UNIVERSITA' DEGLI STUDI DI PADOVA

DIPARTIMENTO DI SCIENZE ECONOMICHE ED AZIENDALI "M.FANNO"

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA MAGISTRALE IN
ECONOMICS AND FINANCE

TESI DI LAUREA

Machine Learning and Portfolio Optimization: an application to Italian FTSE-MIB Stocks

RELATORE:

CH.MO PROF. CLAUDIO FONTANA

LAUREANDO: ANDREA MASIERO

MATRICOLA N. 2004261

ANNO ACCADEMICO 2022 – 2023

Dichiaro di aver preso visione del "Regolamento antiplagio" approvato dal Consiglio del Dipartimento di Scienze Economiche e Aziendali e, consapevole delle conseguenze derivanti da dichiarazioni mendaci, dichiaro che il presente lavoro non è già stato sottoposto, in tutto o in parte, per il conseguimento di un titolo accademico in altre Università italiane o straniere. Dichiaro inoltre che tutte le fonti utilizzate per la realizzazione del presente lavoro, inclusi i materiali digitali, sono state correttamente citate nel corpo del testo e nella sezione 'Riferimenti bibliografici'.

I hereby declare that I have read and understood the "Anti-plagiarism rules and regulations" approved by the Council of the Department of Economics and Management and I am aware of the consequences of making false statements. I declare that this piece of work has not been previously submitted – either fully or partially – for fulfilling the requirements of an academic degree, whether in Italy or abroad. Furthermore, I declare that the references used for this work – including the digital materials – have been appropriately cited and acknowledged in the text and in the section 'References'.

Firma (signature) ……

Table of Contents

INTRODUCTION

Portfolio theory is one of the most discussed themes in literature, and a fundamental concept in modern finance that deals with the management of investments. The first scientific approach to portfolio optimization problems was developed by Harry Markowitz in the 1950s and it is widely regarded as one of the most important breakthroughs in the field of finance. Markowitz Optimization, also known as Modern Portfolio Theory, aims to maximize returns while minimizing risk by creating a well-diversified investment portfolio. The basic idea behind Markowitz Optimization is that investors should not put all their eggs in one basket, instead, they should spread their investments across a range of assets that have different levels of risk and return. This strategy reduces the risk of investing in a portfolio of businesses and offers an efficient diversification mechanism for loss mitigation by selecting the best stocks with the lowest combined risk. The method of portfolio optimization entails determining the participation of each stock, or its proportion in the portfolio value, that minimizes portfolio risk at a desired portfolio return or, in the case of the dual problem, maximizes portfolio return at a given risk. By varying the desired portfolio return (or risk) across all feasible values for optimum portfolios, the efficient frontier, that is a region in the risk-return space, is identified. Portfolios with the distinctive quality of being efficient are those that are inside the efficient frontier. The opportunity set does not contain any alternative portfolios that provide a greater return for a given level of risk or a lower risk for a given level of return.

The basic premise of the model is that in the time series of returns for each stock, the mean is used to estimate future returns, the variance to assess risk, and the covariance of each pair of time series to assess the aggregate risk of each pair of stocks. Since then, several more models have been created, all of which are based on the fundamental idea of the Markowitz mean-variance model. In all these models, which are now known as classic models, the expected return on the portfolio is determined by the linear combination of the stocks' weight in the portfolio and its expected returns (the mean returns). Although there are many other ways to determine a portfolio's risk, they are generally based on moments around the mean of a linear combination of participations and stock return time series. Despite the classic models of portfolio selection being widely used, real-world data has in many ways undermined their central premise: the series of returns frequently deviate from normality, showing kurtosis and

skewness, in this case the variance of the time series can't capture the true investment risk. Additionally, using mean returns as a forecast of future stock returns has a negative impact on the performance of the models for short-term investments because it has a low-pass filtering effect on the stock markets' dynamic behavior. This leads to imprecise estimates of future short-term returns.

The key word so far is diversification, that has a long history, dating back to the 16th century when merchants in Venice spread their investments across multiple ships to reduce the risk of losing everything on one voyage. After the development of Markowitz theories, in the 1960s, Sharpe (1964) introduced the capital asset pricing model (CAPM), which helped investors determine the expected return of an asset based on its level of risk. The CAPM reinforced the importance of diversification by showing that investors could reduce risk by investing in a portfolio of assets with different levels of risk. The Capital Asset Pricing Model (CAPM) has long been regarded as a fundamental pillar of modern finance, revolutionizing the way investors analyze and evaluate investment opportunities. CAPM provides a framework for understanding the relationship between risk and expected returns in financial markets. One of the main contributes of CAPM, is the distinction of systemic and specific risk.

Systemic risk refers to the risk of widespread disruptions or failures in the overall financial system, typically caused by events or factors that affect the entire economy or a significant portion of it. It is the risk that a disturbance or shock to the system could lead to a domino effect, triggering a cascade of financial distress and contagion across various institutions and markets. Systemic risk is often associated with events such as economic recessions, financial crises, or major policy changes that impact multiple sectors and regions. It arises from interconnectedness and interdependencies within the financial system, where the failure of one institution or market can have far-reaching consequences. Factors that can contribute to systemic risk include macroeconomic shocks, geopolitical events, interest rate changes, systemic banking failures, regulatory changes, and market-wide sentiment shifts. A significant market downturn, for example, can be considered a manifestation of systemic risk as it affects a wide range of stocks and investors.

Specific risk, also known as idiosyncratic risk or unsystematic risk, refers to the risks that are unique to individual stocks or companies and are uncorrelated to broader market movements. These risks are specific to a particular company or industry and can arise from company-specific factors, such as management decisions, operational performance, industry competition,

technological changes, or legal and regulatory issues. Specific risk can have a direct impact on the performance of individual stocks or a subset of stocks within a portfolio. Unlike systemic risk, specific risk is not correlated with market-wide events or factors. By holding a well-diversified portfolio, investors can mitigate the impact of specific risk on their overall investment returns. However, it is important to note that while diversification can reduce specific risk, it cannot eliminate all risk. Systemic risks can still influence the performance of diversified portfolios to some extent. Central to the CAPM is the concept of beta, which measures the sensitivity of an asset's returns to market movements. Beta serves as a proxy for an asset's systematic risk, indicating how much the asset's returns are expected to vary relative to the overall market. A beta of 1 implies that the asset's returns move in tandem with the market, while a beta greater than 1 suggests greater volatility, and a beta less than 1 indicates lower volatility than the market.

In the 1960s and 1970s, mutual funds and other investment vehicles made it easier for individual investors to create diversified portfolios. As a result, diversification became more accessible to a broader range of investors. In the 1980s and 1990s, the popularity of index funds, which track the performance of a broad market index, made diversification even more accessible to individual investors. Index funds offered a low-cost way for investors to create a diversified portfolio of stocks or bonds without having to do extensive research or pay high fees. Today, diversification is a cornerstone of modern portfolio management. Investors use a variety of techniques, such as asset allocation and sector rotation, to create diversified portfolios that can help maximize returns while minimizing risk. While diversification cannot eliminate all investment risk, it remains the fundament of almost every investment strategy.

These traditional portfolio construction methods often face limitations when it comes to incorporating subjective investor views and dealing with estimation errors in asset return inputs. Additionally, they tend to produce portfolios that are highly sensitive to small changes in input parameters, leading to unstable and suboptimal asset allocations. The Black-Litterman model, developed by Black & Litterman (1992), represents a significant advancement in portfolio construction methodology. This innovative approach combines the principles of Bayesian analysis, modern portfolio theory, and market equilibrium to provide investors with a more sophisticated and flexible framework for asset allocation. Recognizing the shortcomings of traditional approaches, Black and Litterman sought to develop a model that could address these challenges and enhance the practicality and robustness of portfolio construction. Their objective was to incorporate subjective investor views into the asset allocation process in a way that

balanced these views with market equilibrium considerations. At the heart of the Black-Litterman model lies the Bayesian framework, which offers a systematic approach for combining prior beliefs (investor views) with available market information. Bayesian analysis provides a formal mechanism to update and revise prior beliefs based on new evidence, resulting in posterior probabilities that reflect a more accurate representation of the underlying reality. By employing Bayesian techniques, the Black-Litterman model enables investors to incorporate their subjective views on expected returns, correlations, and risk preferences while simultaneously considering the collective wisdom of the market. This blending of subjective and market-based information helps to mitigate estimation errors and improves the stability of portfolio allocations. In the Black-Litterman model, the market equilibrium prior plays a crucial role. It represents the expected return and covariance structure of assets based on the assumption that financial markets are generally efficient and that prices reflect all available information.

The Efficient Market Hypothesis (EMH), developed by Fama (1970), states that financial markets are efficient in incorporating all available information into asset prices. According to the EMH, it is impossible to consistently outperform the market by using any publicly available information because stock prices already reflect all relevant information.
The EMH is based on three main forms of market efficiency:

1. Weak-Form Efficiency: Weak-form efficiency implies that current stock prices fully reflect all historical price and trading volume information.
2. Semi-Strong Form Efficiency: Semi-strong form efficiency suggests that stock prices fully reflect all publicly available information, such as financial statements, news releases, and market rumors.
3. Strong-Form Efficiency: Strong-form efficiency is the most robust form of the EMH. It posits that stock prices fully reflect all available information, including public and private information.

It's important to note that the EMH is not without criticism. Some argue that markets are not perfectly efficient due to behavioral biases, market frictions, information asymmetry, and other factors.

The incorporation of the market equilibrium prior helps anchors the portfolio allocations within the bounds of rational expectations and provides a reference point that prevents extreme deviations driven solely by investor views. By combining the market equilibrium prior with the investor's subjective views, the Black-Litterman model achieves a more balanced and realistic representation of asset returns and correlations. The resulting portfolio strikes a balance

between the investor's views and the market equilibrium, taking into account risk and return objectives. The Black-Litterman model offers several advantages over traditional portfolio construction approaches. It allows investors to incorporate their subjective views in a systematic and disciplined manner, enhancing the customization and personalization of portfolios. Furthermore, the model provides more stable and robust allocations by blending subjective views with market equilibrium considerations. The Black-Litterman model finds applications in various areas of finance, including asset management, institutional investing, and wealth management. It can be employed for strategic asset allocation, tactical asset allocation, and risk management purposes. The flexibility and adaptability of the model make it a valuable tool for investors seeking to achieve more efficient and tailored portfolio allocations. The Black-Litterman model represents a significant advancement in portfolio construction methodology, bridging the gap between subjective investor views and market equilibrium considerations.

In recent years, the integration of machine learning techniques in the financial markets has gained significant attention and has the potential to revolutionize various aspects of financial analysis, decision-making, and trading. Machine learning, a subset of artificial intelligence, offers powerful computational algorithms that can analyze large volumes of data, identify patterns, and generate predictions or insights that traditional approaches may overlook. Here are some key areas where machine learning has been integrated into the financial markets:

1. Trading Strategies and Predictive Models: Machine learning algorithms have been applied to develop trading strategies and predictive models. These models can analyze historical data, identify patterns, and generate signals for buying or selling assets. Machine learning techniques such as support vector machines, random forests, and neural networks are used to identify complex patterns in market data, news sentiment analysis, and other relevant information sources to generate trading signals with the aim of outperforming traditional strategies.
2. Risk Management and Fraud Detection: Machine learning is being utilized to enhance risk management and fraud detection in the financial industry. By analyzing vast amounts of data, machine learning algorithms can identify anomalies, detect fraudulent activities, and mitigate risks in real-time. These algorithms can identify patterns that may indicate potential fraudulent transactions or unusual market behavior, helping financial institutions take proactive measures to safeguard against fraud and manage risks effectively.
3. Credit Scoring and Lending Decisions: Traditional credit scoring models are being augmented or replaced by machine learning algorithms. By analyzing a broad range of

data, including credit history, financial statements, and alternative data sources, machine learning models can assess creditworthiness more accurately. These models help lenders make informed decisions, assess default risks, and improve the efficiency and accuracy of lending processes.

4. Portfolio Management and Asset Allocation: Machine learning techniques are being employed to optimize portfolio management and asset allocation decisions. These algorithms can analyze historical market data, factor in investor preferences and constraints, and generate optimal portfolios tailored to individual investors' risk-return objectives. Machine learning models can also incorporate sentiment analysis from news, social media, and other sources to gauge market sentiment and adjust portfolio allocations accordingly.
5. High-Frequency Trading and Market Microstructure: Machine learning algorithms are being applied in high-frequency trading (HFT) and market microstructure analysis. These algorithms can process vast amounts of market data, identify short-term patterns, and exploit market inefficiencies to execute trades with reduced latency. Machine learning can also aid in understanding market microstructure dynamics, including order book analysis, liquidity prediction, and market impact modeling.
6. Natural Language Processing and Sentiment Analysis: Natural Language Processing (NLP) techniques and sentiment analysis are used to extract information and sentiment from unstructured text data, such as news articles, social media posts, and financial reports. This information is leveraged to gain insights into market sentiment, news impact on asset prices, and sentiment-driven trading strategies.
7. Algorithmic Trading and Execution: Machine learning algorithms are employed in algorithmic trading and execution systems to optimize trade execution strategies. These algorithms analyze historical trade and order book data to model market impact, liquidity dynamics, and optimal order routing decisions. Machine learning can improve execution efficiency, reduce transaction costs, and mitigate market impact.

While the integration of machine learning in the financial markets offers significant potential, it also poses challenges. Data quality, overfitting, model interpretability, regulatory compliance, and ethical considerations are among the key concerns that need to be addressed to ensure the responsible and effective application of machine learning techniques in finance.

The aim of this thesis is to take the main idea behind the Black-Litterman model, integrating it with the modern machine learning techniques. The importance of Black and

Litterman publication lies on the relevance given to analyst "opinions" with respect to the stocks in the portfolios. In the Markowitz and CAPM models, the entire process of portfolio selection was automatic and mechanic: there is no value added from the investment manager. In the Black-Litterman model human analysis on the stocks are used as input and integrated on the optimization process. In the following chapters we will use the same framework, but we will replace human analysis with machine learning estimations. To sum up, we are going to use the machine learning technique to retrieve estimations on a stock basket and use those estimation to build an efficient portfolio. We will propose a portfolio composed of Italian FTSE-MIB shares, setting an investment time horizon of one year. To do so, we are going to build a model that can be divided in four different phases.

1. In the first phase we retrieve financial data of the FTSE-MIB shares, including daily closing price and trade volumes. We then calculated the daily values of six relevant indicators, like ATR and EMA, for every stock considered. We finally divided the dataset in train set, from 2000 to 2019, and test set, that is the entire 2019 year.
2. In the second phase we estimate, with an ARMA model, the value of the six indicators for the test set.
3. In the third phase we apply SVM to retrieve the daily stock returns for the test set, starting from the ARMA estimations of indicators.
4. In the last phase we use return forecast to build a stock portfolio, that we compare with a classic Markowitz portfolio in the performance.

The model has the fundamental premise that the efficient market hypothesis doesn't completely hold, and that is possible to use past data to predict future performance.

The thesis is structured as follow; chapter two reviews previous literature on the application of SVM to financial market, chapter three sheds light on the methodology used to develop the proposed model, in chapter four the model is applied to real data, and in chapter five we sum up our work in the conclusion.

CHAPTER ONE: LITERATURE REVIEW

In this chapter we are going to review past literature on the application of machine learning to portfolio building process and return forecasts. With this review we want to create a theoretical basis that justify our choice to use machine learning for our forecasts. We decided to include literature on SVM technique, that we are going to use in the following chapters, comparing it with other machine learning techniques.

Ma et al., 2021 propose a novel approach to portfolio optimization by integrating deep learning and machine learning methods for accurate return prediction. The main objective of the study is to develop a framework that leverages deep learning techniques to predict asset returns based on historical financial data. To enhance the return prediction, the authors extract various financial features, including technical indicators, market sentiment, and macroeconomic factors, and use them as inputs to the deep learning models. The paper explores traditional machine learning algorithms, in particular support vector regression (SVR) and random forest regression (RFR), and three deep learning techniques, that are deep multilayer perceptron (DMLP), long short-term memory (LSTM) neural network and convolutional neural network (CNN). The proposed portfolio optimization framework utilizes the predicted returns from the machine learning models to construct an optimal portfolio allocation. The objective is to maximize the expected return while considering risk measures such as variance or downside risk. The authors evaluate the effectiveness of their approach using performance metrics like Sharpe ratio, cumulative return, and maximum drawdown. They compare the results with traditional portfolio optimization methods to demonstrate the superiority of the integrated deep learning and machine learning approach. The model has been applied to Chinese stock market, using historical data of 9 years from 2007 to 2015 of component stocks of China securities 100 index. An ARIMA model to predict returns has been used as benchmark. The empirical results show that the proposed approach outperforms traditional methods in terms of return prediction accuracy and portfolio performance. By combining deep learning and machine learning techniques, the authors achieve improved accuracy in return prediction, capturing complex patterns and dependencies in the data. The study highlights the potential benefits of using these advanced techniques in the portfolio optimization. The limitation highlighted by the authors is the portfolio turnover, that may considerably erode the profit of the strategy.

Another application of support vector machine on financial investments and portfolio selection have been proposed by Gupta et al. (2012). The author's aim is to divide financial asset (stocks)

in three asset classes, based on three criterions: risk, return and liquidity, based on four financial criteria, that are the short-term return, the long-term return, the risk and the liquidity. Liquidity is considered in terms of the probability of conversion an investment into cash without any significant loss in value. They implemented real codec genetic algorithm to build an optimal portfolio, starting from the previously created asset classes. They applied the model to Mumbai National Stock Exchange, that has 150 stocks listed. The high classification accuracy of 91% promised by the SVM classifier created in this study is impressive. The key benefit of the suggested method is the ability to categorize any collection of assets into the appropriate classes after obtaining a suitable classifier. This gives investors a basic understanding of the asset class and aids in their decision-making on the best possible investment options. The investors can then select from among these options by utilizing RCGA to build the appropriate portfolio. The benefit of employing RCGA is that it is possible to answer the portfolio selection problem in its original form without having to linearize the risk objective. The results indicate that the approach proposed is capable of classifying assets with good accuracy and further provides optimal portfolios for each class of assets based on the investor-preferences regarding the financial criteria used.

The stock selection problem has been faced by Fan & Palaniswami (2001). Authors focused their paper on Australian stock Exchange, for the period of 1992-2000. They examined financial indicators, as net income on total capital or sales growth, extrapolated from the annual reports that listed company must disclose. Similar financial indicators have been grouped into eight categories, that are the input vector of the support vector machine model. They trained the SVM to group the stocks in two categories: performance stocks, that are the stocks in the top 25% ranking for annual returns and the non-performance stocks, that are the remaining 75%. Authors then compared the performance of the portfolio composed of the predicted performance stocks, with the equally weighted portfolio for a following period of five years. The performance stocks portfolio had a return of 207%, outperforming the equally weighted portfolio of 71%. Of course, the described methodology cannot be used as a valuable investment strategy, since stock performance variance, and thus risk for the investor, has not be considered in the model. However, the aim of the authors was to prove the effectiveness of the SVM model also when it's applied to the stock market, and, in particular, to the stock performance.

Gururaj & R (2019) compared linear regression with SVM in the prediction of stock prices with matching prerequisites. The dataset is composed of Coca-Cola Company stock details from 2017 to 2018, acquired from Quandl. A root mean square error (RMSE) of 3.22, a mean

absolute error (MAE) of 2.53, a mean square error (MSE) of 10.37, and a correlation coefficient (R) of 0.73 were observed for linear regression, whereas for SMV, the following values were observed: 1.58 for RMSE, 1.33 for MAE, 2.51 for MSE, and 0.93 for R-squared. Thereby proving that SVM performs better than the Linear Regression model.

Sadia et al. (2019) aim is to find the best model for stock market forecasting and how to get the most out of it by combining several strategies. Their paper's primary goal was to choose the appropriate database. With the aid of 11 distinct parameters, they worked on it. SVM and RFs (Random Forest classifier) were used on the Kaggle dataset, and the results showed that SVC (Support Vector Classifier) produced an accuracy of 78.7% while RFS produced an accuracy of 80.8%. They concluded that they are among the finest algorithms for stock prediction based on these datasets, and that brokers and investors can benefit from them. In order to achieve more accuracy, adding additional options may require removing many restrictions.

In Hargreaves et al. (2017) Support Vector Machine was evaluated to forecast stock prices for the large and small capitalizations stocks and in three different stock markets, using prices with both daily and hourly frequency. The main goal of this study was to gather a large amount of data from several global financial markets to forecast their daily movements more effectively. In this work, they use the four characteristics of value instability, value force, segment unpredictability, and area energy to forecast stock value direction. Their model shows promising results in predicting daily stock price direction.

In order to determine the best AI algorithm, Zheng & Jin, (2017) evaluated the impact of applying several AI algorithms to time series data. To predict the short-term prices of one particular company, "MSFT" (Microsoft), they used Logistic Regression, Bayesian Network, Simple Neural Network, and SVM with RBF kernel. SVM with RBF (Radial Base Function) kernel fared the best. The SVR was trained on 5% of the dataset due to computational limitations, yet it still outperformed the Linear Regressor. This indicates that SVM's may be utilized in real-world for stock forecasting.

Patil et al. (2016) wanted to overcome the inaccuracy problems of the standard prediction models. According to the study, SVM, has the essential properties for the forecasting task. Their empirical work is composed of two phases: in the first one they downloaded and pre-processed the dataset, which contained starting and ending daily price values for stocks. Then, in GUI, they applied SVM to stock price data to forecast future value of that stock. According to their

results, SVM showed 100% accuracy in their training phase and more than 90% accuracy in their testing phase. They determined that SVM performs well on large datasets, and it doesn't show any overfitting problem. Among the different benchmark strategies considered in the paper for prediction, the SVM model delivers more profit compared to benchmarks.

