removing sections that are non-critical and redundant. Pruning techniques generally reduce the complexity of the final classifier thus improving the predictive accuracy by reducing overfitting.

Figure 23 – Recursive Feature Elimination with Cross-Validation

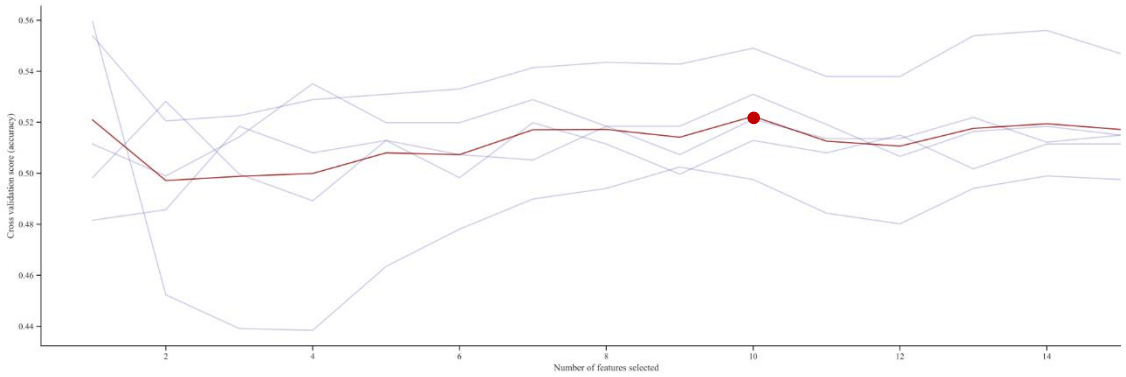


Table 8 – Removed Features

Feature	Removed
Bollinger Band (Upper)	No
Bollinger Band (Middle)	Yes
Bollinger Band (Lower)	No
5-day EMA	Yes
15-day EMA	No
MACD	No
Percentage Price Oscillator	Yes
Market Momentum	No
Rate of Change	No
5-day RSI	No
15-day RSI	No
5-day STOCH %K	Yes
15-day STOCH %K	Yes
15-day Williams %R	No
On-Balance Volume	No

Notice that some indicators have been completely removed by the algorithm in all their variants as happens to the stochastic oscillator in favor of its counterpart, the Williams %R. Looking at the Bollinger bands indicator, the algorithms identified only the middle band as useful, keeping in mind that it consists of a 15-day simple moving average. Finally, Table 9 provides an example of the first ten rows of the complete dataset, with all the features that will be used during the train and the test phase of the algorithm.

Table 9 – Final Dataset Draw

(+ Second Experiment)

Technical Indicators

Date (Removed)	OBV	15 Williams %R	15 period RSI	5 period RSI	ROC	MOM	MACD	15 period EMA	BB_LOWER	BB_UPPER
1/26/2016	-89764834	-76.79	30.79	42.48	-12.36	-0.01	-0.02	0.85	0.77	0.98
1/27/2016	-65672243	-68.75	36.82	53.14	-10.33	0.03	-0.02	0.85	0.77	0.97
1/28/2016	-86091097	-84.82	31.03	36.32	-14.16	0.00	-0.02	0.85	0.77	0.95
1/29/2016	-66158613	-69.23	40.49	53.64	-10.85	0.02	-0.01	0.85	0.77	0.93
2/1/2016	-55477438	-66.83	40.75	54.10	-11.13	0.03	-0.01	0.85	0.79	0.90
2/2/2016	-70029665	-73.03	37.15	42.94	-8.16	0.00	-0.01	0.85	0.80	0.88
2/3/2016	-84990347	-68.00	35.86	39.10	-7.82	-0.02	-0.01	0.84	0.80	0.87
2/4/2016	-98425221	-67.65	34.73	35.61	-5.91	0.00	-0.01	0.84	0.80	0.86
2/5/2016	-120009303	-89.19	29.46	22.42	-8.21	-0.06	-0.02	0.83	0.80	0.86
2/8/2016	-140227247	-98.53	24.15	13.79	-9.77	-0.09	-0.03	0.82	0.77	0.87

Additional Environmental Information

GDP (YoY)	GDP (QoQ)	CPI (YoY)	CPI (MoM)	PPI (YoY)	PPI (MoM)	Services PMI	Composite PMI	Consumer Confidence	Business Confidence	COP Conferences	Friday	Monday	IT_Holyday	Target Direction
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

3.3 Random Forest Algorithm

The scikit-learn random forest implementation has been chosen to implement the algorithm at the core of the automated trading system. Scikit-learn consists in a free software machine learning library for the Python programming language, and the actual version used in this essay is the 1.1. Regarding the basic structure of the single tree, scikit-learn uses an optimized version of the CART algorithm that can be resumed as follows. Given a set of training vectors $x_i \in R^n$ and a label vector $y \in R^l$, the decision tree recursively splits the feature space in a way that samples with the same label are grouped together, as in the classic implementation explained in section 1.3 – CART and Random Forest Algorithms. According to the Scikit-learn documentation, the data at node m is defined as Q_m with n_m total samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m the data is partitioned into $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ subsets, following Equation 17:

$$\begin{aligned} Q_m^{left}(\theta) &= \{(x, y) \mid x_j \leq t_m\}; \\ Q_m^{right}(\theta) &= Q_m \setminus Q_m^{left}(\theta). \end{aligned} \tag{17}$$

The quality of a candidate split of node is then computed using an impurity function $H()$. In the Scikit-learn classification framework two impurity functions are considered, Gini and Shannon entropy. Gini formulation is substantially equivalent to the formulation given in Equation 1, while Equation 18 provides the mathematical foundations behind the Shannon entropy:

$$H(Q_m) = -\sum_k p_{mk} \log(p_{mk}). \tag{18}$$

Since this alternative impurity function is provided, it will be considered as a possible alternative to the Gini impurity index while searching for the best model hyperparameters, as explained in section 1.3.3 – Hyperparameter Tuning. Notice that entropy is more computationally expensive due to the log in the equation. Logarithms are chosen because of many advantageous properties such as the additive property.

This result is particularly useful since in the case of two independent events the probability that they both occur is equal to the product of the probabilities with which they occur. Once one criterion between Gini impurity and Shannon Entropy is chosen, the quality of split is defined following Equation 19:

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H\left(Q_m^{left}(\theta)\right) + \frac{n_m^{right}}{n_m} H\left(Q_m^{right}(\theta)\right). \quad (19)$$

The left component of the above-mentioned equation consists of the number of observations in the left branch over the total number of observations multiplied by the impurity of the left branch. On the other hand, the right component consists of the number of observations in the right subset over the total number of observations multiplied by the impurity of the right subset. Finally, the parameters are chosen minimizing the impurity via Equation 20:

$$\theta^* = argmin_\theta \, G(Q_m, \theta). \quad (20)$$

This process is recursively repeated for all subsets $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ until the maximum depth i.e., the number of nodes from the root down to the furthest leaf node, is reached, as stated in Equation 21:

$$\begin{gathered} n_m < \min_{samples}; \\ or \\ n_m = 1. \end{gathered} \quad (21)$$

The above-mentioned steps are performed in growing all the trees belonging to the forest and the final prediction is given by majority voting, according to the classical implementation (Breiman, Random Forests, 2001). Moreover, as already explained in Chapter I, random forests are based on bagging and feature randomness in training, and it is extremely important to give the user the ability of adjusting the behavior of those components.

For this reason, the scikit-learn implementation of the algorithm provides a variegated set of hyperparameters that can be adjusted to improve the performance of the model and reduce overfitting[13]. The process of finding the right parameters of a model considering the underlying data is defined 'hyperparameter tuning' and will be deeply analyzed in the next section.

3.3.1 Hyperparameter Tuning

When approaching machine learning models, a distinction between parameters and hyperparameters should be made. From a practical point of view, model parameters are internal to the model, and it learns them on its own from historical training data. Typical examples of parameters are the coefficients in linear regression or weights in an artificial neural network. On the other hand, hyperparameters are configurations external to the model that are used to help to estimate the actual model parameters and cannot be learned within the estimator. As an example, the number of trees in a forest and their maximum depth are model hyperparameters. The comprehensive set of hyperparameters that have been taken into account in building the random forest classifier with the scikit-learn implementation is summarized in Figure 24. Notice that each hyperparameter has a default value, enabling the user to train a general model without any kind of acquired knowledge of the data.