Pan et al. (2016) worked on Data Normalization on a stock market prediction by using SVM and Technical Indicators. SVM applies a structural risk minimization concept compared to the empirical risk minimization principle implemented by numerous other algorithms, including ANN. Using structural risk minimization concept, they work toward minimizing the upper bound of generalization error compared to minimizing empirical error. The authors also noted that SVM might be employed globally because of its improbable propensity to overfit. Hence, before working upon datasets, data Normalization is carried out, which scales down the transformation of attributes, decreasing the value magnitude to shallow values. Various Normalization techniques are being used, which include Min–Max Normalization, which scales down the values between [0, 1] or [− 1,1], Z-Score Normalization, which scales down the values by using Mean and Standard Deviation, Decimal Scaling Normalisation which scales down the values by moving the decimal point of that value and Median and Median Absolute Deviation (MMAD) Normalization which is comparable to Z Normalisation but it uses median and median absolute deviation for rescaling. They employed a radial basis function kernel to generate SVM models. They determined that by utilizing SVM, the model produced the best results compared to the other techniques. With data Normalization, they achieved a bit more accuracy compared to other techniques without Normalization.

Di, (2014) built an SVM model to forecast short term stock values (1–10 days). The author analyzed APPL, AMZN, and MSFT stocks historical data to anticipate the immediate future's closing values. The model was trained using technical indicators retrieved from the data as the features. The results demonstrate that with APPL stocks, the model showed a prediction accuracy of 70% and above. AMZN stocks were prone to volatility in the near term, but the model could perform well in the long term. MSFT was in the middle; the most accurate predictions were between 5-days and 10-days in the future. On average, the SVM model properly forecasts stock prices for one-day in the future with 56% accuracy and an accuracy of 70% for the next couple of days, illustrating the model's capacity to precisely forecast the trend.

Karazmodeh et al. (2013) worked on Particle Swarm Optimization (PSO) Improved via Genetic Algorithm (IPSO) based on Support Vector Machines (SVM) for efficient prediction of various

market indices. They stated that the risk minimization was done by SVM as its structural in nature of the model. They demonstrated that IPSO SVM (improved particle swarm optimization) outperformed PSO SVM for the same stocks, with an average hit ratio of 64.012% (for three equities: DJI, S&P 500, and NASDAQ). They achieved superior outcomes and precision as a result of the particle mutation. Additionally, the accuracy of this model may be increased by adding political and economic factors.

Hu et al. (2013) evaluated an SVM on the data of numerous firms, retrieved from the Federal Reserve Bank, Big Charts Historical Stock Quotes, and each company's annual report. The purpose of the article was to illustrate the superiority of SVM compared to prior traditional predictive regression models and to highlight the predictive potential of SVM on previously unexplored data. The data was separated into a training set and a test set, comprising 78 sample entries and 10 sample entries, respectively. The SVM input features comprised company-specific variables (i.e., net revenue, net income, price per earnings ratio of stock, etc.) and six macroeconomic variables (i.e., consumer investment, consumer spending, etc.). Regression models failed to provide correct predictions since this problem is not linearly separable, while SVM demonstrated an accuracy of 97.43% on training sets and 70% on test sets, inferring that SVMs outperform other conventional techniques in anticipation of financial exchange. The authors also advise using multivariate statistics to examine the connection between a company's success and the direction of the stock market.

Shen et al. (2012) attempted to ascertain the opening value of the American stock market, starting from the data from foreign stocks and commodities markets. The value of a given stock market (in this case, the US stock market) is significantly influenced by the value of the global stock market. Experiments on single feature predictions show a substantial link between features like the DAX and Australian dollars (AUD). Long-term forecasts exhibit a similar trend in prediction accuracy. The SVM used showed a prediction accuracy of up to 74% when using multi-feature prediction, with an RMSE of 21.6. Overall, the article demonstrates that, among the numerous models employed, SVM provides an exceptional ability for speculating and precisely determines the covert relationship between global stock prices and the US securities market.

In order to forecast the direction of daily stock price movement in the Korea Composite Stock Price Index (KOSPI), Kim (2003) employed SVM. In this investigation, 12 technical indicators were chosen. The indicators include momentum, ROC, Williams' R, A/D oscillator, disparity5,

disparity10, OSCP, CCI, RSI, Stochastic K, Stochastic D, Stochastic slow D, and momentum. Additionally, this study compared SVM to back-propagation neural networks (BPN) and case-based reasoning (CBR) in order to assess the viability of using SVM in financial prediction. According to experimental findings, SVM is a more effective stock market forecasting method than BPN and CBR.

Kumar & M. (2011a) utilized SVM and random forest to forecast the daily direction of the S&P CNX NIFTY Market Index of the National Stock Exchange and compared the results with those of the standard discriminant and logit models and ANN. In their investigation, they employed the same technical indicators as input variables applied by Kim (2003). The experimental findings indicated that SVM outperformed random forest, neural network and other classic models.

Huang et al. (2005) in their work, explored the predictability of financial movement direction with SVM by forecasting the weekly movement direction of the NIKKEI 225 Index. To test the prediction capacity of SVM, they compared its performance with linear discriminant analysis, quadratic discriminant analysis, and Elman backpropagation neural networks. The results of the experiment indicated that SVM outperformed the other categorization methods.

Kumar & M. (2011b), in an extension of their previous study studied the efficacy of ARIMA, ANN, SVM, and random forest regression models in forecasting and trading the S&P CNX NIFTY Index return. The performance of the three nonlinear models and the linear model is tested statistically and financially via a trading experiment. The empirical findings revealed that the SVM model is able to outperform other models employed in their investigation.

Hsu et al. (2009) created a two-stage architecture by merging a self-organizing map with support vector regression for stock price prediction. They evaluated seven important stock market indexes. The results showed that the two-stage design provides a feasible option for stock price prediction.

To ascertain the feasibility of SVM in financial forecasting, in Cao & Tay (2003) a comparison is made between SVM, the multilayer back-propagation (BP) neural network, and the regularized radial basis function (RBF) neural network. Experimental investigations are conducted to analyze the performance variability of SVM concerning its free parameters. To incorporate the nonstationary of financial time series into SVM, adaptive parameters are

proposed. The study employs five real futures contracts obtained from the Chicago Mercantile Market as the datasets. The simulation results reveal that among the three methods, SVM surpasses the BP neural network in financial forecasting. The empirical data suggest that SVM forecasts are significantly better than the BP network, while the regularized RBF neural network and SVM exhibit comparable performance. The regularized risk function, as opposed to the empirical risk function utilized by the BP neural network, is minimized by both SVM and the regularized RBF neural network, explaining the reason behind the comparable performances. As a result, they may withstand overfitting and finally outperform the BP neural network in terms of generalization.

In Lee (2009), the author developed a prediction model for stock market trends using a support vector machine (SVM) and a hybrid feature selection method. The hybrid method, called F-score and Supported Sequential Forward Search (F_SSFS), combines the strengths of filter methods and wrapper methods to select the optimal subset of features from the original set. To assess the prediction accuracy of the SVM-based model with F_SSFS, the researcher conducted a comparison with a back-propagation neural network (BPNN) and three commonly used feature selection methods: Information gain, Symmetrical uncertainty, and Correlation-based feature selection. The performance evaluation employed a paired t-test. The grid-search technique with 5-fold cross-validation was utilized to determine the best parameter value for the SVM kernel function. The study demonstrates that SVM outperforms BPNN in predicting stock market trends. Furthermore, the experimental results indicate that the SVM-based model combined with F_SSFS achieves the highest accuracy and generalization performance compared to the other three feature selection methods. Based on these findings, the researchers suggest that SVM combined with F_SSFS holds promise as an effective addition to existing methods for stock market trend prediction.

Dunis, Rosillo, et al. (2013) aims to explore the application of support vector machines (SVMs) in forecasting the weekly change in the Madrid IBEX-35 stock index. The data analyzed span from October 18, 1990, to October 29, 2010. To enhance statistical efficiency and incorporate economic performance measures, a trading simulation is conducted. The study focuses on traditional technical trading rules commonly used in equity market analysis, specifically the Relative Strength Index (RSI) and the Moving Average Convergence Divergence (MACD) decision rules. SVMs are employed using predetermined values of the RSI and MACD indicators to determine optimal buying or selling situations in the market. The SVM outputs encompass both the market direction and the associated probability for each forecasted market

move. The model is benchmarked with two traditional methods (the Multilayer Perceptron neural network and the Buy and Hold strategy). The optimal scenario for SVM trading outcomes is a 100% hit ratio when a 90% chance of occurrence is chosen. SVMs outperform neural networks and the buy-and-hold strategy in the analysis for the period and market chosen. Additionally, a variety of different investors can utilize the model since the SVM optimization procedure allows the user to choose their own level of risk aversion.

In Dunis, Likothanassis, et al. (2013), a hybrid Genetic Algorithm (GA) and Support Vector Machine (SVM) strategy was developed for the job of forecasting and trading the daily and weekly returns of the FTSE 100 and ASE 20 indexes. In particular, GA is used to improve the SVM parameters and discover the best feature subset. Furthermore, the suggested hybrid technique incorporates a problem-specific fitness function, which is thought to create more lucrative prediction models. The suggested method's outcomes were benchmarked with a HONN (Higher-Order Neural Network), a Naïve Bayesian Classifier, an ARMA, a MACD, plus a naïve and a buy and hold strategy. More precisely, the trading and statistical performance of all models were studied in a forecast and trading simulation on the daily and weekly FTSE 100 and ASE 20 time series spanning the period January 2001–May 2010, utilizing the past 18 months for out-of-sample testing. Only autoregressive terms were used as input for all the forecasting models. Empirical findings show that, in terms of accurate directional change, annualized return, and information ratio after transaction costs, the suggested technique overperformances all its benchmarks. When leverage is given to the models, these results are confirmed.

CHAPTER TWO: THE METHODOLOGY

This chapter will illustrate the methodology that we used to achieve the results. A model that combines econometric ARMA model with new machine learning techniques will be developed to build an efficient portfolio, composed of Italian FTSE-MIB stocks. The goal of this portfolio is to over-perform a benchmark portfolio obtained throw traditional Markowitz optimisation. The procedure to obtain the efficient portfolio starts from some stock technical indicators calculated for every stock considered; the future value of those indicators is then estimated through the univariate ARMA model. Those estimations, are then used as input for a particular machine learning technique, called SVM, to estimate future return of the stocks. This output is then used in the optimization process to come up with an efficient portfolio. The chapter is organized as follow. Section 3.1 presents the indicators used in the model and how they have been extracted from the starting dataset. Section 3.2 introduces the ARMA econometric model and its application on the data of this thesis. Section 3.3 is dedicated to the machine learning part of the model, and it shows how we used it to estimate future returns of the stock market. Finally, section 3.4 presents the portfolio building process.

2.1 Stock Indicators

Technical indicators are powerful tools used by traders and investors to analyse stock price movements. They are function of the stock opening price, closing price and volume of trades time series, that provide insight into market trends, momentum, and volatility. Technical analysis is based on the premise that historical price and volume data can reveal patterns and signals that can be used to make informed trading decisions. The hypothesis behind the use of these indicators in the model is that their value over time is correlated with the stock price evolutions.

Despite there is a great variety of technical indicators, it is possible to classify them in three main categories, depending on their use. There are the volatility indicators, that describe the volatility of the price of a stock, the trend/momentum indicators, used to analyze the magnitude and the strength of the variation of the price over time, and, finally, the performance indicators,

that give an insight on the pure price evolution over time. In this model 6 of the most widespread indicators has been used, chosen from each of the three categories. They are listed below.

ATR

The Average True Range (ATR) is a technical analysis indicator that measures the volatility of a financial instrument, such as a stock, commodity, or currency pair. It was developed for the first time by Wilder (1978). Unlike traditional risk indicators, the ATR calculation is not based on the variance or the standard deviation, but it is based on the so-called ranges. Ranges can be defined as the difference between the highest and lowest prices reached by a stock in a given period of time. In particular, the ATR is a moving average of the previous 14 period (days) true ranges. True ranges are a modified version of the ranges, calculated as the largest of:

- The difference between the highest and lowest prices reached by the stock price on the reference day;
- The absolute value of the difference between the highest price reached by the stock on the calculation day and the closing price of the previous day;
- The absolute value of the difference between the lowest price reached by the stock on the calculation day and the closing price of the previous day.

The main advantage of considering true ranges instead of simple ranges is to avoid an underestimation of the variance. Simple ranges only consider the price fluctuation over a working day, but one must also consider the gap between the opening and closing price. This gap contributes to the variance of the stock price, and it is captured only by the true ranges.

The combination of this indicator and machine learning algorithms to make investment decisions is not a new theme in financial literature, and it was explored by Zbikowski (2015). In this paper, the author uses the average true ranges, along with the support vector machine to evaluate different short-term trading strategies. The paper excludes that this kind of strategy can be remunerative for the investors, however the research is limited only to short-term investment, and the author suggests that more research should be done when a longer period of time is considered.

The true range (TR) is calculated for each day of the time horizon as:

$$TR_t = max(high_t, close_{t-1}) - \min(\mathrm{low_t}, \mathrm{close}_{t-1})$$

Where:

- *"$High_t$"* is the highest price reached by the price variable during the day *"t"*
- *"Low_t"* is the lowest price reached by the price variable during the day *"t"*
- *"$Close_{t-1}$"* is the closing price of the day *"t-1"*.

The following step is to calculate the first ATR of the period:

$$ATR_1 = \frac{1}{n} * \sum_{i=1}^{n} TR_i$$

Where *n* is the total number of true ranges calculated at the previous point. Finally, the ATR is calculated for all other days:

$$ATR_t = \frac{ATR_{t-1} * (n-1) + TR_t}{n}$$

It is possible to calculate the ATR only from the second day of the time window: defined "n" the number of days composing the time horizon, in the output dataset there is "n-1" ATRs.

SMA

SMA stands for Simple Moving Average, which is nothing but the average of closing prices of shares in a given period. Unlike EMA, the average is not weighted, so the recent and less recent data have equal weight. Traditionally, this type of indicator is often used to estimate future returns.

EMA

The Exponential Moving Average (EMA) is a moving average that is widely used in technical analysis of financial markets. This trend-following indicator gives more weight to the more recent prices of a security and less weight to the older prices. In this way, EMA, responds more quickly to recent price changes than the simple moving average. In technical analysis, these two indicators are commonly used together, because SMA defines the long-term trend, while EMA informs about a deviation from this trend. The big advantage of introducing EMA in the model is, indeed, to consider the most recent developments in market prices, and make the model more responsive to them.

To calculate the EMA, it is necessary to define the time horizon on which it is calculated. The standard choice in literature is a time window of 12 periods, where every period is a trading day. Also in our model, a 12 period EMA has been chosen. The EMA is calculated by multiplying the current period's closing price by a smoothing constant and then subtracting the previous period EMA value. The smoothing constant is determined based on the, co-called, smoothing factor, that is a constant that determines how sensitive the EMA is on most recent prices. The larger the smoothing factor is, the more sensitive the EMA is. The default value in the TA-LIB function for the smoothing factor is 2, which is the value most used in technical analysis, and it has not been modified for this thesis.

The exponential moving average is calculated as:

$$EMA_t = close_t * \left(\frac{s}{1+n}\right) + EMA_{t-1} * \left(1 - \left(\frac{s}{1+n}\right)\right)$$

Where:

- $close_t$ is the closing price on the day considered
- s is the smoothing factor.
- n is the number of periods in the time horizon, in our model 12 periods.

ADX

ADX indicator is a trend indicator. International Statistical Institute defines trend as "a long-term movement in an ordered series, such as a time series, which may be regarded, along with the oscillation and random component, as generating the observed values". Therefore, a trend can be either increasing or decreasing, and it has its own strength. Trend strength is the tendency of the price over time to converge in the same direction as the trend, and it is measured by the ADX. It is important to remark that this indicator provides no information about the direction of the trend, but only on its magnitude.

Trend indicators are also a widely discussed topic in financial literature. The article by Peachavanish (2016) aims to demonstrate the effectiveness of trend in trading, by constructing a portfolio of Thailand's stocks based on trend and momentum data. The paper uses a combination of long-term and short-term trend indicators as signals to rebalance his portfolio of stocks. In the long run author's portfolio outperformed the SET, the reference index of Thailand stock market.

The calculation of ADX proceeds in several steps, but, again, it's first necessary to establish its time window. As the other indicators we choose to use the most frequent time window in literature, 14 periods (every period is a working day). The first step is to calculate the first ADX of the historical series, that refers to the 14th period, so it is ADX_{14}.

$$ADX_{14} = (\sum_{t=1}^{14} DX_t)/14$$

Where DX it's the "directional index", and it's calculated as:

$$DX_t = \left(\frac{|PDI_t - NDI_t|}{|PDI_t + NDI_t|}\right) + 100$$

PDI and NDI are the positive directional indexes, calculated as:

$$PDI_t = \left(\frac{smoothed\ PDM_t}{ATR_t}\right) * 100$$

$$NDI_t = \left(\frac{smoothed\ NDM_t}{ATR_t}\right) * 100$$

And *"smoothed PDM"* (positive direction movement) and *"smoothed NDM"* (negative direction movement) are:

$$Smoothed\ PDM_t = \sum_{t=1}^{14} PDM_t - \left(\frac{\sum_{t=1}^{14} PDM_t}{14}\right) + \ current\ PDM$$

$$Smoothed\ NDM = \sum_{t=1}^{14} NDM_t - \left(\frac{\sum_{t=1}^{14} NDM_t}{14}\right) + \ current\ NDM$$

Where the directional movement, both negative and positive, indicate the difference between the maximum (minimum) price value reached on the day considered and the maximum (minimum) value reached on the previous day. Finally, for the calculation of ADX after the fourteenth day, given t > 14.

$$ADX_t = \frac{(ADX_{t-1} * 13) + current\ DX}{14}$$

MACD

The Moving Average Convergence Divergence (MACD) is a popular technical indicator used in the technical analysis of financial markets. It is a trend-following momentum indicator that is derived by subtracting two different exponential moving averages, that are calculated on two different time windows. The standard choice in literature is to subtract the 26-period Exponential Moving Average (EMA) from the 12-period EMA, and we adopted this scheme. The resulting line is called the MACD line, and it shows how the short-term average follows (or deviates) the trend defined by the long-term average. Even if it measures the price trend as ADX, MACD and ADX differ are built in a different way, and, since they can lead to different results, we decided to include both the indicators in our model.
The calculation of MACD is straightforward:

$$MACD_t = EMA_t(12) - EMA_t(26)$$

RSI

The relative strength index (RSI) is a momentum indicator developed by Wilder (1978). Momentum is a measure of the rate at which price changes over time: the greater the momentum, the greater the price changes are in a unit of time. The RSI identifies moments when the price of an asset changes more quickly than average and it is therefore an indicator of market interest in the asset, both long and short.

The most common RSI version can be calculated form the 14th period of the time series. From the 14th period, RSI is calculated as:

$$RSI_t = 100 - \left[\frac{100}{1 + \frac{(previous\ avg\ gain * 13) + current\ gain}{(previous\ avg\ loss * 13) + current\ loss}}\right]$$

Where *current gain* is the positive percentage price change in the day *t,* and *current loss* is the negative percentage price change in the day *t*, and t > 14. There can therefore be two possible situations:

- Current gain > 0, current loss = 0
- Current loss < 0, current gain = 0

2.2 Developing the ARMA Model

This section focuses on the ARMA model in general, and how it has been integrated in the portfolio optimization. The section develops as follow: a brief introduction to ARMA model is provided in first sub-section, sub-section 2 is dedicated to the choice between univariate and multivariate ARMA model, sub-section 3 focuses on autoregressive and moving-average models that composes the general ARMA model, and the final sub-section 4 explains how to fit the model to data.

An ARMA (Autoregressive Moving Average) model is a type of time series model that combines two components, an autoregressive component (AR) and a moving average component (MA), to describe the behavior of a time series, and it was developed by Moran & Whittle (1951). The AR component of the model captures the relationship between the current observation and the previous observations of the time series. Specifically, the AR component models the current value of the time series as a linear combination of its past values, with the weights or coefficients of each past value determined by a lag parameter called the order of the AR model (p). The MA component of the model captures the influence of past errors or residuals on the current value of the time series. Specifically, the MA component models the current value of the time series as a linear combination of its past errors, with the weights or coefficients of each error determined by a lag parameter called the order of the MA model (q). Together, the AR and MA components allow the ARMA model to capture both the short-term and long-term dependencies in a time series, and can be written as ARMA(p,q).

The autoregressive component is a model in which the dependent variable is a function of past values of itself. An autoregressive model of order p can therefore be written as:

$$y_t = c + \phi 1 y_t - 1 + \phi 2 y_t - 2 + \cdots + \phi p y_t - p + \varepsilon_t,$$

where ε_t is the white noise.

The order of the model expresses the number of lags between the dependent variable and the independent variable furthest away in time; this model is in the form of a stochastic difference equation.

In the moving average model, the dependent variable is a function of the mean of the series, the current error term, and the past error terms, where error terms are assumed to be normally distributed and independent. The generic MA (q) model is defined by the order of the model, q, the number of past error terms that influence the present value. The MA (q) generic model is as follows:

$$y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

In this thesis the model has been applied to stock indicators. The input of each of the ARMA models estimated is, indeed, the time series of every stock indicator, for every stock. Then, the ARMA model has been fitted to data, and it was used to make daily forecasts of stock indicators values.