Figure 24 – Scikit-Learn "RandomForestClassifier" Class

```
Class sklearn.ensemble.RandomForestClassifier(
                    n_estimators = 100,
                    criterion = 'gini',
                    max_depth = None,
                    min_samples_split = 2,
                    min_samples_leaf = 1,
                    max_features = 'sqrt',
                    bootstrap = True)
```

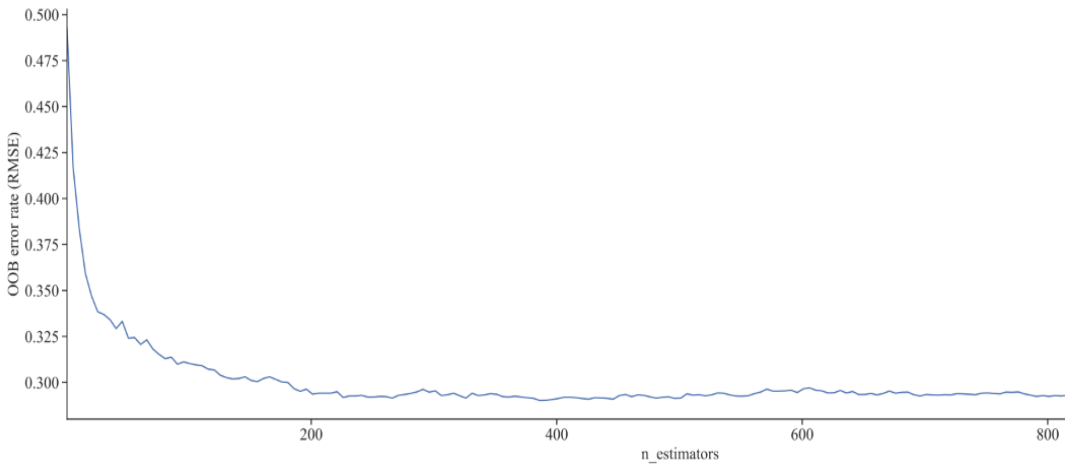
[13] Overfitting can be defined as the production of an analysis which corresponds too closely or exactly to a particular set of data and may therefore fail to fit additional data or predict future observations reliably (Oxford, 2022). A model overfits the training data when it describes features that arise from noise or variance in the data, rather than the underlying distribution from which the data were drawn (Claude Sammut, 2010).

It is important to understand the effect that each of these configurations has on the model before examining the actual procedure of finding the best parameters. Scikit-learn user guide[14] gives an extensive explanation of each parameter, but for the sake of completeness and clarity, a concise breakdown is provided.

Number of estimators (n_estimators)

This hyperparameter controls the number of trees in the forest. This number should be sufficiently large to stabilize the error rate and usually, a good rule of thumb is to start with a value that is ten times the number of features. Figure 25 illustrates an example of this behavior, considering a random forest implementation on the complete dataset, in which all the hyperparameters are kept as default except for the number of trees that ranges from 0 to 800. Moreover, it is important to remember that the impact on computation time increases linearly with the number of trees.

Figure 25 – Number of Estimators and Out of Bag RMSE



Impurity function (criterion)

With this hyperparameter, the user can select the function that measures the quality of a split. As already specified the implemented functions are Gini and Shannon entropy.

[14] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Maximum depth (max_depth)

This hyperparameter controls the maximum depth of the trees. The default value 'None', consents the growth until all leaves are pure or until all leaves contain less than a minimum number of samples chosen by the user. The deeper the tree i.e., the more complex, the more likely it overfits the training data resulting possibly in poor generalization performance. For this reason, there is a balance to be achieved in the depth and complexity to optimize predictive performance on unseen data.

Minimum samples for split (min_samples_split)

The minimum number of samples for split hyperparameter controls how many samples are required to split an internal node. This parameter is used to control over-fitting, preventing a model from learning relations that might be specific to a particular sample. The higher this value, the more samples are required to perform the split, and consequently the tree growth is limited.

Minimum samples for leaf (min_samples_leaf)

The minimum number of samples for leaf hyperparameter specifies the minimum number of samples required to be at a leaf node. The operating principle of this hyperparameter is similar to minimum number of samples for split, in the sense that it is meant to prevent over-fitting. Generally speaking, small values for this hyperparameter makes the model more prone to capturing noise in train data.

Number of features (max_features)

As already stated in Chapter I, each tree in a random forest is allowed to consider only a random subset of the features. This hyperparameter controls how many features should be included in each tree growing phase. Several choices are possible, and for this reason tested, such as:

- A predefined number of randomly chosen features, specified as an integer.

- A randomly chosen fraction of the total features, specified as a floating-point number ranging from 0 to 1, extreme excluded.

- ‘sqrt’, the algorithm will randomly choose $\sqrt{n}$ features, where n is the total number of features.

- ‘log2’, the algorithm will randomly choose $\log_2(n)$ features, where n is the total number of features.

Bootstrap (bootstrap)

Bootstrap hyperparameter controls whether bootstrap samples or the whole dataset is used when building trees and it is set by default to ‘True’.

Once all the hyperparameters that have been considered in the following experiments are defined, it is necessary to understand the actual process that leads to finding the optimal combination of these hyperparameters. The hyperparameter tuning process is usually treated as a black-box optimization problem whose objective function is associated with the predictive performance of the model induced by a machine learning algorithm (Mantovani, et al., 2018). First of all, a function that measures the predictive performance of the model given a hyperparameter configuration should be chosen. Since the experiments are based on a classification framework the F_1 score is chosen as the function that must be maximized. F_1 score belongs to the family of F-measures (F_β measures). The F_β measures can be interpreted as a weighted harmonic mean of the precision and recall. The first step in defining precision and recall consists in giving a visual representation of the model predictions through a confusion matrix. A confusion matrix consists of a table in which each row represents the instances in an actual class while each column represents the instances in a predicted class, as shown in Figure 26.

Figure 26 – Confusion Matrix Illustration

		Prediction Class	
		Negative (0)	Positive (1)
True Class	Negative (0)	**True-Negative (TN)**	**False-Positive (FP)**
	Positive (1)	**False-Negative (FN)**	**True-Positive (TP)**

In the scientific literature it may be found that the axes are actually inverted even if the interpretation remains the same. Confusion matrices are widely used in the field of machine learning since they provide an easy visualization of the performance of an algorithm. Considering the notation in Figure 26, precision and recall can be computed following Equation 22:

$$\text{Precision} = \frac{TP}{TP + FP}\ ; \qquad \text{Recall} = \frac{TP}{TP + FN}. \tag{22}$$

Precision represents the percentage of observations that have been correctly labeled as positive over all the positive labeled observations while recall represents the percentage of observations that have been correctly labeled as positive over all the observations that should have been labeled as positive. To fully evaluate the effectiveness of a model, both precision and recall should be examined. Unfortunately, precision and recall are often in tension, meaning that improving precision typically reduces recall and vice versa. Finally, F_β measures can be computed following Equation 23:

$$F_\beta = (1 + \beta^2)\frac{\text{precision} \times \text{recall}}{\beta^2\text{precision} + \text{recall}}\ . \tag{23}$$

A F_β measure reaches its best value at 1 and its worst score at 0. With $\beta = 1$, F_β and F_1 are equivalent, and the recall and the precision are equally important (Fabian Pedregosa, 2011). The intuition behind the usage of F_1 score lies in the intention of choosing a classifier that equally weights the ability not to label as positive a sample that is negative and the ability to find all the positive values. The methodology that will be used to find the optimal set of hyperparameters consists of a 5-fold cross-validation on the train part of the dataset (Mantovani, et al., 2018). The training folds are used to find good hyperparameter settings, while the test fold is used to evaluate the performance of the optimal solution found. At the end of the process, the set of hyperparameters and their predictive performance are returned and will then be used by the algorithm in the actual training phase.

Finally, Table 10 summarizes the random forest hyperparameter spaces explored in the experiments.

Table 10 – Hyperparameters Search Space

Short Name	Hyperparameter	Setting	Type
n_estimators	Number of estimators	{100, 200, 400, 600}	integer
criterion	Impurity function	{gini, entropy}	factor
max_depth	Maximum depth	{5, 10, 20, 50}	integer
min_samples_split	Minimum samples for split	{5, 10, 20, 30}	integer
min_samples_leaf	Minimum samples for leaf	{2, 5, 10, 15}	integer
max_features	Number of features	{0.5, sqrt(n), log2(n)}	integer
bootstrap	Bootstrap	{True}	logical

Once the search space is defined and the optimal parameters are found, some rules must be defined to compute the performances of the algorithm in predicting the direction of the market in the test trading windows. Moreover, the financial performances should be evaluated against a predefined benchmark and all the metrics should be stored for additional investigation. For this reason, a comprehensive set of dedicated Python functions is built, whose purpose is to build a flexible tool that can be used by the user to replicate the experiment providing different parameters without excessive effort.