The application of the ARMA model to financial market dynamics and stock indicators is explored by Gilli et al. (2019). The authors justify this choice through the components of the model itself. The autoregressive model is used because there is a relationship between the value of indicators and the error terms that has previously taken on. The underlying idea is that anomalous indicator values are a symptom of a particular market moments that attract bull or bear investors. The resulting trades then affect price and volume of trades, and future indicator values. Moving average model, on the other hand, captures the effect of new price-sensitive information release. Price-sensitive information are publicly available, and they affect price fluctuations in one direction (example are quarterly results of a company, or bad credit ratings). If that information is not immediately discounted by the price, but its effect is spread over time, each new signal (information) has a future effect on stock indicators, that is captured by moving average model.

Since ARMA model only applies to stationary series, the first step is to check the stationarity of the time series, for every indicator of every stock. A time series is said to be stationary if its statistical properties do not change over time. Stationarity is an important concept in time series analysis because many of the techniques used in modeling and forecasting assume stationarity or can only be applied to stationary time series. Specifically, a time series Y(t) is stationary if it satisfies the following conditions:

- Constant mean: The mean of Y(t) is constant and does not depend on time t.
- Constant variance: The variance of Y(t) is constant and does not depend on time t.
- Constant autocovariance: The autocovariance function of Y(t) is constant for all values of t and depends only on the time lag h between two observations, not on the specific time t.

If a time series is non-stationary, it can be transformed into a stationary time series through techniques such as differencing, seasonal differencing, or logarithmic transformations. These transformations can remove trends, seasonality, or other patterns that may cause non-stationarity in the data. It is important to note that while some time series may exhibit stationary behavior over a certain period or interval, they may not be stationary over their whole-time horizon. To apply ARMA model to the entire series, its stationarity must be checked over the complete time horizon. To test the stationarity of the historical series, the Augmented-Dickey Fuller test was performed, with a 1%, 5% and 10% confidence interval. The Augmented Dickey-Fuller (ADF) test is a statistical test used to determine whether a time series is stationary or not. The ADF tests the null hypothesis that a unit root is present in the time series, which implies that the series is non-stationary.

The ADF test is a regression-based test that estimates the following equation:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{P-1} \Delta y_{t-p+1} + \varepsilon_t$$

Where: Y(t) is the time series, ΔY(t) is the first difference of the time series, α and β are constants, γ is the coefficient of the lagged value of Y(t-1), δ_1, δ_2, ..., δ_p are the coefficients of the lagged differences of Y(t-1), and ε_t is the error term. The null hypothesis of the ADF test is that β = 0, which indicates the presence of a unit root, and the alternative hypothesis is that β < 0, which indicates stationarity. The ADF test then calculates a test statistic based on the t-value of the estimated β coefficient and compares it to critical values from the Dickey-Fuller distribution. If the calculated test statistic is less than the critical value, the null hypothesis is

rejected, and the time series is considered stationary. If the calculated test statistic is greater than the critical value, the null hypothesis cannot be rejected, and the time series is considered non-stationary. The test statistic is:

$$\mathrm{DF}_\tau = \frac{\hat{\gamma}}{\mathrm{SE}(\hat{\gamma})}$$

In Python, the Augmented-Dickey Fuller can be performed through the "adfuller" function in the statsmodel library. The function, given the historical series, returns the value of the test statistic and the p-value with a confidence level of 1%, 5% and 10%.

The model described so far, is the simplest version of ARMA model, the univariate ARMA model, in which only a time series is considered. It is possible to extend the analysis to a multivariate ARMA model, where more time series are simultaneously included in the model. A multivariate ARMA model is used to analyze and forecast the behavior of multiple related time series simultaneously. In a multivariate ARMA model, each time series is modeled as a linear combination of its past values and the past values of the other time series in the model. Like the univariate version, the model includes both autoregressive (AR) and moving average (MA) terms, where the AR terms capture the dependence of each time series on its own past values as well as the past values of the other time series, and the MA terms capture the influence of past errors or residuals on each of the time series. Multivariate ARMA models are useful in situations where there are multiple time series that are related to each other and modeling them simultaneously with a multivariate ARMA model can capture the dependencies between them and improve the accuracy of the forecasts. However, multivariate ARMA models can be more difficult to estimate and interpret than univariate ARMA models due to the increased complexity and the presence of cross-dependencies between the different time series. In addition, the number of parameters in the model can increase rapidly with the number of time series, making it more prone to overfitting if not properly calibrated. In this thesis we've chosen to develop different univariate ARMA model (one for every indicator), instead of a unique multivariate model for the following reasons.

- Increased complexity: Multivariate ARMA models involve modeling multiple time series simultaneously, which increases the complexity of the model. As a result, estimating the model parameters can be less precise.
- High dimensionality: The number of parameters in a multivariate ARMA model increases rapidly with the number of time series included in the model. This high dimensionality can make estimation less precise.

- Cross-dependencies: In a multivariate ARMA model, each time series is dependent on its own past values as well as the past values of the other time series. This can make it more difficult to isolate the influence of each time series on the others.
- Redundancy: the estimations of the univariate ARMA models developed in this phase will be then used in a SVM model. The SVM model will consider the interdependency between the input data, so it's assumed that if there is a correlation between stock indicators, it will be captured with the SVM model.

For the reasons above, we decided to use a univariate approach, and we calculate a single ARMA model for every indicator of every stock.

We then proceed to establish the order of the autoregressive and moving average models. To do this, it is first necessary to introduce the autocorrelation (ACF) and partial autocorrelation (PACF) functions.

Autocorrelation measures the correlation between a time series and a lagged version of itself. The lag is the number of time periods by which the series is shifted. A positive autocorrelation means that a high value in the current period is likely to be followed by a high value in the next period. Conversely, a negative autocorrelation means that a high value in the current period is likely to be followed by a low value in the next period. The autocorrelation function measures the strength and significance of the autocorrelation between a time series and its lagged values. The ACF is a plot of autocorrelation coefficients against the lag, where the autocorrelation coefficient is a measure of the strength of the linear relationship between the time series and its lagged values. The autocorrelation coefficient ranges from -1 to 1, where -1 indicates a perfect negative correlation, 0 indicates no correlation, and 1 indicates a perfect positive correlation. The formula for the autocorrelation function (ACF) of a time series is as follows:

$$ACF(h) = corr(Y_t, Y_{t-h})$$

Where h is the lag, Y_t is the value of the time series at time t, and Y_{t-h} is the value of the time series at time t-h. "Corr" is the correlation function, which measures the linear relationship between two variables.

The order of an AR model refers to the number of lagged values used in the model, in general, an AR(p) model uses p lagged values of the time series to predict the current value. The ACF can be used to identify the order of the AR model by examining the pattern of the

autocorrelation coefficients at different lags. Specifically, for an AR(p) model, the ACF will have significant autocorrelation coefficients at lags 1 to p, and the autocorrelation coefficients will decay to zero for lags greater than p. Therefore, by examining the ACF plot of a time series, we can identify the order of the AR model by looking for the point at which the autocorrelation coefficients become statistically insignificant. This is known as the "cut-off" point, and it indicates the maximum lag that should be included in the AR model.

The partial autocorrelation function (PACF) is a statistical tool that measures the linear relationship between a time series and its lagged values, while controlling for the effects of the intermediate lags. In other words, the PACF measures the correlation between a time series and its lagged values, after removing the effect of the other lags. The partial autocorrelation function of a stationary time series can be calculated by using the Durbin-Levinson Algorithm:

$$\Phi_{n,n} = (\rho(n) - \sum_{k=1}^{n-1} \Phi_{n-1,k}\rho(n-k))/(1 - \sum_{k=1}^{n-1} \Phi_{n-1,k}\rho(k))$$

Where $\Phi_{n,k} = \Phi_{n-1,k} - \Phi_{n,n}\Phi_{n-1,n-k}$ for 1≤k≤n-1 and $\rho(n)$ is the autocorrelation function.

The partial autocorrelation function (PACF) is closely related to the moving average (MA) model. Specifically, the PACF plot can be used to identify the order of an MA model, which is the number of past error terms that are included in the model. For a MA model, the correlation between a time series and its lagged values is expected to be high for the lagged values that are included in the model, and close to zero for all other lagged values. This is because the MA model assumes that the time series is a linear combination of its past error terms, with no direct dependence on its past values. Therefore, the PACF plot for an MA model is expected to have significant partial autocorrelation coefficients only for the lags that are included in the model, and close to zero for all other lags. The order of the MA model can be identified by looking at the number of significant partial autocorrelation coefficients in the PACF plot.

The estimation of an ARMA model involves determining the parameters of the model that best fit the data. There are several methods used to estimate multivariate ARMA models, including maximum likelihood estimation, least squares estimation, and Bayesian methods.

- Maximum Likelihood Estimation (MLE) is a commonly used method for estimating ARMA models. In this method, the model parameters are chosen to maximize the likelihood of observing the data given the model. The likelihood function measures the probability of observing the data given the model parameters. The parameters that

maximize the likelihood function are estimated using optimization techniques, such as gradient descent or Newton's method. This kind of estimation assumes that time series is stationary, and that error terms are normally distributed.

- Least Squares Estimation: Another method for estimating ARMA models is least squares estimation. This involves minimizing the sum of the squared differences between the observed data and the predicted values of the model. The model parameters that minimize the sum of the squared differences are estimated using optimization techniques. To use this approach the time series must be stationary.
- Bayesian Methods: Bayesian methods involve specifying a prior distribution for the model parameters and updating this distribution based on the observed data. This results in a posterior distribution for the model parameters, which can be used to estimate the parameters and make forecasts. Bayesian methods can be particularly useful for estimating multivariate ARMA models with complex structures, as they allow for the incorporation of prior knowledge and uncertainty in the estimation process.

Since we found out that the indicators time series are stationary, we choose to estimate ARMA model with the Least Square Estimation. The objective function for LSE is defined as:

$$SSE = \sum(y_t - \hat{y}_t)^2$$

where y_t is the observed value at time t, $\hat{y}_t$ is the predicted value at time t, and SSE is the sum of squared errors. One disadvantage of the LSE method is that it is sensitive to outliers in the data. Outliers can have a large impact on the estimates of the model parameters, leading to biased estimates.

2.3 Developing the Machine Learning Model

In this section, the predictions obtained from the ARMA model are used for the actual prediction of the returns of the considered stocks. To do this, a machine learning technique called SVM, or support vector machine, is used. The section is divided as follow; the first sub-section contains an introduction to machine leaning, and to SVM, the technique for our model. Then, in the second sub-section, the kernel of support vector machine will be chosen, and the model will be developed.

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and models that allow computers to learn from data without being explicitly programmed. It is based on the idea that machines can learn patterns and relationships in data, and then use that knowledge to make predictions or decisions. Machine learning has many real-world applications, such as image recognition, speech recognition, natural language processing, and recommender systems. With the increase in available data and computing power, machine learning has become a rapidly growing field with many developments and breakthroughs. This type of approach is vital for the proposed method, mainly for two reasons:

- Firstly, it is not necessary to establish a unique algorithm, for every indicator and every stock. It will be the machine learning process that decides, autonomously, and in a personalised way for each stock in the portfolio, how to match the different values of the predicted indicators to different levels of return.
- Secondly, machine learning techniques, compared to traditional programming techniques, allow the model to evolve over time, learning and modifying it. This fluid structure allows investors not to reprogram their algorithm after the evolution or changes in the market.

The machine learning algorithm used in this thesis belongs to the support vector machine, or SVM, that has proven to be effective in solving a wide range of classification and regression problems. The first literature reference on Support Vector Machines (SVMs) was published by Vapnik & Chervonenkis (1971), in the Soviet Union. While the paper did not explicitly describe the SVM algorithm as we know it today, it laid the theoretical foundation for the development of SVMs and introduced the concept of maximum margin hyperplanes, which is a key component of the SVM algorithm. Vapnik and Chervonenkis continued to develop the SVM algorithm over the years, with their seminal work on SVMs for binary classification. Cortes & Vapnik (1995) introduced the SVM algorithm that we use today and showed its effectiveness in solving a wide range of classification problems.

SVMs are based on the idea of finding the best decision boundary that separates the data points of different classes. The algorithm achieves this by identifying the "support vectors," which are the data points closest to the decision boundary. SVMs then use these support vectors to calculate the optimal hyperplane that maximizes the margin between the different classes.

The choice of the support vector machine involves numerous advantages over other machine learning techniques:

- It is effective in high-dimensional spaces, i.e., when there are multiple dimensions. In our case, in fact, return predictions are not based on a single indicator, but on a plurality of these.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile in the goal: it can be used for classification, regression, or outliers' detection. In our case, it will be used for regression purposes, and therefore prediction.
- Versatile in the input data: it allows the use of different types of kernels, which best fit the input data.

The support vector machine model is divided into two phases: the training and the test phase. In the first phase, the input is a dataset, called a training set, composed of many samples. Every row of the dataset contains a sample, which has a vector x_i and a variable y. The vector x_i is further composed of the variables, defined as "features", that characterise the sample, while the variable y is the so-called "correct answer", and it is the target variable of the classification. The training dataset can therefore be described as (X, y_j), where X is the matrix composed of all the vectors x_i, and the vector y_j is composed of all the "correct answer" variables. The algorithm then identifies an i-dimensional hyperplane, where i is the number of features. The hyperplane, called "decision boundary", divides the data set (i.e., the vectors x_i) into two classes and the points closest to the hyperplane are called support vectors. These points play a fundamental role, as they determine the margins of the model. The margins are vectors that are orthogonal to the decision boundary and equidistant from it. The margin vectors pass through the support vectors. Given a dataset, it will be possible to have more than one hyperplane that is able to separate the dataset into two classes, and the goal of the model is to find the best hyperplane, the most efficient one. To do this, margins should be considered. Intuitively, the greater the margin, the greater the space between the classified points and the decision boundary, and therefore the better the classification will be.

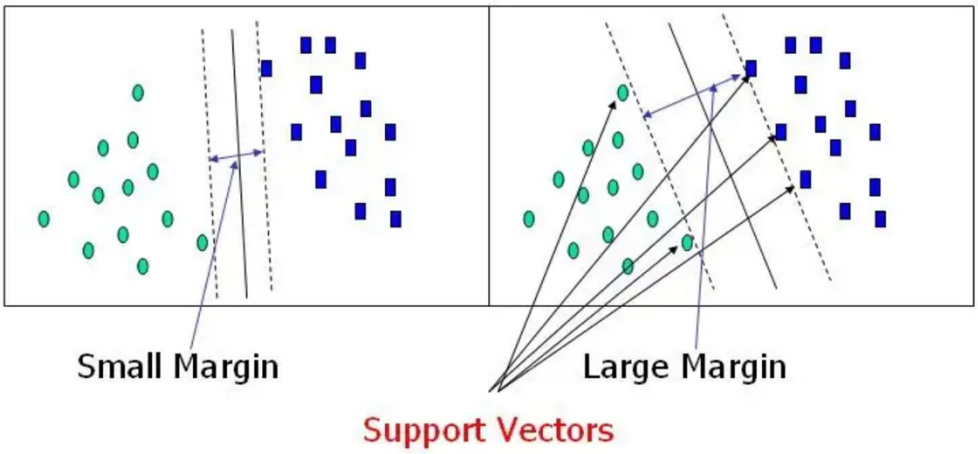

Fig 2.1: an example of two different 2-dimensional hyperplane

Considering the figure 2.1: the data set is the same, and the task of the support vector machine is to best classify the green points compared to the blue ones. In the example on the left, the margins are more restricted, and it is a sub-optimal solution compared to the example on the right. The largest margins possible are sought because this allows new points to be classified with greater precision, reducing the possibility of miss-classifications. The equation of the optimal hyperplane is thus obtained by maximising the margins. The support vector machine, in the simplest case of a linearly separable problem, therefore, solves the following optimization problem, identified, and solved for the first time by Cortes & Vapnik (1995):

$$Min\,\frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{n} ç_i$$

Subject to

$$y_i(wK(x_i) + b) \geq 1 - ç_i,$$
$$ç_i \geq 0, i + 1, \ldots, n$$

Where the objective variables are w, b.
In the above function:

- x_i is the feature vector

- y_i is the correct response vector
- K is a non-linear function.
- w is a vector of weight coefficients
- b is a constant
- n is the number of samples.

Special mention should be made for the constant C and the variable ç. It may happen that no hyperplane is able to correctly separate all the points in the dataset. To overcome this problem, it is allowed for some points to be misclassified, but in the optimization process, they are penalized. The coefficient C is in fact called the penalty coefficient and it is constant, while the variable ç is a measure of error. ç is actually the deviation, or the distance of the misclassified point from its correct boundary.
The problem is solved through a Lagrange minimization and the solution is given by:

$$w = \sum_i \alpha_i y_i X$$

Where alpha is the Lagrange multiplier and $0 \leq \alpha_i \leq Ç$. Considering that the hyperplane equation is w*X+b=0, once the optimal hyperplane is found, and therefore w and b, it is possible to classify every new point into the two regions of the space identified by the hyperplane. In the case of non-linear separable problems, as addressed in this thesis, it is necessary to use a kernel to have a solution. In the optimal solution the kernel is integrated in the following way:

$$w = \sum_i \alpha_i y_i K(x_i)$$

Where K is the kernel.

To explain why it was necessary to introduce the kernel, it's useful to define the concept of linear separable problems. In machine learning, a linear separable problem is a classification problem in which the classes can be separated by a linear boundary or hyperplane in the feature space. In other words, there exists a line, plane or higher dimensional hyperplane that can completely separate the two classes of data. On the other hand, in non-linear separable problems, the classes cannot be separated by a linear boundary or hyperplane in the feature space. This means that there is no straight line, plane, or hyperplane that can completely

separate the two classes of data. Non-linear separable problems are more complex than linear separable problems and require the use of a kernel. The fastest way to analyze the separability of the data is to compute the decision boundary of a linear classifier, such as the Perceptron or Support Vector Machine (SVM). If a linear classifier can separate the two classes of data points perfectly, then the problem is linearly separable. However, if the classification error is non-zero, then the problem is non-linearly separable.

A kernel is a function, or a series of functions that allow the machine learning algorithm to transform complex input data into a form that lead to a solution. There are different types of kernels, to adapt to different kind of data. Generally, a linear kernel is used in text categorisation problems because it performs well in the presence of a large amount of data but with reduced dimensions. On the other hand, a polynomial kernel is used in image processing processes and in the study of digital images, where there is a large number of dimensions but fewer data to compute. When there is the need of more generalist kernel, suitable for most datasets, or because the form of the dataset is unknown, the RBF kernel is the most suitable choice. This kernel function was also used in the model developed in this thesis.

The RBF kernel is a non-linear combination of the dataset features, and it defines K as:

$$K = e^{\frac{-||x_1 - x_2||^2}{2\sigma^2}}$$

Where σ_2 is the variance between x_1 and x_2.

2.4 Portfolio Optimization

This final section of the chapter is dedicated to the portfolio construction. There are many different ways to build an efficient portfolio, the most common are Markowitz optimization and Index model. Since the analysis conducted so far allowed us to estimate daily return for the stock, the most appropriate method is Markowitz optimization. The core of the dissertation is the comparison between the portfolio built with machine learning estimation and a portfolio built in traditional way. The section is therefore divided as follow. In sub-section one the general idea of the portfolio optimization is developed, and the benchmark portfolio requirements are presented. Sub-section two and three are dedicated to the estimation of

benchmark portfolio and machine learning portfolio. Finally, in sub-section four, transaction costs are considered.

Markowitz optimization technique seeks to maximize expected returns for a given level of risk. The main idea behind this kind of approach is to use the power of diversification to obtain the maximum expecting return, keeping the variance low. To estimate Markowitz portfolio is necessary first to estimate the expected return and the expected variance of the single stocks, and the covariance between every pair of stocks. The portfolio built in this thesis uses the Markowitz optimization technique, but estimation from machine learning is used for expected returns. To build the portfolio, we choose to fix variance parameter, and to maximize the expected return; in this way we could make an estimation of the average European investor risk aversion and set the expected variance of the portfolio consequently. We had to choose, then, how to build a benchmark portfolio, that can be compared with the machine learning portfolio. In order to be comparable, the benchmark portfolio must have specific characteristics:

- It must be derived with the same optimization technique, Markowitz optimization.
- The optimization process must use the same financial instrument (stocks and risk-free asset) of the machine learning portfolio.
- It must have the same expected risk (i.e., expected variance) of the machine learning portfolio.

Prior to choosing the variance, it is advisable to introduce the concept of risk-aversion. An investor is risk adverse when her utility function does not only depend on the expected value of the investment, but also on the risk, or the variance of the expected value. In other words, in any system that can be reconducted to the lottery paradigm, an individual is risk-averse if she prefers a sure amount of money equal to the expected value of the lottery, rather than playing the lottery. An individual is risk-neutral if she is indifferent between the sure amount of money and playing the lottery, while she is risk-lover if she prefers to play the lottery. Over time, different measures of risk aversion have been developed, however, in this thesis, we will use the most famous and widely used, the Arrow-Pratt coefficient. The main assumption behind Arrow-Pratt risk-aversion measure is that investor preferences can be described with the Von Neumann-Morgenstern utility function. Under this assumption an individual can be:

- Risk-averse, with a concave utility function;
- Risk-neutral, with a linear utility function;
- Risk-lover, with a convex utility function.

Defined U(w) as the utility function of a generic individual, the Arrow-Pratt coefficient is defined as:

$$A = -\frac{u''(w)}{u'(w)}$$

The Arrow-Pratt coefficient increases as the individual's risk aversion increases, allowing for comparison of different individuals based on their risk aversion. The coefficient is negative for risk-lover individuals. It is then possible to define the relative risk aversion coefficient, starting from the Arrow-Pratt coefficient. The relative risk aversion coefficient is widely used in finance and investment choice theory to design investment decision and portfolio construction to match individual preferences. Considering a generic investor, with wealth "w", who decide what percentage of w to allocate to risky assets, and what to invest in risk-free assets: the relative risk averse coefficient is then defined as the actual amount that the individual would allocate as investment in risky assets. The relative risk aversion coefficient R is calculated as:

$$R = -w[\frac{u''(w)}{u'(w)}]$$

Knowing the risk aversion of a single individual (or the average risk aversion of a population), it is then possible to determine what the optimal portfolio risk is for the individual or the population. Considering a portfolio p, and an individual's utility function V_p, under the assumption that the individual is risk-averse,

$$V_p(E(r_p - r), var(r_p))$$

where r is the risk-free rate, will depend positively on $E(r_p\text{-}r)$ and negatively on $var(r_p)$. The portfolio p will be optimal for the individual if, and only if, for each asset j that makes it up, the derivative of V_p with respect to x_j (the weight of the asset) is equal to zero.
Analytically, this efficiency condition becomes:

$$\frac{\partial V}{\partial \mathrm{E}(\cdot)}\frac{\partial \mathrm{E}(\cdot)}{\partial x_j} + \frac{\partial V}{\partial var(\cdot)}\frac{\partial var(\cdot)}{\partial x_j} = 0$$

$$= \frac{\partial V}{\partial \mathrm{E}(\cdot)}E(r_j - r) + \frac{\partial V}{\partial var(\cdot)}2cov(r_j, r_p)$$

$$E(r_j - r) - \lambda_p cov(r_j, r_p) = 0$$

Where, $\lambda_p = -2\frac{\partial V/\partial var(\cdot)}{\partial V/\partial E(\cdot)}$

Sercu (2009) demonstrates that λ_p is equal to Arrow-Pratt's relative risk aversion R.

$$\lambda = \frac{E(r_p - r)}{var(r_p)}$$

In this way, by knowing the average investor's relative risk aversion, it is possible to obtain the corresponding variance, which outlines the optimal portfolio for the average investor. To know what the average relative risk aversion is, one must investigate the literature. The most recent meta-analysis on the subject was conducted by Elminejad et al. (2022). The authors conclude that the average worldwide risk aversion coefficient is between 2 and 7. By narrowing the field to a European investor, the average value is 2.5.

To build benchmark portfolio, traditional Markowitz optimization has been implemented, where some risky asset and a risk-free asset are combined to achieve the highest possible utility for an investor.

Traditionally, this model is developed in three phases:

- In the first phase, all possible risk-return combinations for the risky assets are identified.
- In the second phase, the optimal risky asset portfolio is identified.
- In the third and final phase, the risk-free asset is integrated with the optimal risky portfolio to obtain the optimal portfolio: a combination of risky assets and risk-free asset.

The first step aims to find the minimum variance frontier. This function defines, for every level of expected return, the risky portfolio with the lowest expected variance, and therefore expresses the best combination of risky assets for each portfolio. The second phase deals with finding the best portfolio within the efficient frontier. To do this, the capital asset line (CAL) is traced. The CAL starts from the point (0, r_f) and has a slope equal to the Sharpe ratio of the optimal risky portfolio. The tangency point between the CAL and the minimum variance frontier is where the optimal risky portfolio lies. Investors can adjust the variance and expected value of their portfolio by modifying the proportions of risky assets and risk-free assets, moving along the CAL. It is important to note that, if investors move to the right along the CAL and exceeds the point identified by the optimal risky portfolio, they are short selling the risk-free asset, to have a proportion of risky assets in their portfolio greater than 100%. This strategy increases the variance of their portfolio and increases the expected return. However, among the

assumptions of this dissertation, we deny the possibility of short-selling and therefore limit investor's possibilities. If an investor wants to reach a greater expected value, she must choose a different combination of risky assets and move along the minimum variance frontier.
This type of approach leads to what is called separation propriety. The separation propriety states that the portfolio choice solution is reduced to two independent tasks:

- In the first task, the optimal risky asset portfolio is chosen. This portfolio is the same for every investor, and it maximizes the Sharpe ratio. This task is purely technical and does not consider either the investor's risk aversion or the portfolio manager's preferences.
- In the second independent task, the combination of risk-free and risky assets is tuned to find a portfolio that best matches the investor's risk preferences.

In this way, hedge fund mangers offer customized solutions to their clients simply by changing the proportions of risk-free assets in the portfolio, without modifying the combinations of risky assets, making the fund management process less expensive.

The selection of the minimum variance frontier is then carried out. To do this, it is first necessary to estimate the expected return of the individual stocks. The simple arithmetic average of the historical series of stock daily return is used as an estimator. The variance-covariance matrix is then calculated. This is a square matrix that contains the variances and covariances associated with all the stock expected returns. The diagonal elements of the matrix contain the variances of the expected returns and the off-diagonal elements contain the covariances between all possible pairs of stocks. The square matrix has order *i*, where *i* is number of stocks in the portfolio. Therefore, any portfolio composed of the stocks in the portfolio will have an expected return and variance.

$$E(r_r) = \sum_{i=i}^{n} w_i E(r_i)$$

$$\sigma_r^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j Cov(r_j, r_i)$$

Where:

- i=j is the number of stocks in the portfolio

- w is the relative weight (expressed as a percentage) that each action occupies within the portfolio
- r is the return

Regarding the risk-free asset, Italian government bond has been considered.

For each level of return, the composition of portfolios that lies on the minimum variance frontier is obtained by maximizing the Sharpe ratio, defined as:

$$Sharpe\ Ratio\ = \frac{(r_r\ -\ r_f)}{\sigma_r}$$

Still considering the ban on short selling, it is necessary to perform a constraint minimization, where each weight "*w*" must necessarily be greater than or equal to zero. Finally, to find the optimal combination of risky assets, the capital asset line is drawn, which, starting from the point (0, r_f), touches the minimum variance frontier.

Considering the condition:

$$\lambda = \frac{E(r_p - r)}{var(r_p)}$$

Where r_p is the expected return of the final portfolio P with risky and risk-free asset, and var(r_p) is the variance of r_p

Together with the conditions:

$$r_p = \alpha * r_r + \beta * r_f$$

$$Var(r_p) = \alpha * r_r$$

Where:

- r_r is the expected return of the risky portfolio
- r_f is the risk-free rate
- Alpha and beta are the percentage of the risky portfolio and the risk-free asset in the final optimal portfolio

lead to the final composition of the investor portfolio P

For the construction of the Black-Litterman portfolio, we adopted a similar process.

The great advantage of estimating returns through machine learning, rather than using an arithmetic mean, is that daily estimates of returns are obtained, and it is possible to rebalance the portfolio during the year considering the evolution of the stock return forecast. We decided, considered the results of Kara et al. (2019), to rebalance the portfolio every twenty working days. Of course, frequent rebalance leads to better portfolio performance, and, ideally, a daily rebalance would be the optimal choice. But since we included in our model transaction cost, the trade-off between frequent rebalance advantages and connected transaction cost should be considered. This procedure leads, at the end of the year, with 12 five different portfolios. Each of the 12 portfolios has been estimated with the procedure adopted for Markowitz portfolio:

- First variance-covariance matrix and expected return for each stock in the 20 days period has been estimated
- Then, minimum variance fronter and capital asset line are used to find the best risky portfolio
- Finally, the risky portfolio and the risk-free asset are combined to create a portfolio that match the Markowitz portfolio variance.

With equal risk, the final comparison between the two approaches will therefore be decided by the portfolio that will be able to obtain a higher actual return. To obtain the actual return, the actual performance data of the stocks in 2019 was simply taken, whose returns were multiplied by the weights in the various portfolios.

Finally, there is one last point to consider when preparing this comparison, that is the cost of continuous rebalancing. A transaction cost value of 0.2% of each transaction is assumed. This implies that costs will be incurred to purchase the first machine learning portfolio, but in subsequent rebalances, only the differences in the stocks will be calculated. In this way, only the actual costs incurred are calculated: no costs are applied to maintaining the stocks in the portfolio, but only to those traded. The 0.2% cost is applied both to the sale and the purchase. The value of the fees has therefore been calculated on the rebalancing of the machine learning portfolio, that is, the fees have been calculated on the actual transactions. By incorporating transaction costs into the calculation of returns, the net return is obtained, which will determine the success or failure of the portfolio.

CHAPTER 3: APPLICATION OF THE MODEL TO THE DATA

This chapter focuses on data analysis and application of the model described in the chapter before to real financial data. The chapter structure is as follow: in the first section we will describe the dataset, as well as a quantitative description of the employed stock basket; then we will describe how our forecasting strategy works in section 2, focusing on a single stock for the sake of readability; section 3 contains the portfolio optimization part, where the optimal portfolio based on the forecast is built; finally, section 4 performs a comparison exercise with benchmark portfolio.

3.1 The Dataset

Our data employs data coming from the FTSE MIB –Italia Mid Cap index. This index was born in 1992 and it is the most important stock market index in Italy, tracking the performance of the 40 most liquid and capitalized stocks listed on the "Borsa Italiana" (the main Italian stock exchange). This index serves as a benchmark for the Italian stock market, providing investors with a snapshot of the performance of the country's leading companies across various industries. It is calculated using free float market capitalization.

The free float market capitalization is the total value of a company's outstanding shares that are available for trading on the open market. It is calculated by multiplying the current market price of each share by the number of shares available for trading. The concept of "free float" means that the only share that are used for the computations are the ones that are not held by the management, controlling entities, the Government, or by any other entity that might exercise a significant influence on the company. That is, the only shares used to determine the value of a firm are the ones whose purpose is to be exchanged, not to directly influence the handling of its business.

In 2019, the FTSE MIB index collected about 80% of total capitalization of the Italian Stock Exchange and 90% of the volume of trades. Its composition is rather varied, reflecting the Italian production structure. More in detail, we can divide the index's companies in the following way:

- 13 companies operate in the financial, insurance and linked activities sector;

- 11 industrial companies;
- 3 companies involved in energy-production and raw material sector;
- 2 telecommunication companies;
- 6 companies that are public services providers;
- 3 companies operating in the health sector;
- 1 in the fashion sector;
- 1 in the food sector.

Firms in raw materials and energy production present the highest weigh (in capitalization terms) in the index, (18.2%), followed by the ones in the financial industry (16.6%). The dataset spans the period 2000-2019. We have decided not to include the data after 2020 for the exceptional period that followed the Covid pandemics. Indeed, we could consider the 2020-2022 period as a "Black Swan". In the time series and forecasting literature, this issue is handled including elements of stochastic volatility, like Lenza & Primiceri (2022), and Carriero et al. (2022). Notice, however, that there is evidence that some machine learning techniques, in principle, could be able to improve forecasting during this period, see Goulet Coulombe et al. (2021). Nonetheless, this level of sophistication is outside of the reach of this work. Notice, however, that data for some companies are not available for the entirety of the period. This happens when the companies are quoted after the beginning of the new millennium, or they changed their financial structure, that makes older data useless. For what concerns our analysis, this issue is relevant for the following companies:

- CNH industrial and Ferrari, with the former being quoted on the stock exchange in 2011, while the latter started to be exchanged in 2015.
- The utility company Terna that was subject to an OPV (offerta pubblica di vendita, that is a public offer of subscription) in 2004. Moreover, in 2011 it changed his financial structure, becoming the holding of "Terna Rete Italiana" and "Terna Plus".
- Snam in 2016 did a spin-off of "Italgas Reti S.p.A.", following European legislations regarding market liberalization.

To have a general overview of the evolution of the index, figure 3.1 represents the evolution of the FTSE-MIB value of the last 20 years, while figure A.1 in the appendix shows in detail the composition of FTSE-MIB. Having a closer look to figure 3.1, we can see two big drops in index value:

- In the months following January 2001, caused by the dot com bubble crisis;
- The 2008 great financial crisis.

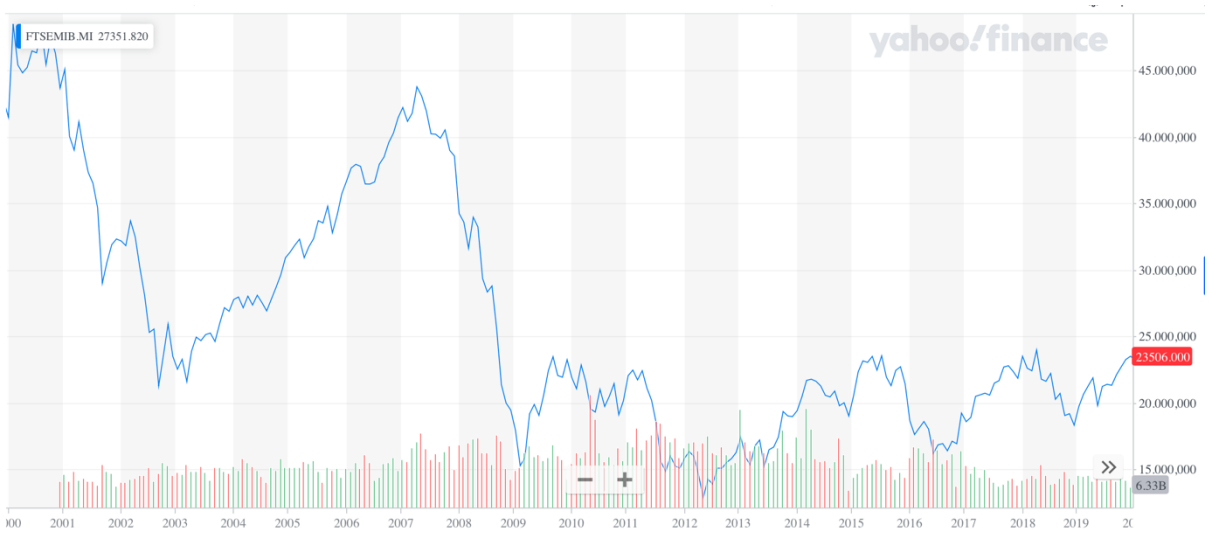

Figure 3.1 – FTSE-MIB value evolution from 2000 to December 2019. In the horizontal axis there is the time, in the vertical axis there is the value of the index. Source: Yahoo Finance.

The starting dataset contains the opening and closing prices, as well as the trading volume of all stocks belonging to the FTSE-MIB index in 2019. Each stock is collected as daily time series. In order to recover these series from "Yahoo Finance", we employed the "datareader" function, in Python. This function is thought exactly to retrieve financial data online. Through this function, the ticker of the stock of interest, the period of interest, and the online source are used as an input. In this way, we are able to recover a daily dataset that, for each stock of interest, contains:

- Opening and closing price;
- Traded volumes;
- Highest and lowest price of the day.

In order to handle this dataset, we use the TA-LIB library in Python. The first step in the data handling process is the creation of the training and test sets. This is done by slitting the sample in 2: data realized before the 1st of January 2019 enter the training set, the remaining observations compose the test set. This is a fundamental step to develop ARMA and SVM models. In the ML jargon the training set is the set used for the "estimation" of the model. The test set, instead, is used to check whether the model performs properly also with new data that were not included in the estimation step. To do so, an out-of-sample forecasting exercise is performed. The TA-LIB library is the tool used for the calculation of the indicators that will be used in the following sections The final, preliminary, output is therefore composed of two different datasets: one training set, with the historical value of indicators and that will be used

to forecast future indicators and stock prices, and one test set, that contains the realized returns in 2019.

3.2 Forecast method applied to a single stock

In this section we will present the model via the application to a single stock. The process here described will be then employed for the other 29 stocks. For this purpose, we selected UNIPOL GRUPPO SPA. Unipol Gruppo SPA (UNI.MI), quoted in Milan stock exchange in 1990, is a company in the financial and insurance sector, and it is the holding of Unipol-Sai Assicurazioni, one of the major insurance companies in Italy.

We retrieved the data starting from the 1st of January 2000 and computed the indicator values. Some key statistics for the time series are shown in table 3.1.

UNI.MI	***Close Price***	***Daily Return***	***ATR***	***EMA***	***ADXR***	***MACD***	***SMA***	***RSI***
AVERAGE	24,90	-0,02%	0,4948	24,87	26,96	-0,0706	24,87	49,27

Table 3.1 - average of the daily indicators for UNI.MI stock.

Before proceeding with the model-building process, let's have a look at the behavior of the UNIPOL's price over time. Figure 3.2 shows the evolution of the stock's price over the analyzed period. The series is clearly trending but, contrary to the classical stock market index series, the trend is negative over time. After a pretty stable price up to 2008 circa, we see clear decline in the aftermath of the Great Financial Crisis and a plateau after mid-2012. The daily returns are instead plotted in figure 3.3. The average daily return is essentially zero (table 3.1 reports a value of -0.02%) and, while the price series is clearly non-stationary, the returns series looks like close to a Gaussian white noise at a first look, albeit it could present slightly fat tails.

Figure 3.2 – Unipol stock price evolution from 2000 to 2019. In the horizontal axis there is the time, while in the vertical axis there are the corresponding closing price, expressed in Euros.

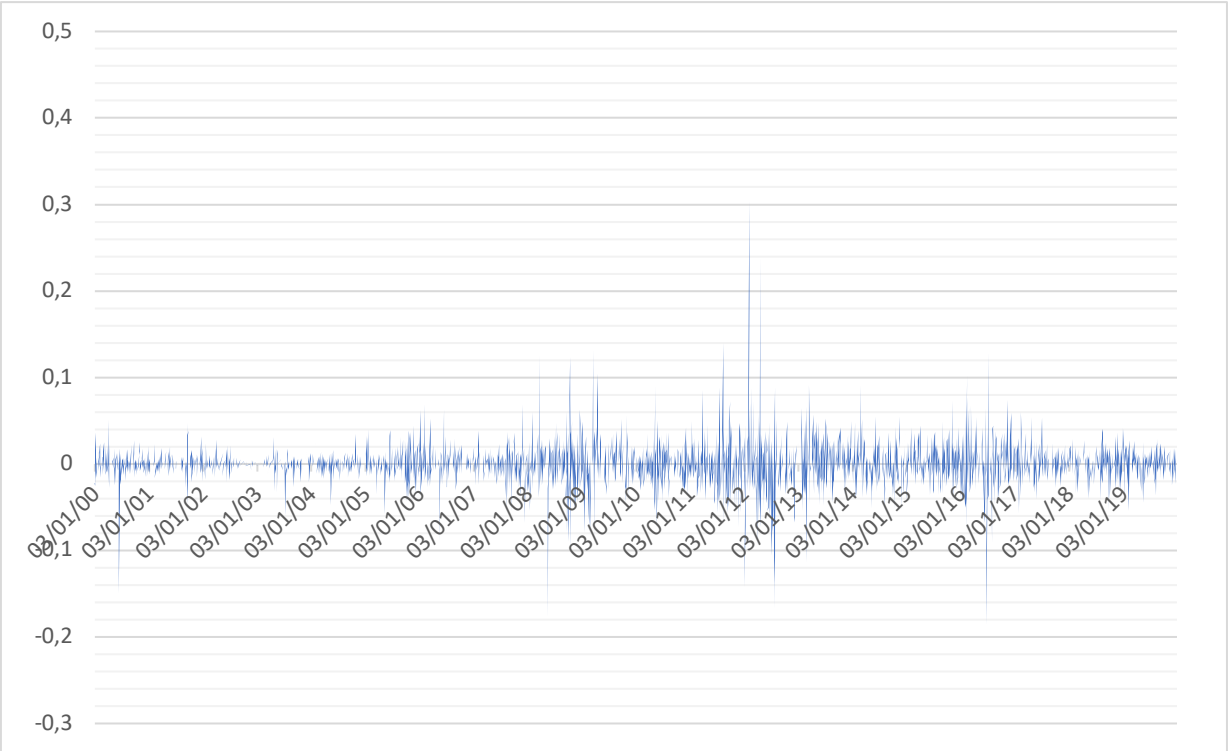

Figure 3.3 – Unipol stock daily return, from 2000 to 2019. In the horizontal axis there is the time, while in the vertical axis there are the corresponding daily returns

In order to forecast the price series, we at first need to forecast our indicators. To do so, we employ univariate autoregressive moving average (ARMA) models for each stock series. A key

requirement for the estimation of an ARMA model is that the underlying data generating process (DGP) is stationary. Nonetheless, as shown in the context of VARs by Sims et al. (1990), provided that a sufficiently large number of lags in the autoregressive part is inserted, to estimate in level a non-stationary process is not a major problem. In particular, differentiating a non-stationary process may imply misspecification of the true model if the series are cointegrated.

The augmented Dickey-Fuller (ADF) has been performed test for every indicator. Table 3.2 summarizes the results of the test for UNI.MI stock.

UNI.MI	***ATR***	***EMA***	***ADXR***	***MACD***	***SMA***	***RSI***
ADF – 1%	-3,43	-3,41	-3,51	-3,44	-3,39	-3,49
ADF – 5%	-2,86	-2,83	-2,90	-2,86	-2,84	-2,89
ADF – 10%	-2,57	-2,51	-2,62	-2,59	-2,52	-2,61

Table 3.2 – ADF test, for three different level of confidence of time series of UNI.MI indicators.

These results are then compared with the critical values of the ADF test for sample larger than 500, that are, respectively -2,567, -2,941, -1,616 for significance levels of 1%, 5% and 10%. Since the ADF test, for every indicator considered, is more negative than the critical values, the null hypothesis of a non-stationary series can be rejected. We can then then consider the time series as stationary and apply the ARMA model without data manipulation techniques.