3.3.2 Automated Trading System Implementation

The most effective approach to describe in detail the inner working of the entire process that leads to the final trading actions is to take advantage of a pseudocode representation, as shown in Figure 27. Once the train and test set are defined, the operating method is retrieved, selecting alternatively only technical indicators or technical indicators along with additional environmental information. After that, the random forest algorithm is initialized with default hyperparameters as a basis to perform the required step of hyperparameter tuning via cross-validation. Notice that at this point an additional parameter known as ‘random_state’ is defined, such that it is possible to control random processes involved in the model assuring consistency across different executions. This hyperparameter is used to set the seed for the random generator so that the results obtained by the algorithm can be reproduced.

Specifically, when set in random forest algorithms, this hyperparameter controls the bootstrapping procedure and the random subset of features to search for the best feature during the node splitting process, ensuring splits that are always deterministic. For this reason, it should be outlined that the value of the random state hyperparameter indirectly affects the model performance score, and several options should be tested along with other hyperparameters.

Figure 27 – Automated Trading System Pseudocode

```
Train_set = database from 01.01.2016 to 12.31.2020
Test_set = database from 01.01.2021 to 06.30.2021

FOR trial = 1 to n
        IF implementation = 'simple' THEN
                Training_features = [Technical Indicators]
                Train_set = Train_set [Training_features]
                Test_set = Test_set [Training_features]
        ELSE
                Training_features = [Technical Indicators] + [Additional Information]
                Train_set = Train_set [Training_features]
                Test_set = Test_set [Training_features]
        ENDIF

        Random_Forest = Model initialization, default parameters (+ random state)
        Train_CV = Perform Cross Validation for hyperparameter tuning on 5 folds
        Parameters = Save the optimal set of parameters found during CV

        FOR date in list of trading days (20 days chosen at least with 6 days lag):
                Fit Random_Forest [Parameters] on test data

                Prediction = Get and store predictions for the current trading window
                Scores = Get and store model evaluation metrics
                Fin_scores = Get and store financial performance metrics
                Feat_imp = Get and store feature importance

        END FOR

END FOR
```

At the end of this step, the optimal set of hyperparameters is defined, the models are trained on the train set and tested on twenty consecutive five-day trading windows. Therefore, the testing phase consists in predicting the direction of the market at the end of each trading window and comparing the predictions with the historical data.

The testing activity is performed on two complementary aspects, the first consists of assessing the model abilities to correctly identify investment opportunities while the second consists in testing the system's capacity of providing consistent gain against chosen benchmarks. In order to do that some metrics must be specified. As for the first aspect i.e., assessing the model predictive capability, the most popular and effective metrics are accuracy, precision, recall, and F1 score (Hossin & Sulaiman, 2015). While the latter has been already defined in Equation 22 and Equation 23, the accuracy metric can be defined as shown in Equation 24:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \,. \tag{24}$$

Accuracy can be defined as the proportion of correct predictions among the total number of predictions. When assessing model predictive performance in correctly classifying observations it is important to include several metrics since they provide different information. On the other hand, specific measures are used to assess the tool's ability to enhance investment profitability. The desired outcome consists in comparing the performances of the portfolio built by the algorithm trained on the technical indicators and additional information against the portfolio built using technical indicators only, the portfolio built considering all the stocks in the utility and energy sectors and the FTSE.MIB performance on each time window and at the end of the experiment. The chosen metrics are the Sharpe ratio and the Sortino ratio along with the definition of compound rate of return and compound annual growth rate. First of all, it is useful to define what is the meaning of computing compound rates of return. It represents the cumulative effect of a series of gains or losses on an amount of capital over a period of time. Given the vector of daily returns, the respective stock cumulative returns from time $t - k$ to time t can be computed as shown in Equation 25:

$$R_t(k) = (R_t + 1) \cdot (R_{t-1} + 1) \cdot \ldots \cdot (R_{t-k+1} + 1) - 1 \,. \tag{25}$$

Notice that the price used to compute daily return is the adjusted closing price since it incorporates the impact of interest, dividends, stock splits, and other changes on the asset price. Usually, compound rates of return are annualized as compound average annual growth rates to compare different investments.

Since the trading windows that have been considered in this essay cover approximately a four-month time period, the compound annual growth rates are computed as:

$$CAGR = \left(R_t(k)\right)^{\frac{12}{4}} - 1\,. \tag{26}$$

The main drawback of those measures is that volatility or overall investment risk are not taken into account. For this reason, additional measures should be added such as Sharpe and Sortino ratios. The Sharpe Ratio is designed to measure the expected return per unit of risk for a zero-investment strategy (Sharpe, 1994). In other words, it measures the performance of an investment compared to a risk-free asset. The Sharpe ratio is often used to compare two portfolios to each other, considering the best portfolio the one that exhibits the highest Sharpe ratio, since it generates higher returns per unit of risk. Equation 27 provides the mathematical formulation of the Sharpe ratio:

$$\text{Sharpe Ratio} = \frac{R_a - R_\text{f}}{\sigma_a}\,, \tag{27}$$

where:

$R_a = Mean$ Asset Return;
$R_\text{f} =$ Risk-Free Return;
$\sigma_a =$ Standard Deviation of the Asset Return.

Notice that the 3-Month Treasury Bill Secondary Market Rate has been used as the risk-free rate of return in computing the Sharpe Ratio. On the other hand, the Sortino ratio can be considered a variant of the Sharpe ratio, since it measures the performance of the investment relative to the downward deviation. Following this idea, the Sortino ratio takes upside volatility out of the equation and uses only the downside standard deviation in its calculation, considering upside volatility as a benefit for the investment and excluding it from the risk calculation.

$$\text{Sortino Ratio} = \frac{R_a - R_\text{f}}{\sigma_d}\,, \tag{28}$$

where:

R_a = Mean Asset Return;

R_{f} = Risk-Free Return;

σ_d = Standard Deviation of the Downside Return.

Finally, the feature importance is computed, after each training process. This value indicates to what extent each feature contributes to decreasing the impurity of the split. Given that the features for internal nodes are chosen via the Gini impurity criterion, it can be measured how each feature decreases the impurity of the split and averaging this measure over all trees in the forest, the overall feature importance is obtained. This method allows to compute importance in a relatively fast way even if it has some drawbacks. In fact, it should be noticed that the method tends to prefer numerical features and categorical features with high cardinality.

In the case of correlated features, this behavior may lead to selecting one of the features and ignoring the importance of the second one. Notice that in selecting the features to include in this experiment, a preliminary evaluation of all the pairwise correlations has been performed, excluding alternatively one of the features with an exceptionally high absolute value correlation coefficient thanks to the recursive feature elimination with cross-validation method. In conclusion, the full process of finding the optimal hyperparameters, fitting the random forest model on the historical data, getting predictions, and evaluating the overall performances is performed five times for each trading window with different predefined “random_state” seeds.

Chapter IV: Results

This last chapter is divided into two sections; the first section provides an analysis of the predictive capabilities of the random forest algorithm with respect to the evaluation metrics that have been presented at the end of the previous chapter. On the other hand, the second section takes into consideration the financial performance of the system, in order to evaluate if its deployment can be convenient when compared to simpler methods or traditional technical analysis signals. In evaluating this kind of system, it is important to consider both the above-mentioned aspects. In fact, resource-demanding even though highly accurate models that provide no more than marginal improvements with respect to traditional applications might be not desirable due to the high disproportion between implementation efforts and benefits.

4.1 Random Forest Model Evaluation

The first step in evaluating the performances of the random forest classifier is to take into consideration Precision, Accuracy, Recall and F_1 scores for each Trial. Moreover, the results achieved by the model implemented using technical indicators only and the model implemented using also additional information must be compared, to comprehend if their inclusion provides consistent improvements.

Table 11 and Table 12 summarize the mean scores achieved by each model in the twenty trading windows. As it can be seen, the additional features bring useful information to the model improving its predictive capabilities. Specifically, the average F_1 score across all the trials of the "simple" model is 0.55 compared to the average F_1 score of the "enhanced" model of 0.68. This means that adding environmental and additional economical information leads to a 24% improvement in the overall classification performances. Notice that F_1 score improvement is mainly due to higher recall scores that are on average 40% better when considering the enhanced model with respect to the "simple" model.

Table 11 – "Simple" Model Evaluation Metrics

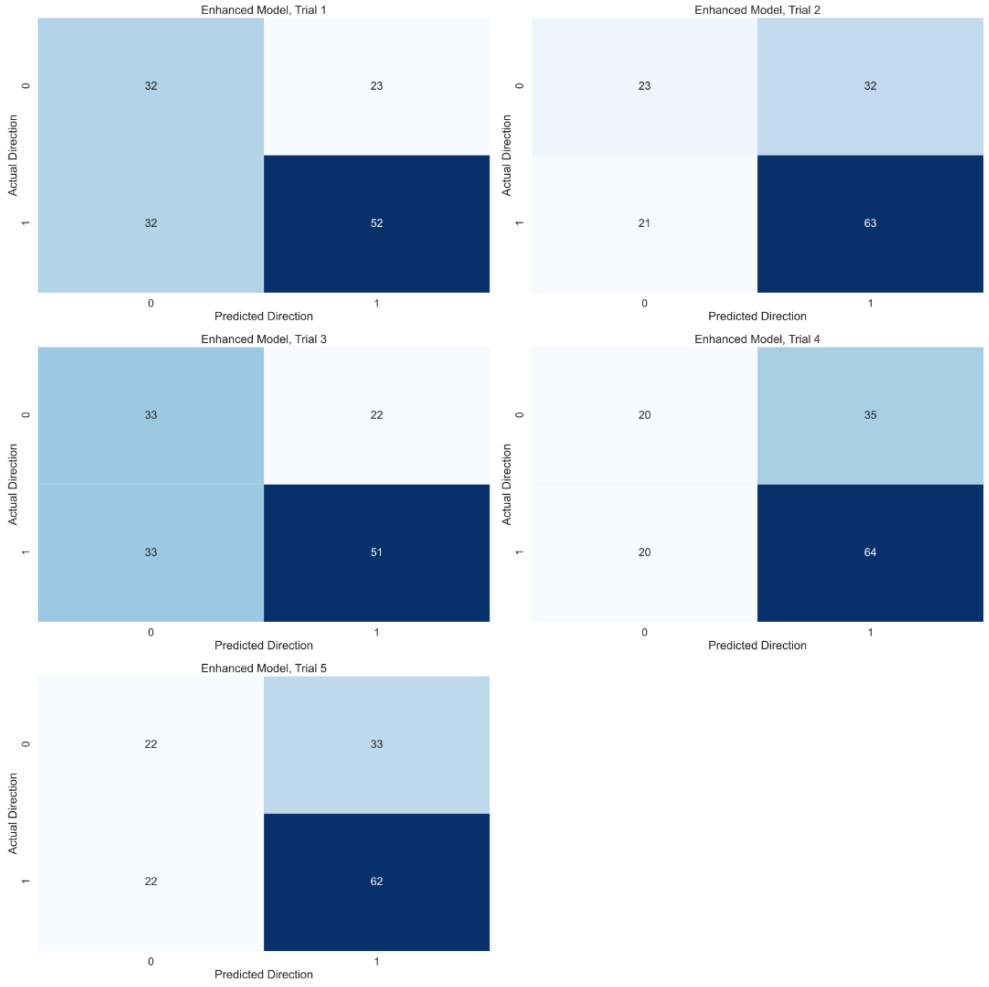
Trial	Precision	Accuracy	Recall	F_1
Trial 01	0.68	0.60	0.47	0.54
Trial 02	0.68	0.58	0.45	0.51
Trial 03	0.71	0.60	0.51	0.57
Trial 04	0.70	0.56	0.45	0.52
Trial 05	0.68	0.61	0.59	0.61
Mean	0.69	0.59	0.49	0.55

Table 12 – "Enhanced" Model Evaluation Metrics

Trial	Precision	Accuracy	Recall	F_1
Trial 01	0.69	0.60	0.61	0.65
Trial 02	0.66	0.62	0.75	0.70
Trial 03	0.70	0.60	0.61	0.65
Trial 04	0.65	0.60	0.76	0.70
Trial 05	0.65	0.60	0.74	0.69
Mean	0.67	0.60	0.69	0.68

In other words, adding information such as economic indicators release dates, weekdays, holidays, and Climate change conference days improves the ability of the model to correctly identify positive future market returns. Moreover, also the reliability of the model in providing consistent results tends to improve, as shown by the lower fluctuation in the accuracy metric computed on the enhanced model predictions (the accuracy score of the "Simple" model ranges from 0.56 to 0.61, while the accuracy score of the "Enhanced" model ranges from 0.60 to 0.62, with all the values but one equal to 0.60). It can be interesting to understand in depth what kinds of classification errors are made by the classifier. Figure 28 summarizes the five confusion matrices that visually represent the performance of the "enhanced" algorithm. Thanks to this kind of visualization, it is possible to compute additional metrics such as specificity, false negative rate, and false positive rate.

Figure 28 – "Enhanced" Model Confusion Matrix



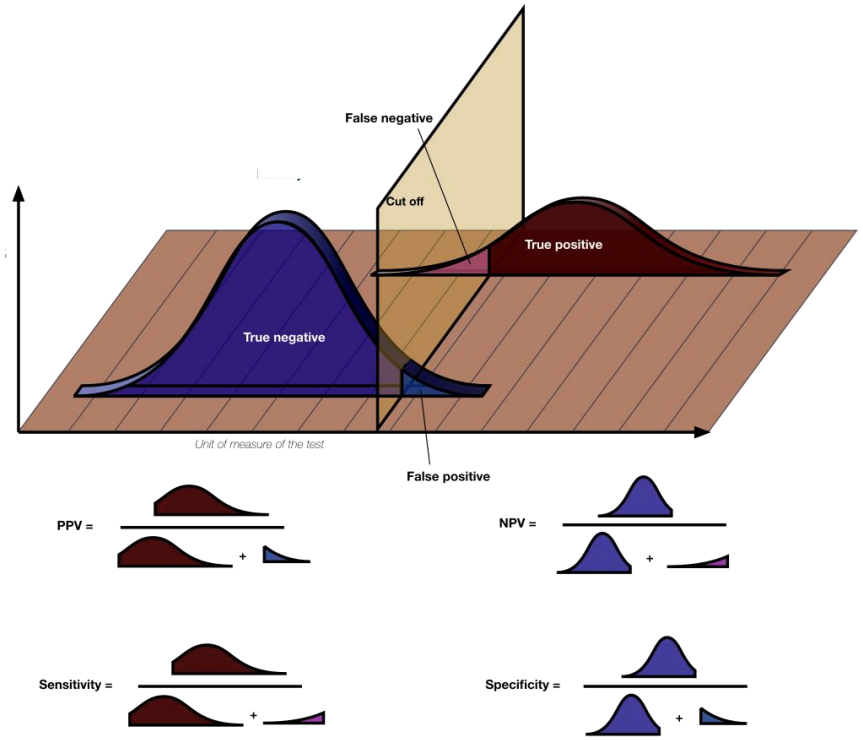
Equation 29 summarizes how those additional metrics can be computed:

$$\begin{aligned} \text{Specificity} &= \frac{TN}{TN + FP}; \\ \text{False Positive Rate} &= \frac{FP}{FP + TN}; \\ \text{False Negative Rate} &= \frac{FN}{FN + TP}. \end{aligned} \tag{29}$$

While sensitivity (recall) measures how well the model identifies positive five-days returns, specificity measures how well the model identifies negative five-day returns. The false positive rate is the proportion of all negative returns that are classified as positive returns (type I error rate) and the false negative rate is the proportion of positive returns that are classified as negative (type II error rate).

The above-mentioned measures are usually analyzed together, depending on the outcomes that the classification model provides and the actual problem to solve. Practitioners should be aware of what kind of error should be avoided when facing a particular task. Considering this experimental framework, a situation in which the model predicts positive returns but in reality, the returns will be negative leads to a loss for the hypothetical investor (type I error). On the other hand, a situation in which the model predicts negative returns but in reality, the returns will be positive leads to a missed opportunity for the investor (type II error). Based on the actual needs, the model can be tuned to limit possible losses or missed opportunities, even if it is not possible to achieve both results at the same time. There is a trade-off between type I error and type II error, as shown in Figure 29. Limiting type I error (also known as a "false positive") leads to an increase in type II error (also known as a "false negative") and vice versa. Varying the threshold (cut-off) value could make the test either more specific or more sensitive, depending on the research needs.