We then proceed with the lag-selection procedure. There are many ways to select the number of lags in this context: one can choose the ones according to an information criterion (like the AIC or the BIC), checking the number of lags that makes the residuals a white noise, or looking to ACF and to the PACF. We choose the employ this latter technique. Notice that, in this case, we choose the number of autoregressive components looking at the PACF and the order of the MA part looking at the ACF. In picture from 3.4 to 3.9 (obtained with the “statsmodel” library in Python), we can see that while the ACFs are declining pretty smoothly, the PACFs presents some spikes at the begins and some then very small values. These indicia point toward to Ars process. More in detail for the ATR, the EMA, for the SMA and for the RSI we spot an AR(1), for the others an AR(2).

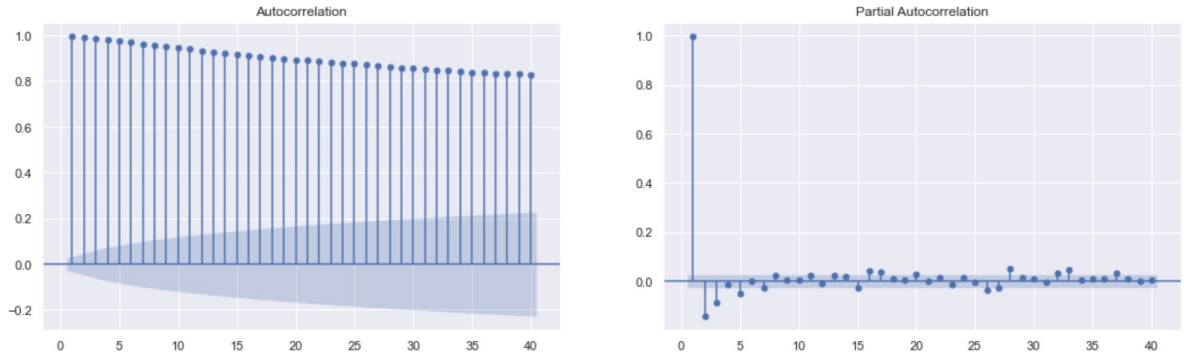

Figure 3.4 – Autocorrelation function and partial autocorrelation function of ATR time series for UNI.MI stock, as a function of number of lags. Light blue area represents statistically significance area at 95%

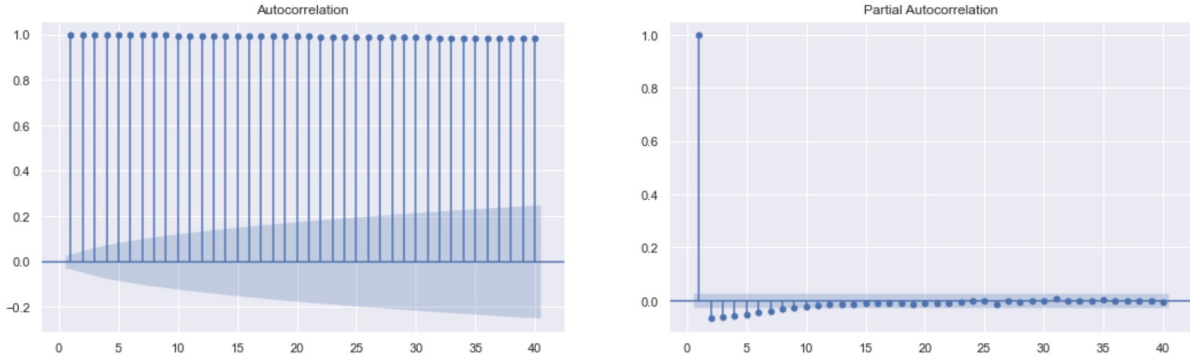

Figure 3.5 – Autocorrelation function and partial autocorrelation function of EMA time series for UNI.MI stock, as a function of number of lags. Light blue area represents statistically significance area at 95%

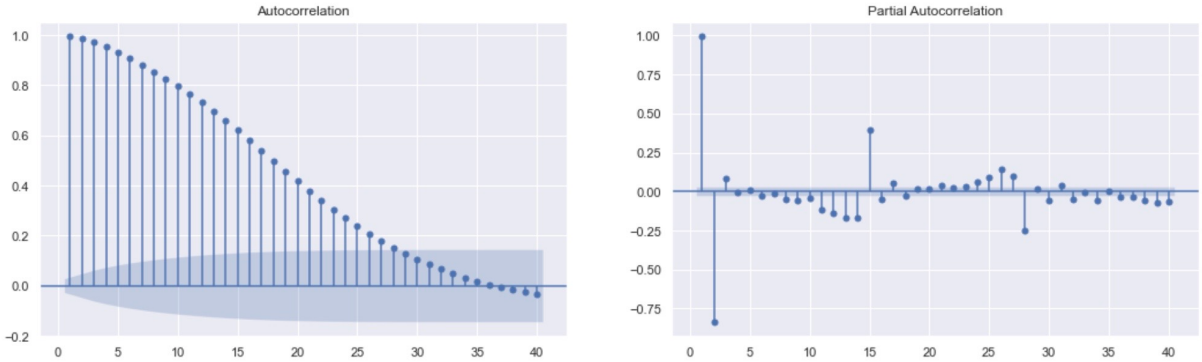

Figure 3.6 – Autocorrelation function and partial autocorrelation function of ADXR time series for UNI.MI stock, as a function of number of lags. Light blue area represents statistically significance area at 95%

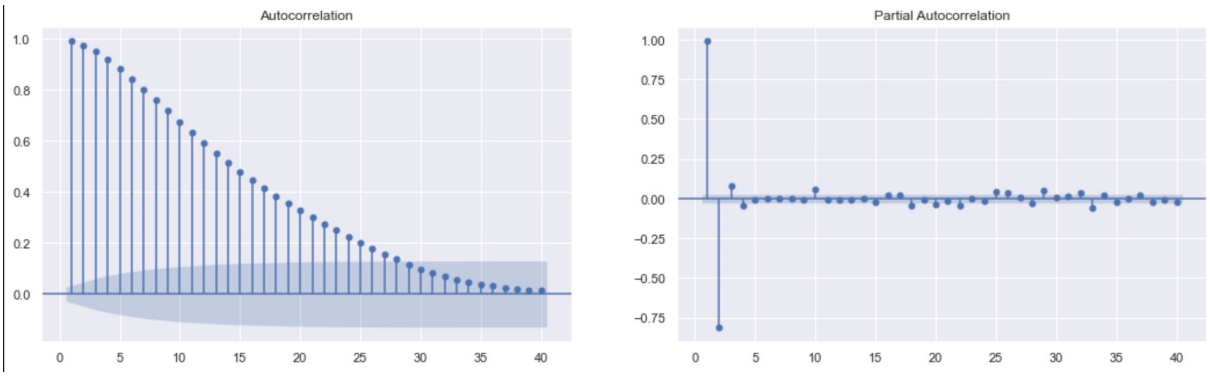

Figure 3.7 – Autocorrelation function and partial autocorrelation function of MACD time series for UNI.MI stock, as a function of number of lags. Light blue area represents statistically significance area at 95%

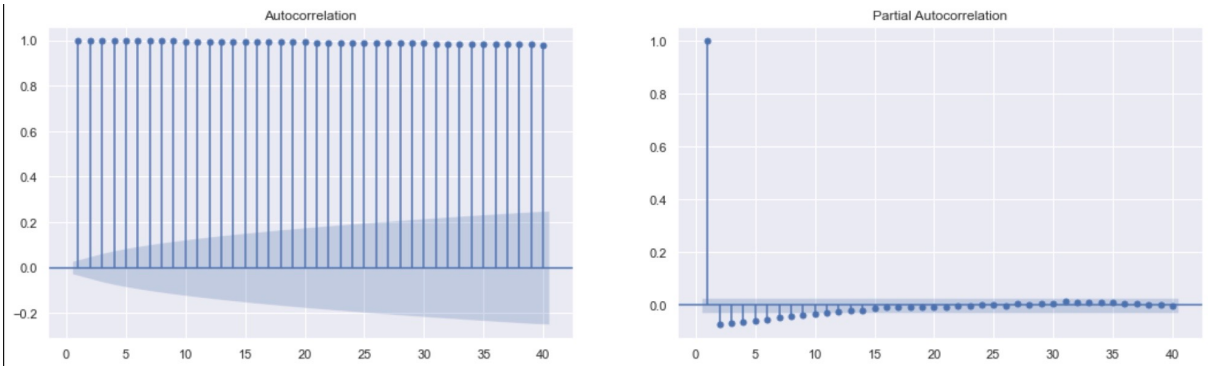

Figure 3.8 – Autocorrelation function and partial autocorrelation function of SMA time series for UNI.MI stock, as a function of number of lags. Light blue area represents statistically significance area at 95%

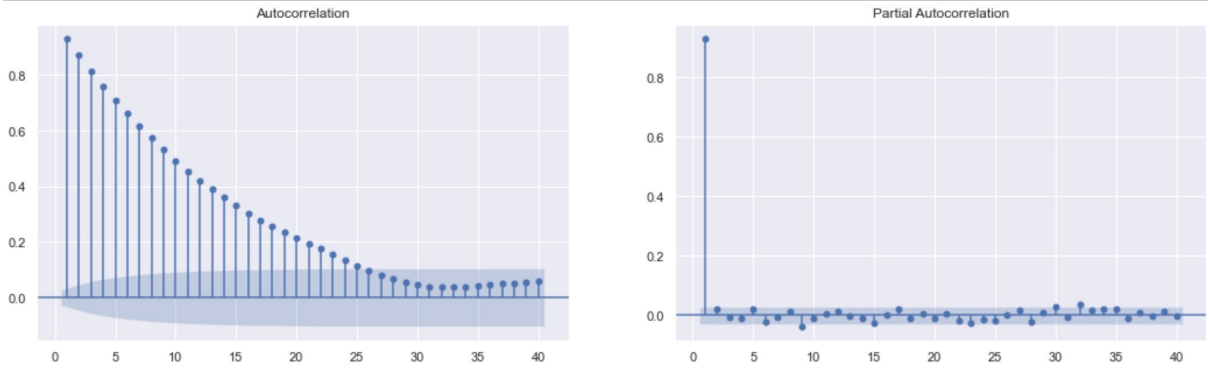

Figure 3.9 – Autocorrelation function and partial autocorrelation function of RSI time series for UNI.MI stock, as a function of number of lags. Light blue area represents statistically significance area at 95%

We now have a closer look at these PACFs and ACFs. EMA and SMA present a significant positive autocorrelation. This behavior can be explained by how EMA and SMA are built: since they are averages of previous periods, their value depends on the value they had in the past.

Moreover, ATR indicator has a strong positive (but slightly decreasing) autocorrelation, RSI, MACD and ADXR autocorrelations functions have a similar behavior, since they are strongly decreasing, till they become statistically non-significant at, respectively, 26th, 29th and 29th lag.

The partial autocorrelation functions, instead, decrease more rapidly, and, for every time series considered become non statistically significant within the 10^{th} lag. The only exception is the PACF of the ADXR, whose plot shows some peak later on. We did this analysis for every stock in the basket, obtaining similar results.

We proceed to estimate an ARMA (10,10) for each series. This allows us to take into account in a simple, effective and versatile the information described in the paragraphs above. An excessive number of lags might seem inappropriate in case the time series are not very long. In our case, we exploit the very high frequency of the series that allows for a rather high number of observations. In this way, we exploit the asymptotic properties of MLEs. Indeed, if the number of observations is big enough, these estimators are consistent.

The ARMA model, starting from the training set returns as output the forecasted daily value of the indicators for the test set. To sum up the results of the model, we included in table 3.3 the average predicted value for every indicator, the average of the realized value as well as the percentage difference between the averages of predicted and realized values:

UNI.MI	*ATR*	*EMA*	*ADXR*	*MACD*	*SMA*	*RSI*
predicted indicators	0,077	3,675	14,833	0,0002	2,905	47,950
realized indicators	0,094	4,565	23,269	0,0434	4,566	57,776
Percentage difference	21,70%	24,20%	56,87%	16114,4%	57,17%	20,49%

Table 3.3 averages of predicted and realized values of the indicators for UNI.MI stock. In the last row there is the percentage difference between the two averages.

In this case, the ARMA model performed better with RSI, ATR, and EMA. The model performed very poorly for the estimation of MACD, with a huge percentage difference between the forecasted values and the realized ones. To show how the ARMA model performance are affected by the length of the test set, i.e., how the model loose precision with the increase of the number of periods to estimate, we plot, in the figure 3.10, the absolute value of the percentage difference between daily estimates and realized values, for every indicator.

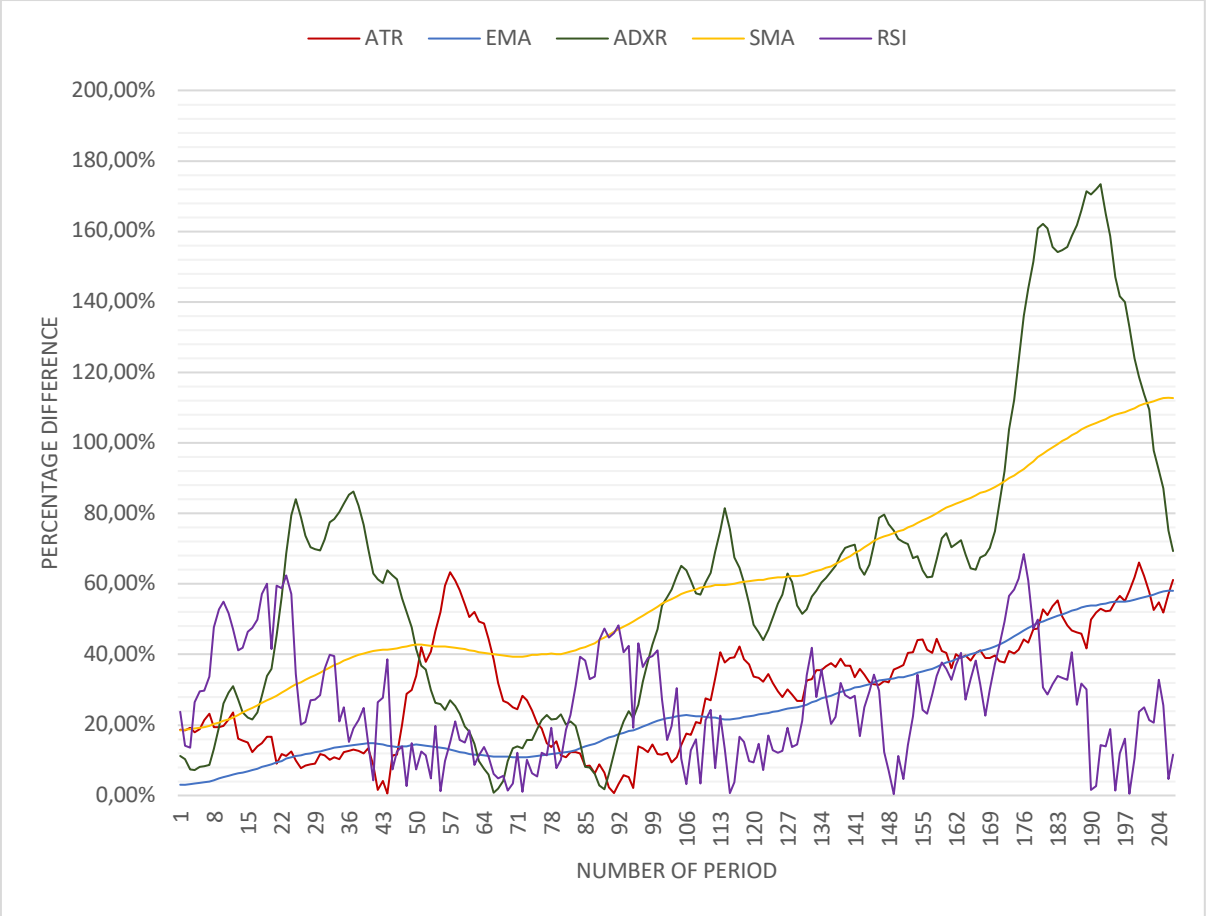

Figure 3.10 – absolute value of the percentage difference between predicted and realized value, for UCG.MI indicators, as a function of number of periods of forecasting.

We excluded from the plot the MACD series, given its anomalous values. The result shows a visible loss of precision for the two averages in the graph (SMA and EMA), with a linear dependence with respect to number of periods of forecast. Also, ADXR and ATR, show a positive correlation between, number of periods of forecast and loss of precision, but the standard deviation of the percentage difference function is higher than the SMA and EMA.

The dataset now contains a training set, composed of price, volume, and the realized indicator values, with temporal limit January 2000 – December 2018, and a test set, also composed by daily price and trading volume, but it also contains the value of indicators predicted by the ARMA model. The latter set has temporal limits January 2019 - December 2019. The two sets are taken as input for the support vector machine model, that aims to forecast 2019 stock prices.

The SVM model has been implemented with the "sklearn" library in Python, using the RBF kernel. In the training phase the model used the value of the indicators in the training set

as features vectors, and the value of the close price (also in the training set) as the correct-answer vector. In the test phase, the features vectors are the values predicted by ARMA model, while the output is a vector composed by the forecasted daily stock price. In the output, also a vector with the realized daily closing prices has been added, to allow comparison and performance valuations. Preliminary to the model fit, "standard scaler" function has been applied to the whole dataset. The use of this function is essential when the data is non-homogeneous in term of orders of magnitudes. Since the starting dataset contains indicators that provide different information, calculated in different ways and with different orders of magnitude, standard scaler function was necessary. This function normalizes the value of the data and uniforms their orders of magnitude, so different variables can be compared. Without the use of this function, more weight would have been assigned to indicators that are greater in absolute value. After the model fit, the reverse standard scaler function has been applied to the output, to make forecasted stock price, that were previously normalized with indicators, comparable with realized stock prices.

The average predicted stock price is 3,68 Euros, while the average realized stock price was 4,29 Euros. We plot the difference between the realized and the forecasted stock price as a function of the number of periods, to test the precision of the forecast over time. The average percentage difference between the two time series is 15,24%

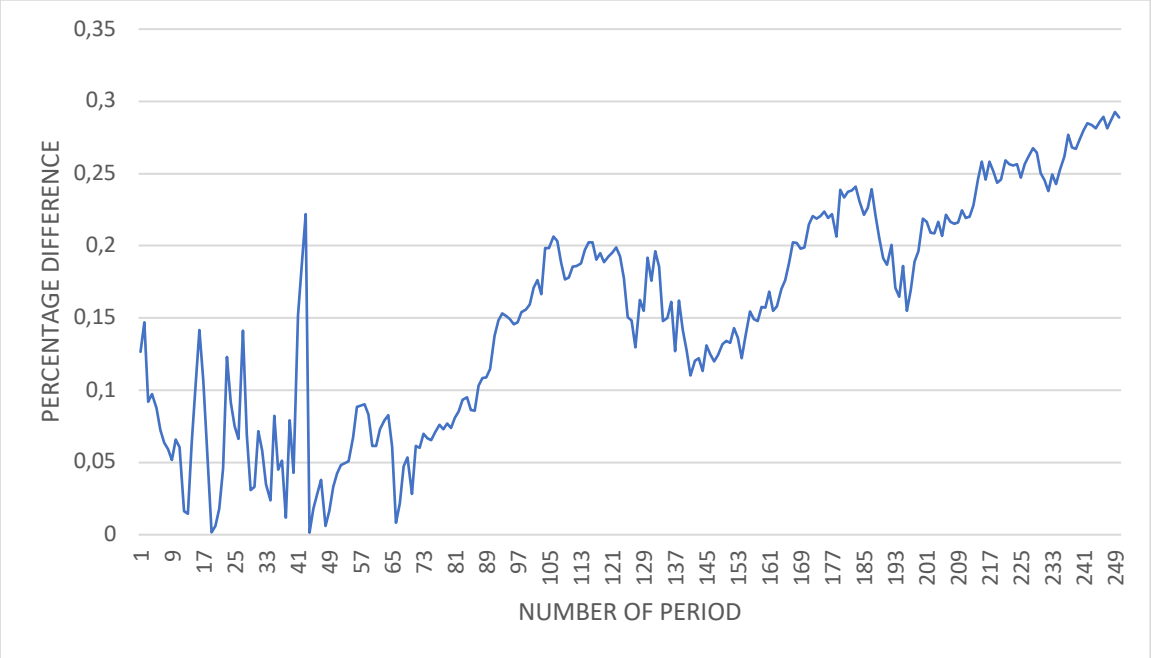

Figure 3.11 – absolute value of the percentage difference between predicted and realized stock price, for UCG.MI as a function of number of periods of forecasting.

The path in the graph shows no correlation between the number of periods of forecast and the loss of precision of the model for the first 70 lag. After the 70th period the difference between forecast and realized stock price is increasing. This behaviour is also captured by the correlation index, calculated on the two time series, that is 0,87.

3.3 Portfolio optimization

Following the framework of the previous chapter, in this section we will use forecast of stock price in Markowitz portfolio optimization. In this section we optimize benchmark portfolio before, and then the machine learning portfolio.