Figure 29 – Sensitivity and Specificity Trade-Off



Source: https://en.wikipedia.org/wiki/Sensitivity_and_specificity (Wikimedia Foundation, 2022)

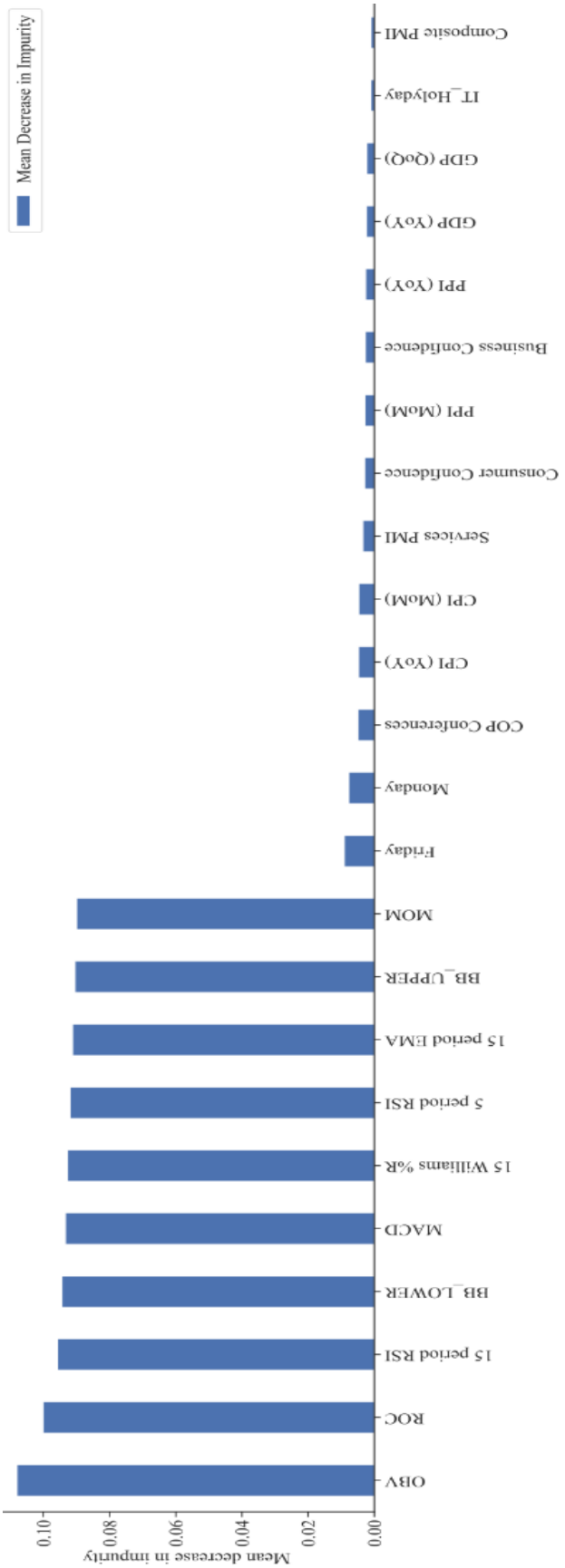
In this research, a balance between type I error and type II error is considered, as a base reference. However, the random forest algorithm that has been developed consent the user to move the threshold shown in Figure 29, to accommodate eventual further research needs. Finally, additional information about the inner working of the model can be obtained by looking at the feature importance, and it is possible to understand to what extent each feature contributes to decreasing the impurity of the split. Table 13 summarizes the feature importance computed on the best performing model i.e., the model that includes additional environmental information. As it can be seen, on-balance volume, rate of change, and the 15-period relative strength index are the features that on average reduce the most the decrease in impurity. From a practical point of view, the combination of those three features suggests that predictions are influenced mainly by trading volumes, momentum, and overbought or oversold conditions.

Table 13 – Random Forest Feature Importance

Feature Name	Mean Decrease in Impurity	Feature Name	Mean Decrease in Impurity
OBV	0.108919	**CPI (MoM)**	0.004460
ROC	0.100193	**CPI (YoY)**	0.004422
15 period RSI	0.096424	**COP Conferences**	0.004340
MACD	0.094030	**Services PMI**	0.003423
15 Williams %R	0.093837	**Consumer Confidence**	0.002529
5 period RSI	0.093320	**PPI (MoM)**	0.002493
BB_LOWER	0.092363	**PPI (YoY)**	0.002482
MOM	0.091828	**Business Confidence**	0.002432
15 period EMA	0.090067	**GDP (QoQ)**	0.002125
BB_UPPER	0.089749	**GDP (YoY)**	0.002046
Friday	0.008796	**IT_Holyday**	0.000934
Monday	0.007919	**Composite PMI**	0.000868

Figure 30 provides a visual representation of the extracted feature importance. Notice that the fact that a particular trading session falls on Friday or Monday slightly affects the final prediction when compared to the technical indicators mean decrease in impurity.

Figure 30 – Random Forest Feature Importance



4.2 Financial Performance Evaluation

The last step consists in evaluating the financial performance achieved using the automated tool against alternative methods and predetermined benchmarks. The two models are tested against a simple buy-and-hold strategy on the entire sector based on the twenty days trading windows and the overall FTSE.MIB performance. Specifically, at the end of each trading session the returns of the portfolios built using the random forest models, the returns of the portfolio that contains all the stocks of the Italian utility and energy sectors, and the returns of the portfolio that contains all the stocks listed in the FTE.MIB index are computed and stored in a vector. In addition to simple daily returns, cumulative returns and compound annual average growth rates of each portfolio are computed, making it possible to compare the overall performances during the whole testing period. Moreover, for each portfolio and trading session, Sharpe and Sortino ratios are also computed. Table 14 summarizes some statistical proprieties of the daily returns while Table 15 resumes all the extracted metrics for each trading session (notice that the trading session date refers to the day on which the position is opened). Since five different trials were performed for each trading session, the metrics in Table 15 represent the average value of the five results for that session.

Table 14 – Daily Returns Summary Statistics

	Sector Return	**FTSE.MIB Return**	**Simple Model Return**	**Enhanced Model Return**
Mean	0.0312	0.0542	0.0334	0.0359
Median	0.0122	0.0559	0.0182	0.0219
Standard Deviation	0.1492	0.1411	0.1049	0.1063
Max Value	0.3057	0.3496	0.2488	0.2329
Min Value	-0.2415	-0.1794	-0.2334	-0.2309

As it can be noticed, the average daily returns of the portfolio built using the automated trading tool are similar to the benchmark average daily returns. The difference lies in the variability (volatility) of the returns since the standard deviation of the daily returns obtained using the automated trading tool is lower than the standard deviation computed on the two benchmark portfolios.

Table 15 – Financial Performance, Cumulative Daily Returns

Date	*01/04*	*01/12*	*01/22*	*01/28*	*02/03*	*02/09*	*02/15*	*02/22*	*02/26*	*03/05*	*03/11*	*03/17*	*03/23*	*03/29*	*04/06*	*04/09*	*04/16*	*04/22*	*04/28*	*05/04*
Sector Return	0.24	-0.05	0.08	0.23	0.08	0.16	-0.20	-0.24	-0.04	0.31	0.01	-0.02	0.24	-0.06	-0.08	0.02	0.02	-0.12	-0.07	0.12
FTSE.MIB Return	0.12	-0.06	-0.16	0.32	0.23	0.04	-0.18	0.08	0.04	0.35	0.07	-0.02	0.15	0.09	-0.05	0.09	-0.10	-0.04	0.00	0.11
Simple Model Return	0.14	0.00	0.06	0.21	0.02	0.09	-0.07	-0.23	-0.05	0.25	0.02	0.04	0.13	-0.01	-0.01	0.01	0.02	-0.01	-0.03	0.09
Enhanced Model Return	0.18	0.00	0.07	0.23	0.03	0.09	-0.07	-0.23	-0.06	0.22	0.02	0.02	0.14	0.01	-0.02	0.02	0.04	0.00	-0.05	0.09
Treasury Bill Rate	0.09	0.09	0.06	0.04	0.05	0.04	0.04	0.05	0.04	0.04	0.01	0.02	0.02	0.02	0.03	0.02	0.03	0.01	0.02	0.02
Sector Cumulative Return	0.24	0.18	0.27	0.57	0.69	0.96	0.58	0.20	0.15	0.50	0.51	0.48	0.83	0.72	0.59	0.62	0.65	0.46	0.35	0.51
FTSE.MIB Cumulative Return	0.12	0.05	-0.12	0.17	0.43	0.50	0.23	0.32	0.37	0.85	0.98	0.94	1.23	1.43	1.32	1.53	1.28	1.19	1.19	1.43
Simple Model Cumulative Return	0.14	0.14	0.21	0.47	0.50	0.64	0.52	0.17	0.10	0.38	0.40	0.45	0.64	0.62	0.61	0.62	0.66	0.64	0.59	0.74
Enhanced Model Cumulative Return	0.18	0.17	0.25	0.55	0.59	0.74	0.61	0.24	0.17	0.42	0.45	0.49	0.70	0.70	0.67	0.70	0.77	0.76	0.68	0.83
Treasury Bill Cumulative Return	0.09	0.19	0.26	0.31	0.38	0.43	0.49	0.56	0.62	0.69	0.71	0.74	0.78	0.81	0.86	0.90	0.96	0.98	1.02	1.06
Sharpe Ratio Sector	7.95	-24.51	3.06	39.37	6.12	7.27	-24.48	-27.24	-3.36	24.15	-0.79	-2.56	16.23	-12.57	-17.62	0.63	-3.44	-10.02	-9.03	9.04
Sharpe Ratio Simple Model	2.78	-15.58	0.05	35.36	-7.21	3.09	-11.47	-26.49	-3.86	18.98	1.68	1.00	8.40	-4.95	-6.11	-2.34	-2.20	-1.71	-4.82	6.66
Sharpe Ratio Full Model	4.84	-16.90	1.19	39.37	-4.86	3.09	-11.47	-26.25	-3.94	15.91	2.01	0.32	9.02	-2.26	-8.69	-0.54	3.23	-0.94	-6.46	6.06
Sortino Ratio Sector	22.97	-24.51	30.00	30.00	30.00	30.00	-34.56	-27.24	-31.38	30.00	-1.70	-30.00	30.00	-12.57	-75.50	30.00	-5.24	-30.79	-30.00	37.15
Sortino Ratio Simple Model	8.03	-15.58	18.00	30.00	-30.00	30.00	-16.19	-26.49	-36.08	30.00	3.60	30.00	30.00	-4.95	-26.17	-30.00	-3.35	-5.25	-30.00	27.36
Sortino Ratio Full Model	13.98	-16.90	30.00	30.00	-18.00	30.00	-16.19	-26.25	-36.80	30.00	4.30	18.00	30.00	-2.26	-37.23	-30.00	4.91	-2.87	-30.00	24.92

Figure 31 – Simple and Enhanced Model Daily Returns

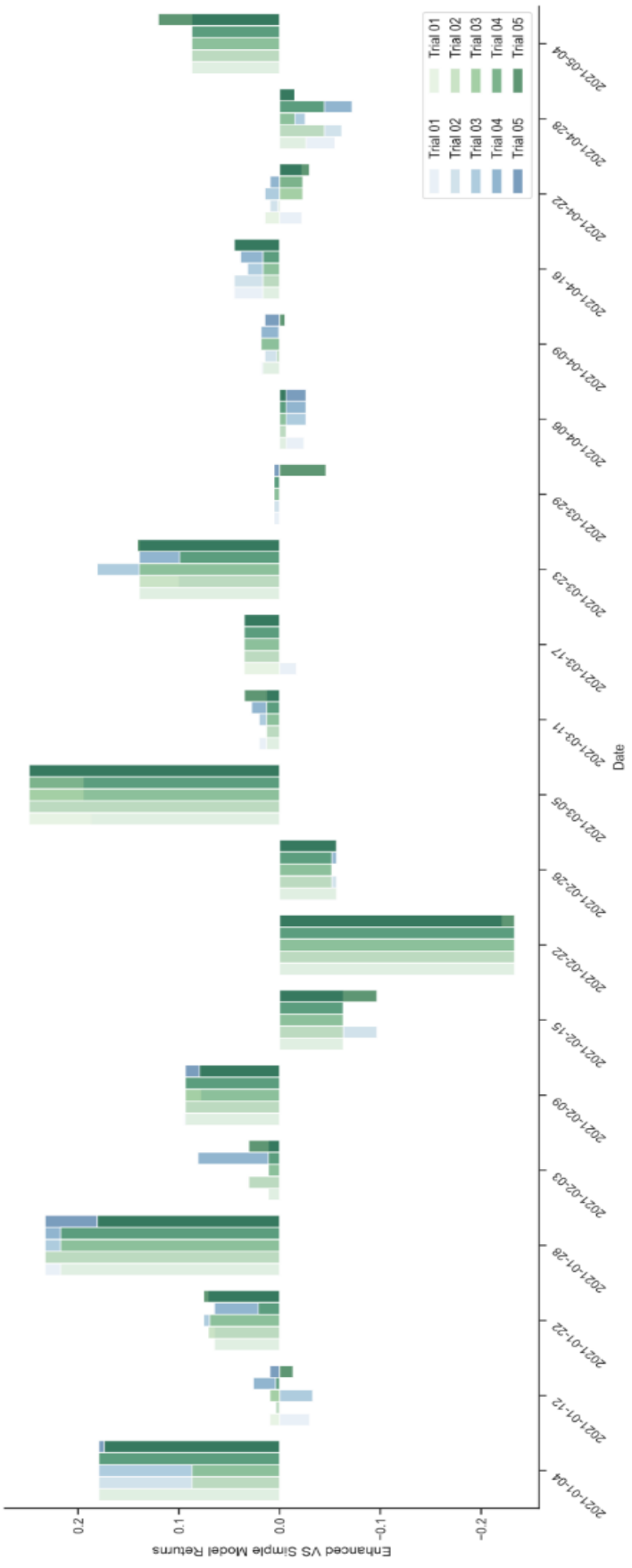
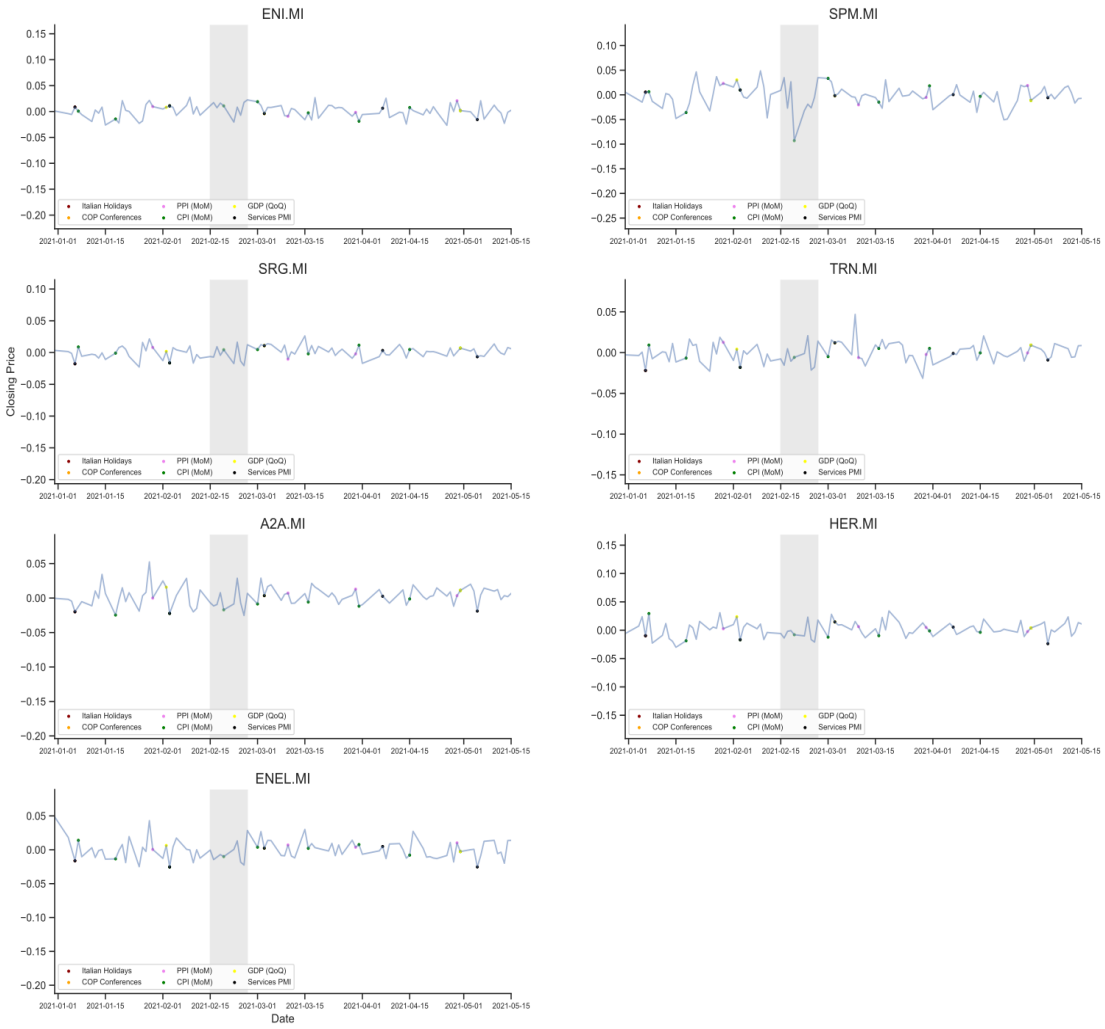


Figure 31 provides a visual comparison of the daily returns obtained using the automated trading tool based on technical indicators (blue bars) and technical indicators enriched with additional information (green bars), for each trading window and each trial. As it can be seen, the random forest algorithm built using the full dataset provides better performances, even if in some trading windows such as 2021-02-22 both models struggle in identifying negative returns. The trading windows between 2021.02.15 and 21.02.28 can be analyzed in-depth by observing the actual daily returns for each stock. Figure 32 shows the daily returns for the entire testing period and the grey shadowed area represents the time frame between February 15th and February 28th. Moreover, the colored dots indicate if on a particular day an economic indicator was released.