The first step for portfolio optimization is to retrieve daily stock returns, starting from the dataset containing daily closing prices. Daily stock returns are calculated as:

$$r(n)_t = \frac{p_t - p_{t-1}}{p_{t(t-1)}}$$

Where $r(n)_t$ is the return realized by the generic stock "n" in a generic working day "t".
Next, we calculated the average return for each stock, denoted by μ_1, μ_2, ..., μ_n, the variance of the returns for each stock, denoted by σ_1^2, σ_2^2, ..., σ_n^2. Finally, we computed the covariance between each pair of stocks, denoted by σ_{ij}, where i and j are indices that represent the two stocks. With these data for all the training set length it is possible to compute the variance-covariance matrix, also known as the covariance matrix, that is a square matrix that shows the variances and covariances of the set of stocks. The variance-covariance matrix for the set of n stocks can be represented by an n x n matrix, where the (i, j)th element of the matrix represents the covariance between stock i and stock j. The diagonal elements of the matrix (i.e., the elements where i=j) represent the variances of the individual stocks, while the off-diagonal elements represent the covariances between pairs of stocks. Table A.1 in the appendix B represents the variance-covariance matrix.

Figure 3.12 shows the average return for every stock considered and the variance of the returns. The graph shows negative average return for most of the stocks in the FTSE-MIB. This is the case especially for stocks in financial industry, that are numerous in the index. Banks stocks suffered the 2008 great financial crisis, which hit particularly financial institutions, but also the sovereign debt crisis after 2011. The sovereign debt crisis, that involved Italy public debt, resulted in a great increase of the public debt perceived risk, and a consequent increase of the

interest rate on government bonds. The situation was particularly critical for Italian banks, because of their massive investment on public debt: the increase of interest rate on newly issued government bond, made banks assets to depreciate, generating impact on their balance sheets and on their liquidity. Also, in 2016 another storm hit financial institutions: Monte dei Paschi di Siena crisis, and then bankruptcy of minor banks in Veneto, raised concern on italian baking system stability.

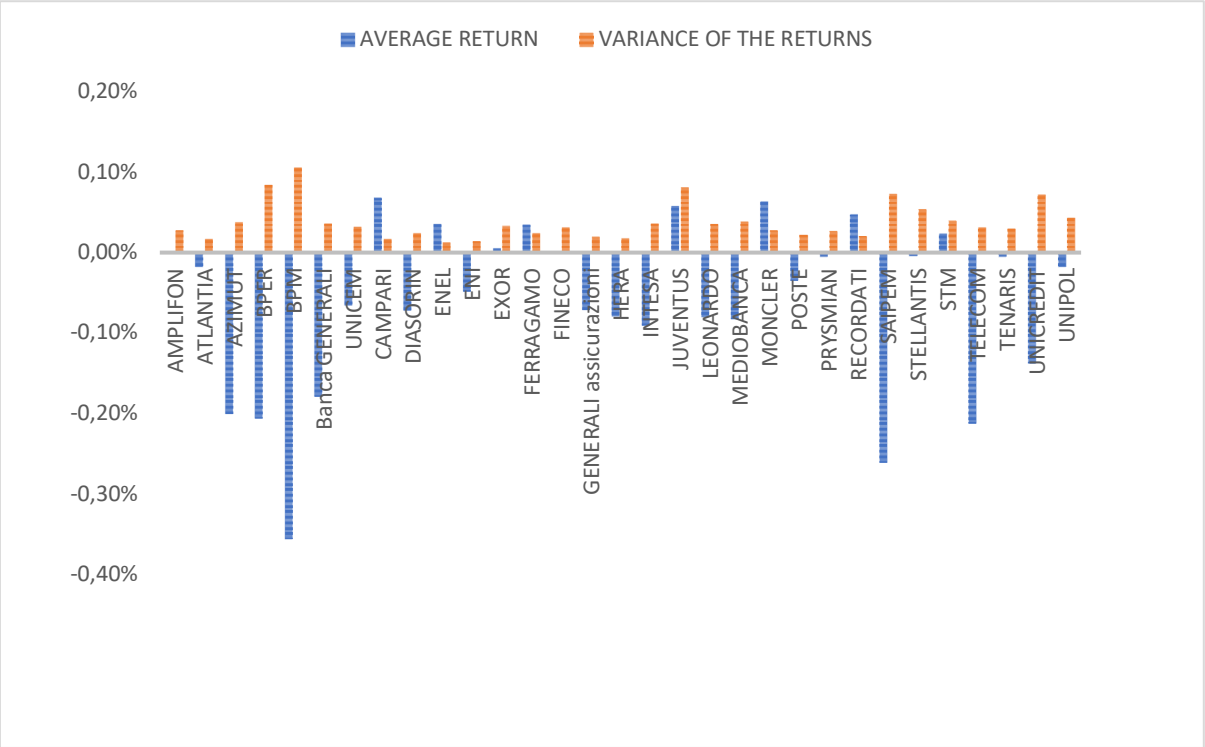

Figure 3.12 – average daily return and variance in the period 2000-2018, for FTSE-MIB stocks.

The optimal portfolio will be composed of two asset classes: risky asset and risk-free asset. Since in real world finance risk-free asset doesn't exist, we had to select a proxy. As a proxy of the risk-free rate, an Italian Government obligation has been chosen. Since the temporal horizon of the portfolios is one year: from January 2019 to December 2019, also the risk-free asset must have the same duration. The most suitable type of obligation is a BOT, a Buono Ordinario del Tesoro, issued by "Ministero delle Economie e delle Finanze". In Italy, the BOTs have duration six month or one year, and they are zero-coupon bonds: they don't pay any coupons, and, at their expiration, they pay their nominal value. In our model the chosen BOT was: IT0005355570, issued in December 2018, duration one year (expiration December 2020), with expected return of 0,37% annually.

Markowitz optimization model uses forecast of future returns of stocks and their expected variances and covariances to select the best portfolio. Regarding the benchmark portfolio, we used the average of the past returns for every stock as the expected future returns. To build the portfolio, we first drew the efficient frontier line that can be achieved with the given set of stocks. The minimum variance frontier is composed of all the portfolios that, for a given level of expected return, have the minimum expected standard deviation, so the minimum risk. Recalling the conditions

$$r_p = \alpha * r_r + \beta * r_f$$

$$Var(r_p) = \alpha * r_r$$

and the no short-sell boundary, we found the minimum variance frontier with the "solver" tool, in excel. With the solver we minimized the expected standard deviation of the portfolio, for different levels of expected returns. Figure 3.13 shows the minimum variance frontier built to find the benchmark portfolio. As we expect, higher levels of risk lead to higher levels of return, defining the risk-return trade-off.

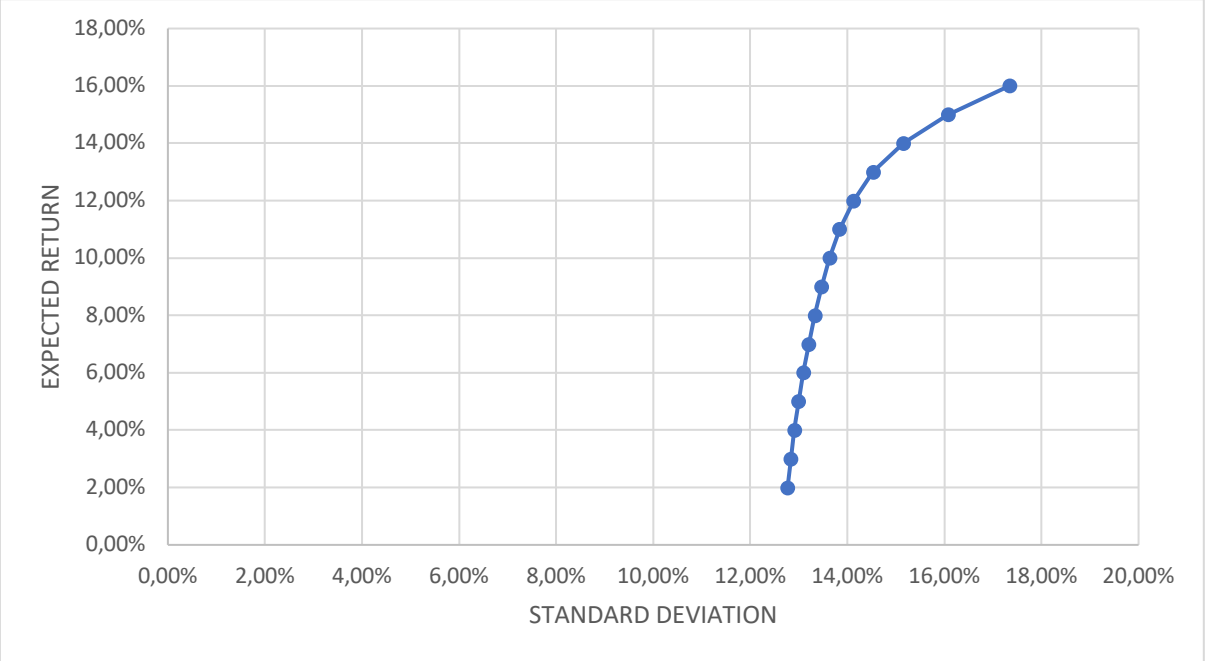

Figure 3.13 - minimum variance frontier. In the horizontal axis the expected standard deviation of the portfolio, in the vertical axis the expected return of the portfolio.

Following the Markowitz framework, the following step is to identify the optimal risky portfolio, the portfolio that maximizes the Sharpe Ratio. The Sharpe Ratio measures the excess return per unit of risk of an investment, where the excess return is the difference between the

expected return of the investment and the risk-free rate of return. A higher Sharpe Ratio indicates that an investment has provided better returns for the amount of risk taken, while a lower Sharpe Ratio suggests that an investment has not provided as much return for the amount of risk taken. It is calculated as:

$$SR_p = \frac{(r_p - r_f)}{\sigma_p}$$

Where r_p is the expected return of the risky portfolio, r_f is the risk-free rate, and σ_p is the expected standard deviation of the risky portfolio.

Once we obtained the Sharpe Ratio of the optimal risky portfolio, we drew the CAL, the Capital Asset Line, which is the line that, by definition, in a Cartesian plane where in the abscissas is expressed the expected standard deviation of the investment strategy, and in the ordinates the expected return, starts from the point (0; r_f), and it has a slope equal to the Sharpe Ratio of the optimal risky portfolio. The average european risk aversion coefficient of 2,5 can be translated in a desired expected standard deviation of 13,81%. Since the optimal risky portfolio has a greater standard deviation, to reach the target standard deviation of 13,81%, the investor must move along the CAL.

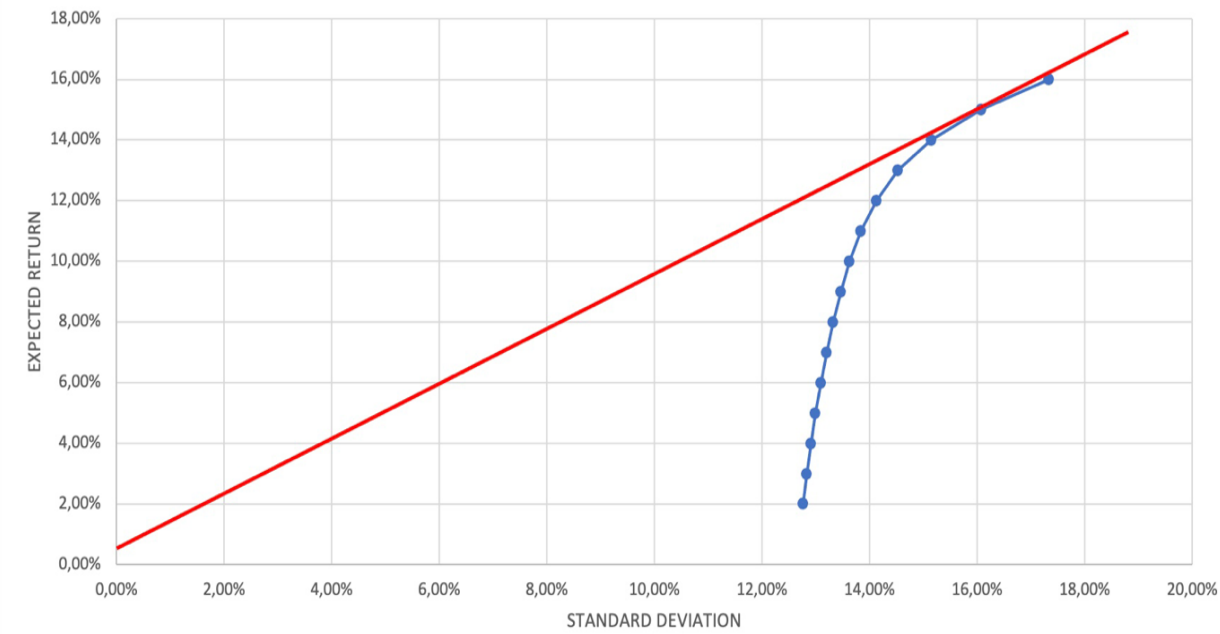

Figure 3.14 - minimum variance frontier and capital asset line. In the horizontal axis the expected standard deviation of the portfolio, in the vertical axis the expected return of the portfolio.

Moving along the CAL, the investor decreases the proportion of risky asset in his portfolio, while she increases the proportion of risk-free asset, in this case BOT. It is clear that also the expected return of the investment strategy is decreased. The final benchmark portfolio is the portfolio that is composed by a proportion of risk-free asset and a proportion of risky assets (the optimal risky portfolio), it lies in the CAL and its variance match the investor's risk preferences. This portfolio, b, has coordinates:

$$\sigma_b = \sigma_r * \%\ of\ risky\ asset + \sigma_{rf} * \%\ of\ risk\ free\ asset$$
$$\mu_b = \mu_r * \%\ of\ risky\ asset + \mu_{rf} * \%\ of\ risk\ free\ asset$$

Where, σ_b is the expected standard deviation of the benchmark portfolio, and μ_b is the expected return of the benchmark portfolio

In the figure 3.15 we represent the optimal risky portfolio stock allocation.

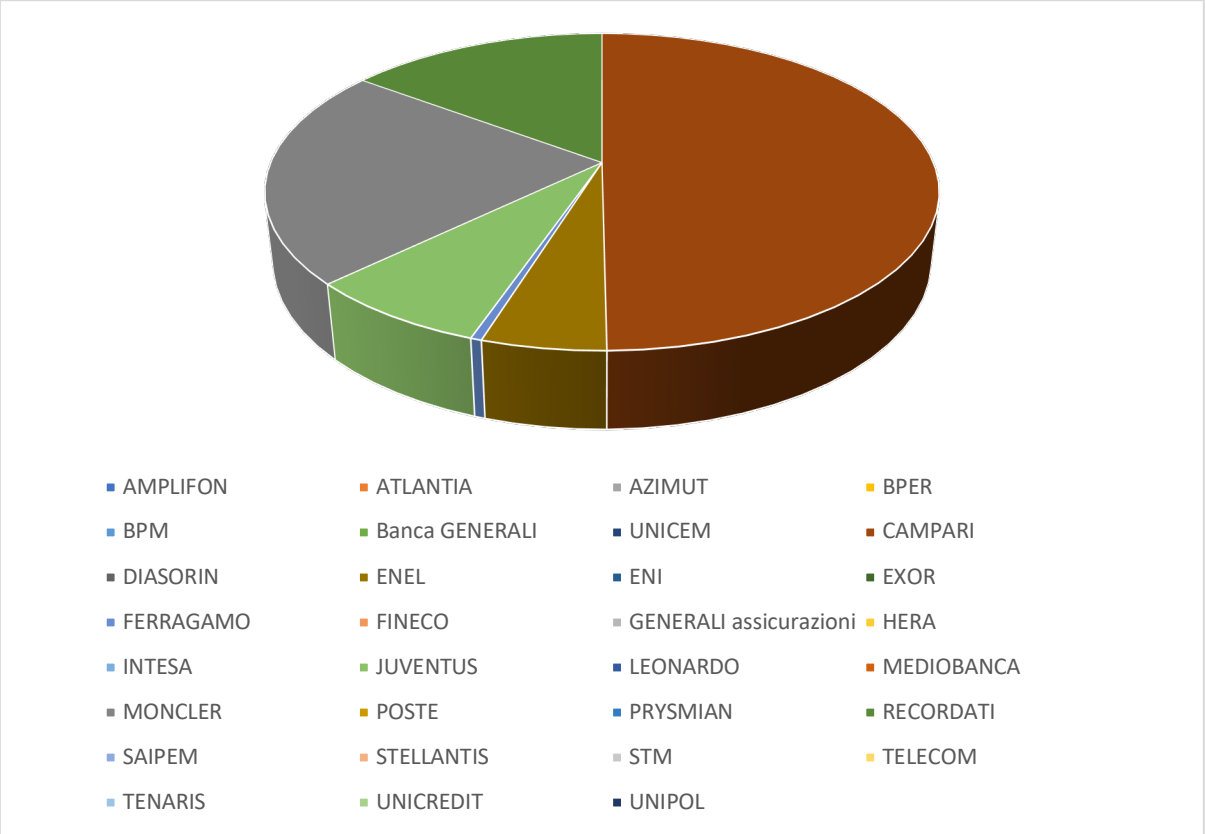

Figure 3.15 – benchmark portfolio allocation of risky asset

The benchmark portfolio shows a lack of diversification: only six of the stock in the basket have been selected to compose the portfolio. This is because the remaining stocks show either

negative returns in the past, or too much standard deviation of the returns. Table 3.4 summarizes the main features of the benchmark portfolio:

	Expected return	Expected standard deviation	% Allocation
Optimal Risky Portfolio	15,19%	16,28%	84,84%
Risk Free Asset	0,37%	0	15,16%
Benchmark Portfolio	12,94%	13,81%	n.a.

Table 3.4 – asset allocation of the benchmark portfolio, with expected return and expected standard deviation of the risky component and the risk-free component.

Similar steps have been followed to optimize the machine learning portfolio. Recalling the dataset, the forecast of future returns are the predicted closing stock prices for the 2019 working days. One of the theoretical main advantages of this kind of estimation is to have forecasts that changes during the year, and, consequently, the portfolio manager can tune its portfolio to match the evolution of the predictions. To do this we decided to set our investment strategy as follow:

1. We divided 2019 in 12 periods of 20 working days (about one month), setting the first working day of the year as time t=0, where t are the working days. We then calculated, for every stock, the predicted return of buying and holding the stock for 20 working days, simply calculating the percentage difference between the predicted stock price at time t+20 and time t, starting from t=0.
2. We used those return forecast as the input to optimize 12 different portfolios, using Markowitz model.
3. The resulting machine learning portfolio is a portfolio that changes every 20 days, adapting to new return forecasts.

The risk component of the machine learning portfolio is the same of the benchmark portfolio. We used the same historical data on variance and correlation between the stocks to come up with the same variance-covariance matrix. With the return forecast, and the variance-covariance matrix, the usual optimization process has been implemented. For the selection of risky asset, we didn't set any boundary, except the short sale constraint; in

figures A.2 to A.13 in the appendix, we sum up the evolution of the risky asset allocation during 2019.

With the introduction of the risk-free asset, we came up with the capital asset lines, for every portfolio. We then set the standard deviation that makes benchmark portfolio and the machine learning portfolio comparable: we set the standard deviation of the 12 portfolios (annualized) equal to the standard deviation of the benchmark portfolio, 13,81%. The resulting asset allocation is reported in the table 3.5, together with the annualized expected return for each portfolio.

PORTFOLIO	*ST. DEVIATION RISKY ASSET*	*EXPECTED RETURN RISKY ASSET*	*% RISKY ASSET*	*% RISK FREE*
1	0,24	14,87	58,01%	41,99%
2	0,21	40,01	65,94%	34,06%
3	0,18	28,44	78,49%	21,51%
4	0,18	8,20	77,19%	22,81%
5	0,18	9,07	76,44%	23,56%
6	0,19	12,15	73,98%	26,02%
7	0,19	12,11	73,52%	26,48%
8	0,20	11,71	70,41%	29,59%
9	0,23	15,64	59,27%	40,73%
10	0,22	18,06	61,41%	38,59%
11	0,23	20,23	60,12%	39,88%
12	0,24	27,68	56,83%	43,17%

Table 3.5 – expected return, standard deviation, and asset allocation, represented for each of the twelve machine learning portfolios

3.4 Portfolio evaluation

To evaluate the portfolios found in the previous section we confronted them with the effective result of the underlying stocks. The true return calculation method is slightly different in the benchmark portfolio and in the machine learning portfolio, due to the twelve rebalances.

For the benchmark portfolio we needed the stocks performances during the entire 2019 year, so we calculated the stock price difference (for each stock) between last day of 2019 and the first day of 2019. We then proportioned the resulting gain or loss to a hypothetical investment of 100 Euros. To compute the benchmark portfolio performance, we multiplied the stock weight

in the portfolio for its gain/loss during 2019. Summing all the weighted gain/loss, we had the overall portfolio return, on an investment of 100 Euros.