Figure 32 – Daily Returns, Test Trading Windows

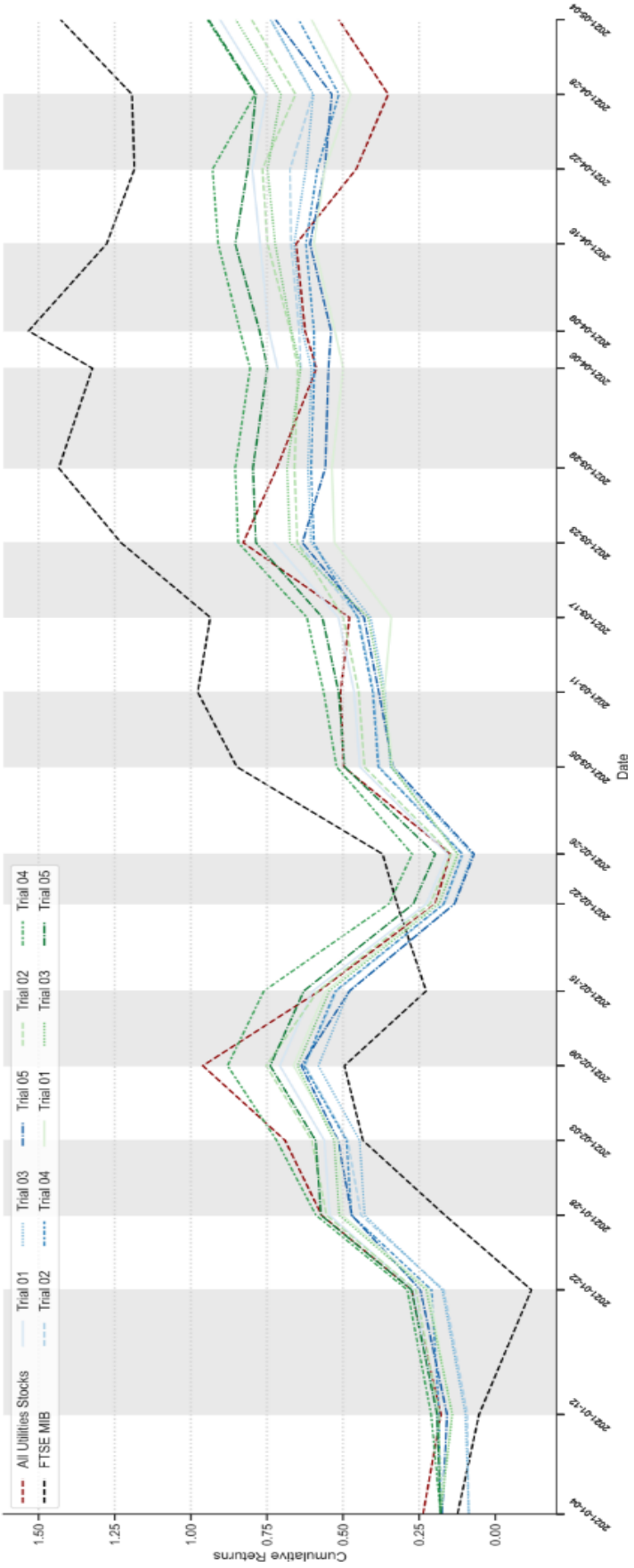


It can be noticed that on February 19th the CPI (MoM) index was released and its growth was lower than expected, + 0.1% instead of the forecasted + 0.5%. Looking at the ISTAT consumer prices notes released on March 16th, 2021, it is stated that:

"The slight speed up of All items [CPI] index was mainly due to the less amplitude of the decrease of prices of Non-regulated energy products (from -6.3% to -3.6%) […]. Core inflation (excluding energy and unprocessed food) and inflation excluding energy slightly sped up to +0.9% for both (from +0.8% in the previous month). The increase on monthly basis was mainly due to the prices of Non-regulated energy products (+1.4%), […]" (ISTAT, 2021)

It is now possible to get a grasp of the reason why the algorithms failed to predict the actual returns in late February 2021. Tensions and concerns about the CPI index and the growing prices of non-regulated energy products are mirrored in the utility sector stock prices. Taking as an example of Saipem SpA it is relevant the February 19th daily return sudden plunge. Finally, a visual representation of the cumulative returns of the two automated trading systems against the chosen benchmarks is given in Figure 33. The blue lines represent, for each trial, the cumulative returns of the portfolio built using only technical indicators, while the green lines represent the cumulative returns of the portfolio built using also additional information. At the end of the period, both portfolios outperform the portfolio built using all the stocks grouped in the utility sector, consistently in each trial.

Figure 33 – Cumulative Daily Returns, Final Comparison



Specifically, at the end of the 20th trading window, the cumulative return of the full-sector portfolio reaches +0.51%, while the cumulative returns of the portfolio built by the system using the "simple" and "enhanced" model reach respectively +0.74% and +0.83%. This means that, on average, the simple model portfolio performs 45% better than the full-sector portfolio while the enchanted model portfolio performs almost 63% better. This result can be expressed in annual terms, as shown in Table 16, where the simple model portfolio and the enchanted model portfolio perform respectively 73% and 106% better than the full-sector portfolio.

Table 16 – Annualized Returns

Sector Annualized Return	Simple Model Annualized Return	Enhanced Model Annualized Return
2.47	4.29	5.11

A different perspective of the same result can be expressed by taking into account Shape and Sortino Ratios, computed over the whole testing period. In practice, R_a consists of the average return of each portfolio over the twenty trading windows, R_f is the average risk-free rate of return, σ_a and σ_d are respectively the standard deviation of the portfolio returns and the standard deviation of the portfolio downside returns. Table 17 summarizes the resulting ratios obtained replacing those values according to Equation 27 and Equation 28.

Table 17 – Sharpe and Sortino Ratios

	Sector Portfolio	Simple Model Portfolio	Enhanced Model Portfolio
Sharpe Ratios	0.02	0.24	0.30
Sortino Ratios	0.07	0.75	0.98

Both for Sharpe and Sortino ratios, higher values are preferred when comparing similar portfolios. This means that the portfolio built using the "enhanced" algorithm is preferred over the portfolio built using the "simple algorithm" and the portfolio that contains all the stocks belonging to the energy and utility sectors and this is true for both the ratios.

Conclusions

This final chapter aims to resume the main findings of this study and provide a direction for further research. It can be safely stated that the application of random forest algorithms in developing automated trading systems using at first technical indicators and subsequently additional information such as economic indicators release dates have proven to be successful and profitable when applied to the Italian utility and energy sectors. In fact, portfolios built following the indications of the automated system provided significant improvements in the expected returns when compared to the portfolio containing all the stocks of firms belonging to these specific sectors. Specifically, when looking at the compound annual growth rates, even the portfolio built taking advantage of the simpler model performed on average 73% better than the full-sector portfolio. The most valuable result is obtained when deploying the system based on the enhanced model since the portfolio built following its predictions performed on average 106% better than the full-sector portfolio. These results are confirmed also when comparing Sharpe and Sortino's ratios computed on each portfolio. The ratios of the portfolio built by the system based on the enhanced model are the highest, followed by the ratios of the portfolio built by the system based on the simple model, and finally the ratios of the full sector's portfolio. Future research could include other types of data, such as fundamental data or investors' expectations and psychological thinking, from which the sentiments can be derived. In fact, it has been extensively proven that their trading behavior has a substantial effect on influencing price changes in financial markets. Finally, further research is needed in finding instruments or predictors that may help the automated trading systems in facing sudden and potentially disruptive events, to limit possible losses, and consequently improve average returns on investments. The difficulties in identifying negative returns caused by the market reaction to the unexpected month-over-month change of the Italian consumer price index released on February 19th, 2021 is a clear example of this necessity.