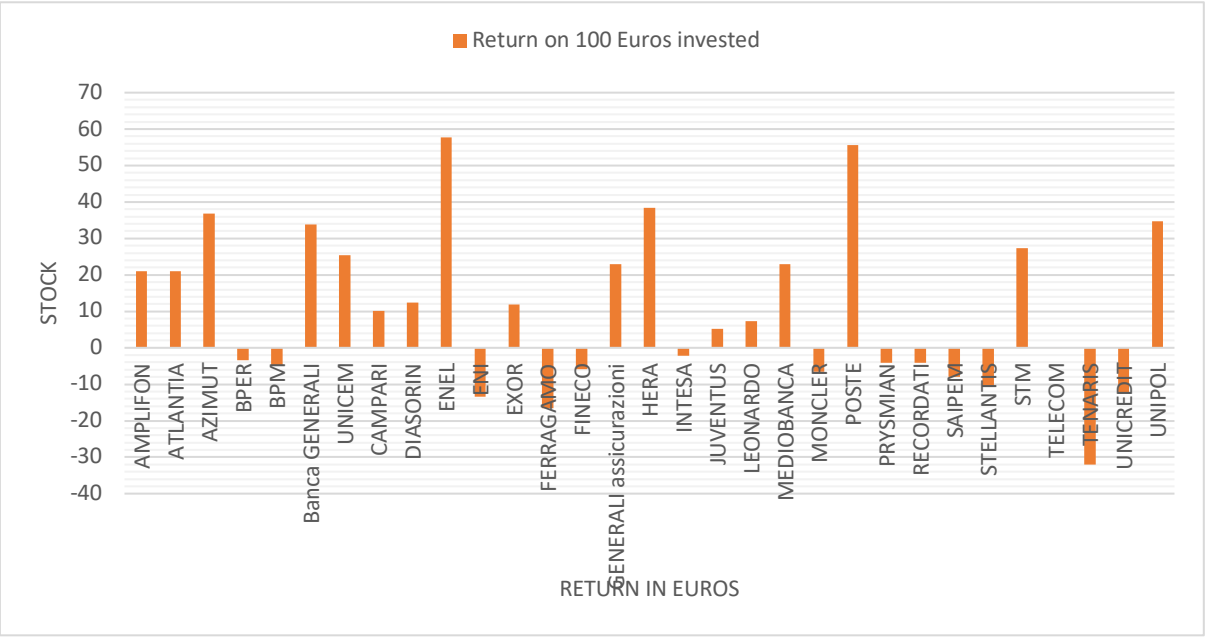

Figure 3.16 – return on 100 euros invested for the stocks of FTSE-MIB, holding the stocks for the entire 2019 year.

Figure 3.16 shows the real return for every stock in the basket. The returns are expressed in Euros, and they are referred to 100 euros of invested capital. As it is possible to see from the picture, in 2019, the stocks that performed better are ENEL and Poste Italiane, while Tenaris, UniCredit and Ferragamo performed poorly. Campari, that weight almost 50% in the benchmark portfolio, realized a positive return of 10%, while Moncler, that is the second stock in the portfolio suffered a negative return of 6,55%.

Overall, the benchmark portfolio, realized a profit on risky asset of 6,15 Euros, on 100 Euros invested. Considering the asset allocation between risky and risk-free asset, the portfolio realized a total return of 5,22 Euros.

The machine learning portfolio performance have been computed in a similar way. Again, the stock price difference has been calculated, but with different time horizons. We calculated, indeed, the stock price difference every 20 working days (that is the same time length of the twelve machine learning portfolios), obtaining the stock performance. We, then weighted the stock performance following the twelve machine learning portfolio, obtaining the realized return of risky asset for those portfolio. Once included the asset class allocation, we

obtained the realized return for each portfolio. The cumulative return of the machine learning investment strategy is the sum of the 12 portfolios returns.

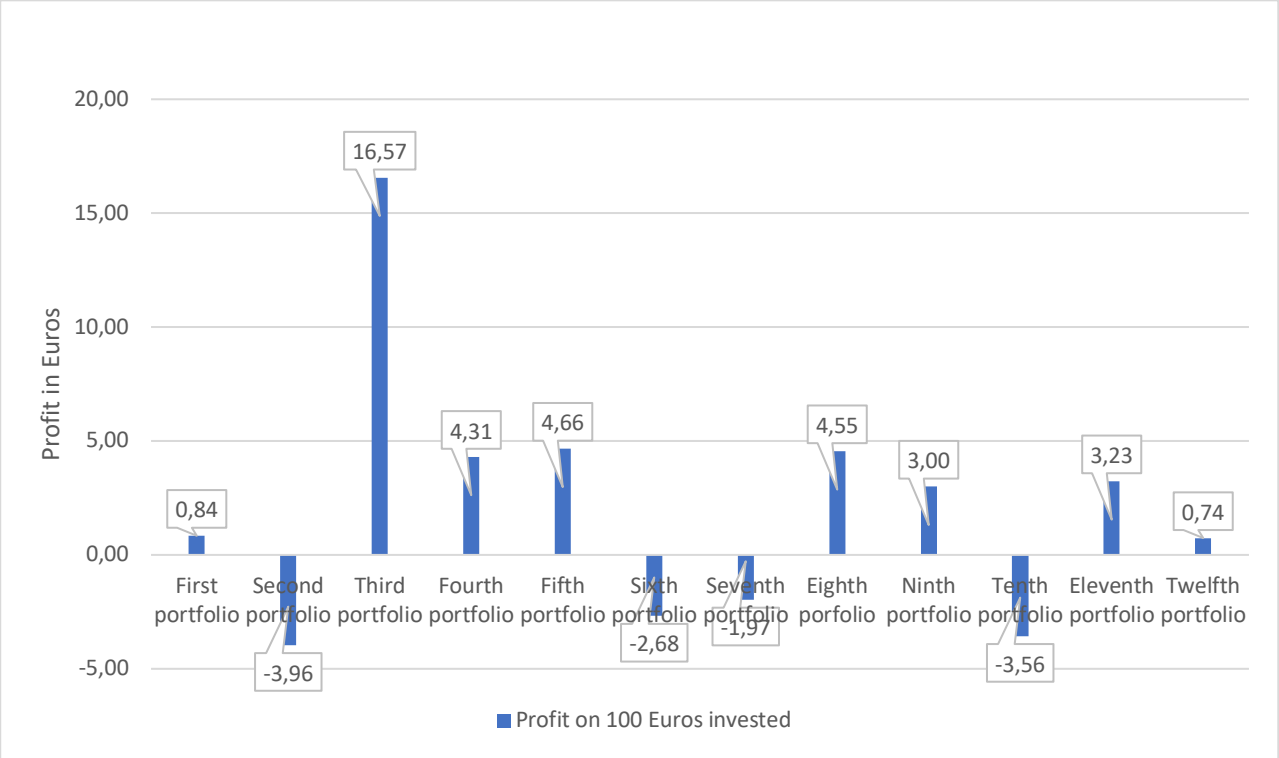

Figure 3.17 profit, in euros, of the twelve machine learning portfolios, in the hypothesis of 100 euros invested.

Figure 3.17 present the total profit in Euros (on 100 Euros invested), for each of the twelve portfolios estimated with machine learning model. Seven portfolios show a positive real return, with a peak in the second portfolio (month of February); still, the average return of the portfolios is 2,15 euros. The cumulative return of the twelve portfolio is 25,74 euros, with a delta of 20,52 euros with respect to the benchmark portfolios.

There is one more thing to consider in the comparison: the transaction costs. These kinds of costs are the fees that the stock exchange, in this case Borsa Italia, applies to the market operator when they trade on their platforms. The transaction costs vary a lot, and they depend on variables such as bargaining power of the fund house, speed of transaction execution, number of transactions per year. In our model, following Kara et al. (2019) work, we assume transaction costs of 0,2% of the value of the transaction, either sell or buy. In the rebalancing process, we calculated the transaction costs on the nominal value of the transaction, excluding the stock that remain in the portfolio.

100 EUROS INVESTED	BENCHMARK PORTFOLIO	MACHINE LEARNING PORTFOLIO
PROFIT WITHOUT TRANSACTION COSTS	5,22 Euros	25,74 Euros
TOTAL TRANSACTION COSTS	0,20 Euros	1,04 Euros
PROFIT WITH TRANSACTION COSTS	5,02 Euros	24,70 Euros

Table 3.6 comparison of 100 Euros investment in Benchmark portfolio, and machine learning portfolio.

In table 3.6, we present the net result of the two portfolios, with the transaction costs and without them. Despite transaction costs decrease the profit of machine learning portfolio by 4,04%, this is still more convenient of the benchmark one.

CONCLUSION

With the development of new technologies, like AI and machine learning, their application in financial market should be considered. In our opinion the implementation of "computer forecasting" can be considered as revolutionary as Markowitz theory. This thesis aim was to propose a concrete example of portfolio building model using machine learning predictions, through a new approach inspired by Black and Litterman model.

To sum up the work conducted so far, we firstly put relevance on technical analysis indicators, that are useful instruments to express the main characteristic of the stock price behavior. Those characteristics are basically the variance and the stock performance, and every indicator considered measures them in a specific way. We choose as the starting dataset the stock composing Italian FTSE-MIB index, that expresses the great majority of the total Borsa Italiana capitalization. We then estimated, with the classical ARMA model, the future value of the stock's indicators, to create the starting point for the machine learning forecasts. Support Vector Machine technique has been applied to the forecasted indicators. They are the features of the SVM, used for the prediction of the future returns of the considered stocks.

In the final part of the dissertation, we implemented the portfolio optimization, using the row data derived from the previous steps. In this section, we built a benchmark portfolio, using Markowitz optimization. We used, for the benchmark, the mean of historical return as the forecast for future returns. In a similar manner, we used past data to derive the variance-covariance matrix. We also used Markowitz optimization to build the machine learning portfolio, but in this latter case we used SVM results as forecasts of future returns. For the machine learning portfolio, we divided the investment period of one year in 12 sub-periods; we built a different portfolio for each of the twelve periods, simulating a portfolio rebalance.

To make benchmark portfolio and machine learning portfolio comparable, we constraint the optimization to have the same expected variance in the two portfolios. We used the realize stock prices to value the portfolios performances, for the 2019 year. Empirical evidence shows the superiority, in terms of absolute returns, of the machine learning portfolio, over the benchmark one. Adding the transaction costs to the model, profits of machine learning strategy shrink, but they are still greater than traditional Markowitz optimization.

Of course, this thesis has some limitation, that can be analyzed in future research. First of all, the model has been applied in a restricted sample, both in geographic term and in time term. The analysis has been conducted only to Italian stocks, and only for the 2019 year. Further experiments may test the model to other stocks or other time windows. Furthermore, the only measure of risk considered in the thesis is the variance of daily return, other measures of risk should be considered, as expected shortfall, or value at risk. One last point to consider is that from the recent literature, promising results emerge in hybrid machine learning and deep learning model applications. More research on this topic and implementation in the model proposed in this in the previous chapters may increase portfolio performance.

APPENDIX A

We used python and various libraries running in python to solve most of the thesis issues. Specifically, we used python to retrieve financial data from Yahoo Finance source and to calculate the value of indicators. Then, we used it to fit the ARMA model and to run all the preliminary analysis. Finally, SVM model has also been implemented in Python. In this appendix, we report entirely the code used, with the description and some comments.

A.1 Retrieving financial data

```
# Importing required packages
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import pandas_datareader as web

# Setting the temporal boundaries
start = dt.datetime(2000, 1, 1)
end = dt.datetime(2019, 12, 31)

# Created a dataset, named “data”, containing opening price, closing price, and trade volumes, for AMP.MI, with daily frequency.
data = web.DataReader("AMP.MI", 'yahoo', start, end)
```

A.2 Indicators calculation

```
# Importing required packages
import talib as ta

# Calculation of stock indicators, with TA library, and add the daily results to “data” dataframe
data['ATR']=ta.ATR(data['High'], data['Low'], data['Close'])
data['EMA']=ta.EMA(data['Close'])
data['ADXR']=ta.ADXR(data['High'], data['Low'], data['Close'])
data['MACD'], data['MACDSIGNAL'], data['MACDHIST']=ta.MACD(data['Close'])
```

```
data['SMA']=ta.SMA(data['Close'])
data['RSI']=ta.RSI(data['Close'])

# Export the "data" to a CSV file named as the name of the stock
data.to_csv(r'/Users/andreamasiero/Desktop/output azioni/italia/ amplifon.csv')
```

A.3 ARMA model implementation

```
# Importing required packages
import seaborn as sns
sns.set()

# Importing Dataset
data = pd.read_csv(r"/Users/andreamasiero/Desktop/output azioni/italia/ amplifon.csv")
data=data.dropna() # drop the null rows of the dataset
data.Date = pd.to_datetime(data.Date)

# dividing the dataset in train and test set
train_df = data.loc[:"4822"]
test_df = data.loc["4822":]

# Importing required packages
from statsmodels.tsa.stattools import adfuller

# Performing the augmented Dickey-Fuller test, to check whether the time series is stationary
or not
adfuller(train_df['ATR'])
adfuller(train_df['EMA'])
adfuller(train_df['ADXR'])
adfuller(train_df['MACD'])
adfuller(train_df['SMA'])
adfuller(train_df['RSI'])

# Importing Required Package
```

```
import statsmodels.graphics.tsaplots as sgt

# Fixing plot size
plt.rcParams["figure.figsize"] = 18, 5

# Defining Subplots
fig, axes = plt.subplots(1, 2)

# Plotting ACF and PACF for ATR
sgt.plot_acf(train_df.ATR[1:], zero = False, lags = 40, ax = axes[0])
sgt.plot_pacf(train_df.ATR[1:], zero = False, lags = 40, ax = axes[1])

# Display the Plot
plt.show()
```

A.4 ARMA Model fitting

```
# Importing Required Package
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Defining the Model
model = SARIMAX(train_df["ATR"][1:], order = (10, 0, 10))

# Fitting the Model
model_results = model.fit()

# Printing the model summary
print(model_results.summary())
```

A.5 ARMA Model forecasting

```
# Building Forecast
arma_forecast = model_results.get_forecast(len(test_df.index))
```

 # Includin the predictions on the dataset "arma_predictions_df". Every column of the dataset contains the daily predictions of on indicator

```
arma_predictions_df ["i"] = model_results.predict(start = test_df.index[0], end = test_df.index[-1])
```

A.6 Preparing the data to the SVM implementation

 # Drop the columns with non-relevant data

```
train_solo_indici = train_df.drop(train_df.columns[[0, 1, 2, 3, 4, 5, 6]], axis=1)
arma_predictions_df = arma_predictions_df.drop(arma_predictions_df.columns[[0, 1]], axis=1)
```

 # Creating a dataset, containing only indicators value. It contains realized values until 01/01/2019 and predicted values after 01/01/2019

```
indici = pd.concat([train_solo_indici, arma_predictions_df])
```

 # Creating a new dataset, containing realized indicators values until 01/01/2019 and predicted values after 01/01/2019. It also contains realized daily close price, in the column "Close"

```
DF = pd.DataFrame(data, columns=['Close'])
indici=indici.reset_index()
DF['ATR'] = indici['ATR']
DF['EMA'] = indici['EMA']
DF['ADXR'] = indici['ADXR']
DF['MACD'] = indici['MACD']
DF['MACDSIGNAL'] = indici['MACDSIGNAL']
DF['SMA'] = indici['SMA']
DF['RSI'] = indici['RSI']
```

 # importing required packages

```
from sklearn.model_selection import train_test_split
```

```
# Removing empty rows
DF = DF.dropna()

# Defining the variable y as the close price, and the vector X as the features of the SVM model, that are the indicator values
X = DF.iloc[:,1:].values
y = DF.iloc[:, :1].values
y = y.reshape(-1, 1)

# Dividing the dataset in train and test set
h=len(DF)
X_train, X_test = train_test_split(X, test_size=250/h, shuffle=False)
y_train, y_test = train_test_split(y, test_size=250/h, shuffle=False)

# Scaling all the datas. This is necessary to compare data with different magnitude
from sklearn.preprocessing import StandardScaler
X_sc = StandardScaler()
y_sc = StandardScaler()
X_train = X_sc.fit_transform(X_train)
y_train = y_sc.fit_transform(y_train)
```

A.7 SVM implementation

```
# Importing required packages
from sklearn.svm import SVR

# Training the model
regrassor = SVR(kernel = 'rbf') # in this line we also set the kernel used to implement the model, the rbf kernel.
regrassor.fit(X_train, np.ravel(y_train,order="c"))

# Using the SVM to get the predictions
y_pred = regrassor.predict(X_sc.transform(X_test))
```

 # Reversing the previously implemented standard scaling. All the data get back to their original magnitude

```
y_pred = y_sc.inverse_transform(y_pred)
```

A.8 Present the data

 # Create a dataset containing the realized closing price and the predicted closing price, for the considered time window

```
y_test = y_test.flatten()
df = pd.DataFrame({'Predicted value': y_pred})
df ['close price']= pd.DataFrame({'CLOSE PRICE': y_test})
```

 # Creating a new dataset, integrating the dataset "df" with the reaslized values of all the indicators for the stock considered

```
definitivo = pd.DataFrame(df, columns=['Predicted value'])
definitivo['prezzo realizzato ']=df['close price']
test_df.reset_index(drop=True, inplace=True)
arma_predictions_df.reset_index(drop=True, inplace=True)
definitivo ['ATR realizzato']=test_df['ATR']
definitivo ['ATR PREVISTO']=arma_predictions_df['ATR']
definitivo ['EMA realizzato']=test_df['EMA']
definitivo ['EMA PREVISTO']=arma_predictions_df['EMA']
definitivo ['ADXR realizzato']=test_df['ADXR']
definitivo ['ADXR PREVISTO']=arma_predictions_df['ADXR']
definitivo ['MACD realizzato']=test_df['MACD']
definitivo ['MACD PREVISTO']=arma_predictions_df['MACD']
definitivo ['MACDSIGNAL realizzato']=test_df['MACDSIGNAL']
definitivo ['MACDSIGNAL PREVISTO']=arma_predictions_df['MACDSIGNAL']
definitivo ['SMA realizzato']=test_df['SMA']
definitivo ['SMA PREVISTO']=arma_predictions_df['SMA']
```

 # Exporting the dataset to an Excel file named as the stock considered

```
definitivo.to_excel(r'/Users/andreamasiero/Desktop/prezzi azioni/ ' + k + '.xlsx')
```

Note that the reported code applies only to a stock, Amplifon, that ticker has been specified in the ‘retrieving financial data’ section. In the original code we used the loops to run only once the Python program. For presenting needs, here we reported the code without the loops.

APPENDIX B

This appendix contains all the data that for space reasons we couldn't insert in the previous chapter. Specifically figure A.1 shows a detailed composition of the FTSE-MIB index, table A.1 expresses the variance-covariance matrix used for portfolios optimization, and figures A.2 to A.13 graphically show the compositions of all the machine learning risky portfolios.

BORSA ITALIANA
CAPITALISATION FTSE MIB BASKET ON 31.01.2019

		eur m	% FTSE MIB DOMESTIC	% FTSE MIB TOTAL	%BORSA DOMESTIC	%BORSA TOTAL
1	ENI	53 797.6	12.41%	12.05%	9.29%	9.09%
2	ENEL	53 436.1	12.32%	11.97%	9.23%	9.03%
3	INTESA SANPAOLO	35 160.8	8.11%	7.88%	6.07%	5.94%
4	GENERALI	24 007.6	5.54%	5.38%	4.15%	4.06%
5	FIAT CHRYSLER AUTOMOBILES	22 996.4	5.30%	5.15%	3.97%	3.89%
6	UNICREDIT	22 635.1	5.22%	5.07%	3.91%	3.82%
7	FERRARI	20 312.5	4.68%	4.55%	3.51%	3.43%
8	ATLANTIA	17 097.2	3.94%	3.83%	2.95%	2.89%
9	SNAM	14 469.0	3.34%	3.24%	2.50%	2.44%
10	EXOR	13 312.5	3.07%	2.98%	2.30%	2.25%
11	TENARIS	12 941.5	2.98%	2.90%	2.23%	2.19%
12	STMICROELECTRONICS	12 644.5	2.92%	2.83%	2.18%	2.14%
13	CNH INDUSTRIAL	11 792.0	2.72%	2.64%	2.04%	1.99%
14	TERNA	10 788.9	2.49%	2.42%	1.86%	1.82%
15	POSTE ITALIANE	9 796.5	2.26%	2.20%	1.69%	1.66%
16	CAMPARI	9 073.8	2.09%	2.03%	1.57%	1.53%
17	MONCLER	8 378.6	1.93%	1.88%	1.45%	1.42%
18	TELECOM ITALIA	7 338.4	1.69%	1.64%	1.27%	1.24%
19	MEDIOBANCA	6 783.3	1.56%	1.52%	1.17%	1.15%
20	RECORDATI	6 520.6	1.50%	1.46%	1.13%	1.10%
21	UNIPOLSAI	6 200.5	1.43%	1.39%	1.07%	1.05%
22	FINECOBANK	5 807.8	1.34%	1.30%	1.00%	0.98%
23	PIRELLI & C	5 761.8	1.33%	1.29%	0.99%	0.97%
24	PRYSMIAN	5 024.5	1.16%	1.13%	0.87%	0.85%
25	A2A	4 977.3	1.15%	1.12%	0.86%	0.84%
26	LEONARDO	4 883.1	1.13%	1.09%	0.84%	0.83%
27	DIASORIN	4 476.5	1.03%	1.00%	0.77%	0.76%
28	ITALGAS	4 256.1	0.98%	0.95%	0.73%	0.72%
29	SAIPEM	4 175.5	0.96%	0.94%	0.72%	0.71%
30	AMPLIFON	3 538.2	0.82%	0.79%	0.61%	0.60%
31	BREMBO	3 324.7	0.77%	0.74%	0.57%	0.56%
32	SALVATORE FERRAGAMO	2 951.6	0.68%	0.66%	0.51%	0.50%
33	UNIPOL	2 875.9	0.66%	0.64%	0.50%	0.49%
34	BUZZI UNICEM	2 761.6	0.64%	0.62%	0.48%	0.47%
35	BANCO BPM	2 586.0	0.60%	0.58%	0.45%	0.44%
36	UBI BANCA	2 581.8	0.60%	0.58%	0.45%	0.44%
37	BANCA GENERALI	2 403.8	0.55%	0.54%	0.42%	0.41%
38	AZIMUT HOLDING	1 581.8	0.36%	0.35%	0.27%	0.27%
39	BPER BANCA	1 436.0	0.33%	0.32%	0.25%	0.24%
40	JUVENTUS FOOTBALL CLUB	1 413.9	0.33%	0.32%	0.24%	0.24%
	TOTAL FTSE MIB (DOMESTIC)	**433 657.0**				
	TOTAL FTSE MIB (+ FOREIGN)	**446 301.6**				
	TOTAL (DOMESTIC)	**579 164.6**				
	TOTAL (+ FOREIGN)	**591 809.1**				
	FTSE MIB IN % BORSA (DOMESTIC)	**74.88%**				
	FTSE MIB IN % BORSA (+ FOREIGN)	**75.41%**				

Figure A.1 detail of the stocks composing FTSE-MIB, in January 2019. For every stock the weight on the FTSE-MIB total capitalization and on the total "Borsa Italiana" capitalization is expressed. Source: Borsa Italiana S.p.A.