Bibliography

Alya Al Nasseri, A. T. (2015). Quantifying StockTwits semantic terms' trading behavior in financial markets: An effective application of decision tree algorithms. *Expert Systems With Applications*, 9192–9210. doi:http://dx.doi.org/10.1016/j.eswa.2015.08.008

Ash Booth, E. G. (2014). Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications*, 3651-3661. doi:http://dx.doi.org/10.1016/j.eswa.2013.12.009

B.V., T. M. (2022, 5 31). *Euribor*. Tratto da Euribor Rates: https://www.euribor-rates.eu/en/

Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications, 42*(20), 7046-7056. doi:https://doi.org/10.1016/j.eswa.2015.05.013

Berkeley, U. (2020, 6). *What Is Machine Learning (ML)?* Tratto il giorno 6 10, 2022 da ischoolonline.berkeley.edu: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

Board of Governors of the Federal Reserve System. (2022, May 4). *3-Month Treasury Bill Secondary Market Rate [DTB3]*. Tratto da FRED, Federal Reserve Bank of St. Louis: https://fred.stlouisfed.org/series/DTB3

Breiman, L. (1996). Bagging Predictors. (K. A. Publishers, A cura di) *Machine Learning, 24*, 123-140.

Breiman, L. (2001, October). Random Forests. *Machine Learning, 45*(1), 5–32. doi:https://doi.org/10.1023/A:1010933404324

C. Jensen, M. (1978). Some anomalous evidence regarding market efficiency. *Journal of Financial Economics, 6*(2-3), 95-101. doi:https://doi.org/10.1016/0304-405X(78)90025-9

Claude Sammut, G. I. (2010). *Encyclopedia of Machine Learning.* Boston, MA: Springer. doi:https://doi.org/10.1007/978-0-387-30164-8

Dev Shah, H. I. (2019). Stock Market Analysis: A Review and Taxonomy of Prediction Techniques. *Int. J. Financial Stud., 7*(2), 26.

Fabian Pedregosa, G. V. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825--2830. Tratto da http://jmlr.org/papers/v12/pedregosa11a.html

Fama, E. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance, 25*, 383–417.

Fama, E. F. (1965). Random Walks in Stock Market Prices. *Financial Analysts Journal, 21*(5), 55-59.

French, K. R. (1980). Stock returns and the weekend effect. *Journal of Financial Economics, 8*(1), 55-69. doi:https://doi.org/10.1016/0304-405X(80)90021-5

FTSE Russell. (2022). *FTSE MIB Index Fact Sheet.* FTSE Russell. Tratto il giorno April 29, 2022 da https://research.ftserussell.com/analytics/factsheets/Home/DownloadConstituentsWeights/?indexdetails=FTSEMIB

Granville, J. E. (1963). *Granville's New Key to Stock Market Profits.* Papamoa Press.

Guyon, I. W. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning, 46*, 389–422. doi:https://doi.org/10.1023/A:1012487302797

Hossin, M., & Sulaiman, N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process, 5*(2), 1. doi:10.5121/ijdkp.2015.5201

Hu, Y. K. (2015). Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing, 36*, 534-551. doi:https://doi.org/10.1016/j.asoc.2015.07.008Get

ISTAT. (2021, 3 16). *CONSUMER PRICES*. Tratto il giorno 5 30, 2022 da Archivio ISTAT: https://www.istat.it/it/archivio/255123

Lane, G. C. (1984). Lane's Stochastic. *Stocks & Commodities, 2:3*, p. 87-90. Tratto il giorno 06 15, 2022

Leo Breiman, J. H. (1984). *Classification And Regression Trees* (1st Edition ed.). New York: Routledge. doi:https://doi.org/10.1201/9781315139470

Li, L. (2017). Research on Machine Learning Algorithms and Feature Extraction for Time Series. *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 1-5.

Mantovani, R. G., Horváth, T., Cerri, R., Junior, S. B., Vanschoren, J., & Carvalho, A. C. (2018). An empirical study on hyperparameter tuning of decision trees. *CoRR, abs/1812.02207*, 36. doi:https://doi.org/10.48550/arXiv.1812.02207

McKinney, W. (2010). Data Structures for Statistical Computing in Python. (S. v. Millman, A cura di) *Proceedings of the 9th Python in Science Conference*, 56 - 61. doi:10.25080/Majora-92bf1922-00a

Mehar Vijha, D. C. (2020). Stock Closing Price Prediction using Machine Learning Techniques. *Procedia Computer Science*, 599–606. doi:10.1016/j.procs.2020.03.326

Muh-Cherng Wu, S.-Y. L.-H. (2006). An effective application of decision tree to stock trading. *Expert Systems with Applications*, 270-274. doi:10.1016/j.eswa.2005.09.026

Murphy, J. J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications.* New York Institute of Finance.

Oxford, U. (2022, 5 24). *Overfitting*. Tratto da Lexico: https://www.lexico.com/definition/overfitting

Pavan Kumar Illa, B. P. (2022). Stock price prediction methodology using random forest algorithm and support vector machine. *Materials Today: Proceedings*, 1776-1782. doi:https://doi.org/10.1016/j.matpr.2021.10.460

Peter F. Christoffersen, F. X. (2006). Financial Asset Returns, Direction-of-Change Forecasting, and Volatility Dynamics. *Management Science, 58*(2), 1273-1287. doi:https://doi.org/10.1287/mnsc.1060.0520

Pring, M. J. (2014). *Technical Analysis Explained* (5th edition ed.). McGraw Hill.

Quinlan, R. (1992). *C4.5: Programs for Machine Learning* (1st edition ed.). Morgan Kaufmann.

Rogalski, R. J. (1984). New findings regarding day-of-the-week returns over trading and non-trading periods: a note. *The Journal of Finance, 39*(5), 1603-1614. doi: https://doi.org/10.1111/j.1540-6261.1984.tb04927.x

Rouf, N., Malik, M., Arif, T., Sharma, S., Singh, S., Aich, S., & Kim, H.-C. (2021). Stock Market Prediction Using Machine Learning Techniques: A Decade Survey on Methodologies, Recent Developments, and Future Directions. *Electronics*, 10, 2717. doi:https://doi.org/10.3390/electronics10212717

Russell, F. (2022). *Industry Classification Benchmark v3.9.* FTSE Russell. Tratto il giorno 06 13, 2022

Sharpe, W. F. (1994). *The Sharpe Ratio*. (S. University, A cura di) Tratto il giorno 5 30, 2022 da web.stanford.edu: https://web.stanford.edu/~wfsharpe/art/sr/sr.htm

Suryoday Basak, S. K. (2019). Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance, 47*, 552-567. doi:https://doi.org/10.1016/j.najef.2018.06.013

The pandas development team. (2020). Pandas. *Zenodo*, V. 1.4.2. doi:10.5281/zenodo.3509134

Trading Calendar Archive. (2022, May 4). Tratto da Borsa Italiana: https://www.borsaitaliana.it/borsaitaliana/calendario-e-orari-di-negoziazione/archivio-calendari-di-borsa.en.htm

Trading Economics. (2022, May 4). *Trading Economics Calendar*. Tratto il giorno May 4, 2022 da Trading Economics: https://tradingeconomics.com/calendar

Trevor Hastie, R. T. (2009). *The Elements of Statistical Learning* (2nd Edition ed.). Springer Nature.

Wikimedia Foundation, I. (2022, 05 13). *Sensitivity and Specificity*. Tratto il giorno 05 30, 2022 da Wikipedia: https://en.wikipedia.org/wiki/Sensitivity_and_specificity#/media/File:PPV,_NPV,_Sensitivity_and_Specificity.svg

Y. Abu-Mostafa, A. A. (1996). Introduction to financial forecasting. *Applied Intelligence*, 205-213. doi:10.1007/BF00126626

Yi, S. K. (2012). The Wisdom of the Crowd in Combinatorial Problems. *Cognitive Science, 36*(3), 452-470. doi:https://doi.org/10.1111/j.1551-6709.2011.01223.x

Zhong, X. a. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications, 67*, 126–39. doi:https://doi.org/10.1016/j.eswa.2016.09.027