	AMPLIFON	ATLANTIA	AZMUT	BPER	BPM	BANCA GENERALI	UNICREDIT	CAMPARI	DIASORIN	ENEL	ENI	EXOR	FERRAGAMO	FINECO	GENERALI ASSICURAZIONI	HERA	INTESA	JUVENTUS	LEONARDO	MEDIOBANCA	MONCLER	POSTE ITALIANE	PRYSMIAN	RECORDATI	SAIPEM	STELLANTIS	STM	TELECOM	TENARIS	UNICREDIT	UNIPOL
AMPLIFON	0,02749%	0,00450%	0,00692%	0,00746%	0,00608%	0,00593%	0,00706%	0,00521%	0,00536%	0,00328%	0,00419%	0,00754%	0,00471%	0,00544%	0,00382%	0,00421%	0,00492%	0,00469%	0,00700%	0,00559%	0,00816%	0,00599%	0,00549%	0,00550%	0,00771%	0,00913%	0,00547%	0,00402%	0,00566%	0,00701%
ATLANTIA	0,00450%	0,01700%	0,00693%	0,01153%	0,01128%	0,00706%	0,00747%	0,00560%	0,00446%	0,00745%	0,00548%	0,00977%	0,00429%	0,00811%	0,00663%	0,00530%	0,00889%	0,00564%	0,00646%	0,00923%	0,00604%	0,00674%	0,00794%	0,00502%	0,00559%	0,00812%	0,00647%	0,00876%	0,00408%	0,01088%	0,00900%
AZMUT	0,00692%	0,00693%	0,03794%	0,02917%	0,02666%	0,02234%	0,01495%	0,00551%	0,00582%	0,00687%	0,00969%	0,01772%	0,00874%	0,01566%	0,01498%	0,00559%	0,02073%	0,00892%	0,01626%	0,02178%	0,01036%	0,01349%	0,01308%	0,00503%	0,02036%	0,02031%	0,01215%	0,01303%	0,01241%	0,02703%	0,02313%
BPER	0,00746%	0,01153%	0,02917%	0,08430%	0,07089%	0,02799%	0,02029%	0,00687%	0,00412%	0,01256%	0,01590%	0,02625%	0,01313%	0,02560%	0,02600%	0,00639%	0,03830%	0,01004%	0,02322%	0,04048%	0,00995%	0,02168%	0,01808%	0,00704%	0,02863%	0,02710%	0,01560%	0,02276%	0,01697%	0,05551%	0,03869%
BPM	0,00608%	0,01128%	0,02966%	0,07089%	0,10593%	0,02961%	0,02113%	0,00579%	0,00392%	0,01359%	0,01793%	0,02824%	0,01353%	0,02895%	0,02610%	0,00726%	0,04295%	0,01281%	0,02506%	0,04459%	0,01127%	0,02466%	0,01924%	0,00705%	0,03240%	0,03061%	0,01788%	0,01316%	0,00986%	0,02585%	0,02111%
Banca GENERALI	0,00593%	0,00706%	0,02234%	0,02799%	0,02981%	0,03637%	0,01433%	0,00493%	0,00577%	0,00745%	0,00801%	0,01516%	0,00702%	0,01805%	0,01476%	0,00710%	0,02020%	0,01011%	0,01529%	0,02231%	0,00799%	0,01425%	0,01242%	0,00589%	0,01615%	0,01788%	0,01128%	0,01316%	0,00986%	0,02585%	0,02111%
UNICEM	0,00706%	0,00747%	0,01495%	0,02029%	0,02113%	0,01433%	0,03222%	0,00671%	0,00719%	0,00678%	0,00815%	0,01614%	0,00816%	0,01260%	0,01096%	0,00382%	0,01488%	0,00708%	0,01467%	0,01493%	0,00996%	0,01043%	0,01524%	0,00781%	0,01480%	0,01809%	0,01504%	0,01087%	0,01225%	0,01757%	0,01667%
CAMPARI	0,00521%	0,00560%	0,00551%	0,00687%	0,00579%	0,00493%	0,00671%	0,01769%	0,00522%	0,00562%	0,00469%	0,00762%	0,00470%	0,00557%	0,00412%	0,00524%	0,00458%	0,00328%	0,00721%	0,00500%	0,00658%	0,00475%	0,00627%	0,00616%	0,00609%	0,00718%	0,00661%	0,00591%	0,00392%	0,00584%	0,00703%
DIASORIN	0,00536%	0,00446%	0,00582%	0,00412%	0,00392%	0,00577%	0,00719%	0,00522%	0,02477%	0,00375%	0,00314%	0,00576%	0,00393%	0,00538%	0,00289%	0,00309%	0,00352%	0,00291%	0,00488%	0,00496%	0,00761%	0,00568%	0,00703%	0,00623%	0,00578%	0,00576%	0,00776%	0,00419%	0,00325%	0,00354%	0,00545%
ENEL	0,00328%	0,00745%	0,00687%	0,01256%	0,01359%	0,00745%	0,00678%	0,00562%	0,00375%	0,01278%	0,00725%	0,00938%	0,00481%	0,00776%	0,00707%	0,00652%	0,00976%	0,00403%	0,00723%	0,00954%	0,00548%	0,00696%	0,00738%	0,00639%	0,00608%	0,00858%	0,00658%	0,00907%	0,00494%	0,01233%	0,00942%
ENI	0,00419%	0,00548%	0,00969%	0,01590%	0,01793%	0,00801%	0,00815%	0,00469%	0,00314%	0,00725%	0,01421%	0,01138%	0,00536%	0,00867%	0,00880%	0,00420%	0,01148%	0,00423%	0,00855%	0,01172%	0,00644%	0,00767%	0,00858%	0,00463%	0,01543%	0,01263%	0,00729%	0,00910%	0,01273%	0,01475%	0,01186%
EXOR	0,00754%	0,00977%	0,01772%	0,02625%	0,02824%	0,01516%	0,01614%	0,00762%	0,00576%	0,00938%	0,01138%	0,02389%	0,01013%	0,01598%	0,01383%	0,00442%	0,01861%	0,01040%	0,01605%	0,01890%	0,01070%	0,01262%	0,01564%	0,00819%	0,01761%	0,02848%	0,01673%	0,01323%	0,01316%	0,02458%	0,02132%
FERRAGAMO	0,00471%	0,00429%	0,00874%	0,01313%	0,01353%	0,00702%	0,00816%	0,00470%	0,00393%	0,00481%	0,00536%	0,01013%	0,02423%	0,00754%	0,00637%	0,00374%	0,00947%	0,00464%	0,00806%	0,00990%	0,00984%	0,00684%	0,00875%	0,00446%	0,00840%	0,01132%	0,00821%	0,00623%	0,00573%	0,01193%	0,00953%
FINECO	0,00544%	0,00811%	0,01566%	0,02560%	0,02895%	0,01805%	0,01260%	0,00557%	0,00710%	0,00776%	0,00867%	0,01598%	0,00754%	0,01388%	0,01279%	0,00670%	0,01912%	0,00896%	0,01299%	0,02007%	0,00744%	0,01312%	0,01174%	0,00553%	0,01464%	0,01599%	0,01085%	0,01109%	0,00979%	0,02706%	0,01818%
GENERALI assicurazioni	0,00382%	0,00663%	0,01498%	0,02600%	0,02610%	0,01476%	0,01096%	0,00412%	0,00289%	0,00707%	0,00880%	0,01383%	0,00637%	0,01279%	0,01999%	0,00344%	0,01807%	0,00600%	0,01235%	0,01983%	0,00709%	0,01133%	0,01058%	0,00492%	0,01376%	0,01388%	0,00896%	0,01187%	0,00799%	0,02357%	0,02101%
HERA	0,00421%	0,00530%	0,00559%	0,00639%	0,00726%	0,00710%	0,00382%	0,00524%	0,00309%	0,00652%	0,00420%	0,00442%	0,00374%	0,00670%	0,00344%	0,01842%	0,00489%	0,00484%	0,00509%	0,00599%	0,00433%	0,00638%	0,00401%	0,00478%	0,00528%	0,00515%	0,00401%	0,00510%	0,00282%	0,00674%	0,00578%
INTESA	0,00492%	0,00889%	0,02073%	0,03830%	0,04295%	0,02020%	0,01488%	0,00458%	0,00352%	0,00976%	0,01148%	0,01861%	0,00947%	0,01912%	0,01807%	0,00489%	0,03656%	0,00888%	0,01676%	0,02935%	0,00826%	0,01680%	0,01423%	0,00608%	0,01757%	0,02030%	0,01158%	0,01466%	0,01192%	0,03967%	0,02596%
JUVENTUS	0,00469%	0,00564%	0,00892%	0,01004%	0,01281%	0,01011%	0,00708%	0,00328%	0,00291%	0,00403%	0,00423%	0,01040%	0,00464%	0,00896%	0,00600%	0,00484%	0,00888%	0,08126%	0,00789%	0,00878%	0,00537%	0,00732%	0,00761%	0,00525%	0,00566%	0,01390%	0,01071%	0,00690%	0,00378%	0,00990%	0,00864%
LEONARDO	0,00700%	0,00646%	0,01626%	0,02322%	0,02506%	0,01529%	0,01467%	0,00721%	0,00488%	0,00723%	0,00855%	0,01605%	0,00806%	0,01299%	0,01235%	0,00509%	0,01676%	0,00789%	0,03551%	0,01772%	0,00927%	0,01189%	0,01329%	0,00784%	0,01621%	0,01781%	0,01367%	0,01101%	0,01085%	0,02035%	0,01790%
MEDIOBANCA	0,00559%	0,00923%	0,02178%	0,04048%	0,04459%	0,02231%	0,01493%	0,00500%	0,00496%	0,00954%	0,01172%	0,01890%	0,00990%	0,02007%	0,01983%	0,00599%	0,02935%	0,00878%	0,01772%	0,03903%	0,00908%	0,01884%	0,01410%	0,00579%	0,01765%	0,01923%	0,01178%	0,01513%	0,01089%	0,03793%	0,02846%
MONCLER	0,00816%	0,00604%	0,00923%	0,01626%	0,00761%	0,00638%	0,00761%	0,00444%	0,00704%	0,00794%	0,01172%	0,01890%	0,00984%	0,01074%	0,00709%	0,00433%	0,00537%	0,00793%	0,00727%	0,00878%	0,00537%	0,00908%	0,00764%	0,00646%	0,01144%	0,01162%	0,01012%	0,00535%	0,01272%	0,01439%	0,01187%
POSTE	0,00559%	0,00646%	0,01349%	0,02168%	0,02466%	0,01425%	0,01043%	0,00475%	0,00568%	0,00676%	0,01262%	0,00684%	0,01312%	0,01174%	0,01133%	0,00638%	0,01680%	0,00732%	0,01899%	0,01684%	0,00646%	0,02180%	0,01023%	0,00535%	0,01272%	0,01432%	0,00923%	0,01024%	0,00816%	0,01979%	0,01745%
PRYSMIAN	0,00549%	0,00794%	0,01308%	0,01808%	0,01924%	0,01242%	0,01524%	0,00627%	0,00703%	0,00738%	0,00858%	0,01564%	0,00875%	0,01174%	0,01058%	0,00401%	0,01423%	0,00761%	0,01329%	0,01410%	0,01144%	0,01023%	0,02711%	0,00752%	0,01129%	0,01477%	0,01547%	0,01045%	0,01047%	0,01617%	0,01425%
RECORDATI	0,00550%	0,00502%	0,00503%	0,00704%	0,00705%	0,00589%	0,00781%	0,00616%	0,00623%	0,00639%	0,00463%	0,00819%	0,00446%	0,00553%	0,00492%	0,00478%	0,00608%	0,00525%	0,00784%	0,00534%	0,00579%	0,00535%	0,00752%	0,02051%	0,00536%	0,00753%	0,00804%	0,00574%	0,00467%	0,00701%	0,00746%
SAIPEM	0,00771%	0,00559%	0,02036%	0,02863%	0,03240%	0,01615%	0,01480%	0,00609%	0,00578%	0,00608%	0,01543%	0,01761%	0,00840%	0,01464%	0,01376%	0,00528%	0,01757%	0,00556%	0,01621%	0,01765%	0,00985%	0,01272%	0,01129%	0,00536%	0,07311%	0,02106%	0,00865%	0,01171%	0,02267%	0,02337%	0,02146%
STELLANTIS	0,00913%	0,00812%	0,02031%	0,02710%	0,03061%	0,01788%	0,01809%	0,00718%	0,00566%	0,00858%	0,01263%	0,02848%	0,01132%	0,01599%	0,01388%	0,00515%	0,02030%	0,01390%	0,01781%	0,01925%	0,01162%	0,01432%	0,01477%	0,00753%	0,02106%	0,05367%	0,01934%	0,01300%	0,02588%	0,01709%	0,02336%
STM	0,00547%	0,00647%	0,01215%	0,01560%	0,01485%	0,01128%	0,01504%	0,00611%	0,00776%	0,00658%	0,00729%	0,01673%	0,00821%	0,01085%	0,00896%	0,00401%	0,01158%	0,01071%	0,01367%	0,01178%	0,01012%	0,00923%	0,01547%	0,00804%	0,00865%	0,01934%	0,03988%	0,00891%	0,01102%	0,01442%	0,01235%
TELECOM	0,00402%	0,00876%	0,01303%	0,02764%	0,02544%	0,01316%	0,01087%	0,00591%	0,00419%	0,00907%	0,00910%	0,01323%	0,00623%	0,01109%	0,01187%	0,00510%	0,01466%	0,00690%	0,01101%	0,01513%	0,00553%	0,01024%	0,01045%	0,00574%	0,00711%	0,01300%	0,01934%	0,03198%	0,00830%	0,02075%	0,01650%
TENARIS	0,00552%	0,00408%	0,01241%	0,01697%	0,01821%	0,00986%	0,01225%	0,00392%	0,00325%	0,00494%	0,01273%	0,01316%	0,00573%	0,00979%	0,00799%	0,00282%	0,01192%	0,00378%	0,01085%	0,01089%	0,00702%	0,00816%	0,01047%	0,00467%	0,02167%	0,01709%	0,01102%	0,00830%	0,03001%	0,01545%	0,01282%
UNICREDIT	0,00566%	0,01088%	0,02703%	0,05551%	0,06054%	0,02585%	0,01757%	0,00584%	0,00354%	0,01233%	0,01475%	0,02458%	0,01193%	0,02706%	0,02357%	0,00674%	0,03967%	0,00990%	0,02035%	0,03793%	0,00974%	0,01979%	0,01617%	0,00701%	0,02337%	0,02588%	0,01442%	0,02075%	0,01545%	0,07186%	0,03255%
UNIPOL	0,00701%	0,00900%	0,02313%	0,03869%	0,04022%	0,02111%	0,01667%	0,00703%	0,00545%	0,00942%	0,01186%	0,02132%	0,00953%	0,01818%	0,02101%	0,00578%	0,02596%	0,00864%	0,01790%	0,02846%	0,01008%	0,01745%	0,01425%	0,00746%	0,02146%	0,02336%	0,01235%	0,01650%	0,01282%	0,03255%	0,04390%

Table A.1: Variance-covariance matrix used both for the benchmark portfolio and the machine learning portfolio.

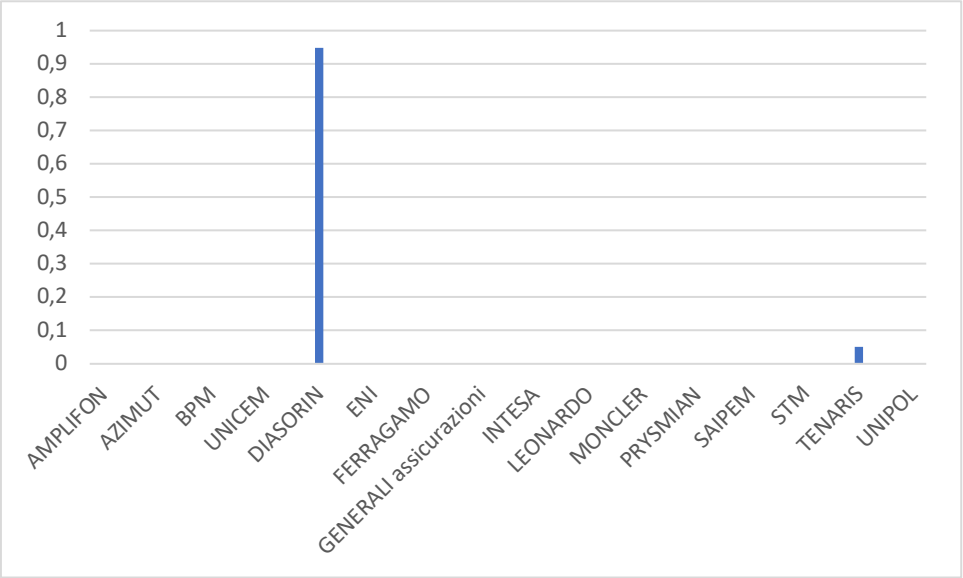

Figure A.2: graphical representation of the first machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

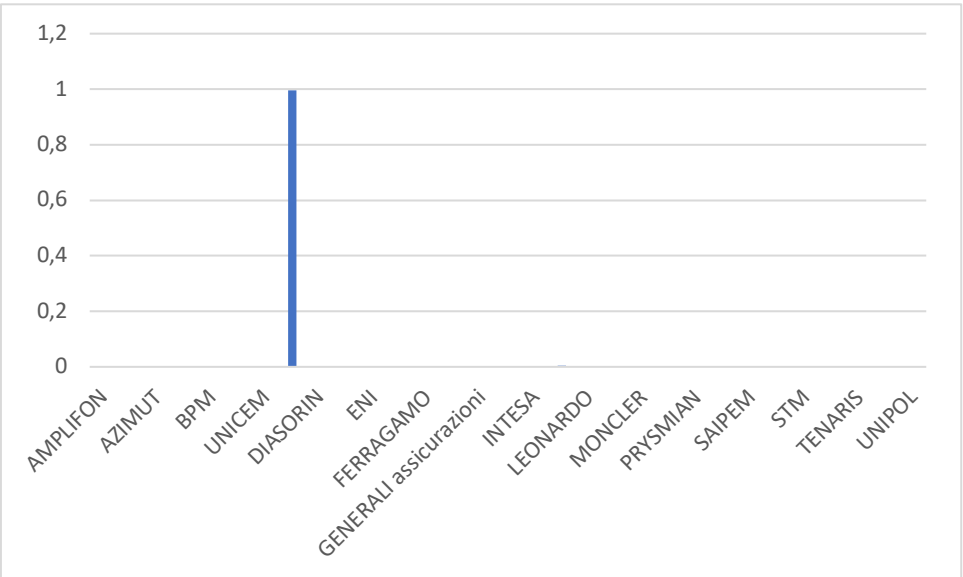

Figure A.3: graphical representation of the second machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

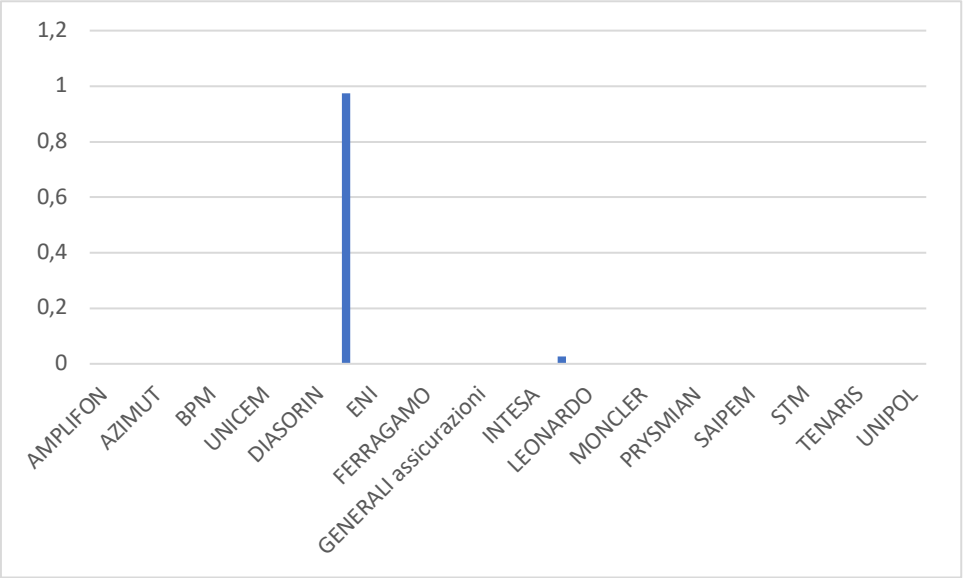

Figure A.1: graphical representation of the third machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

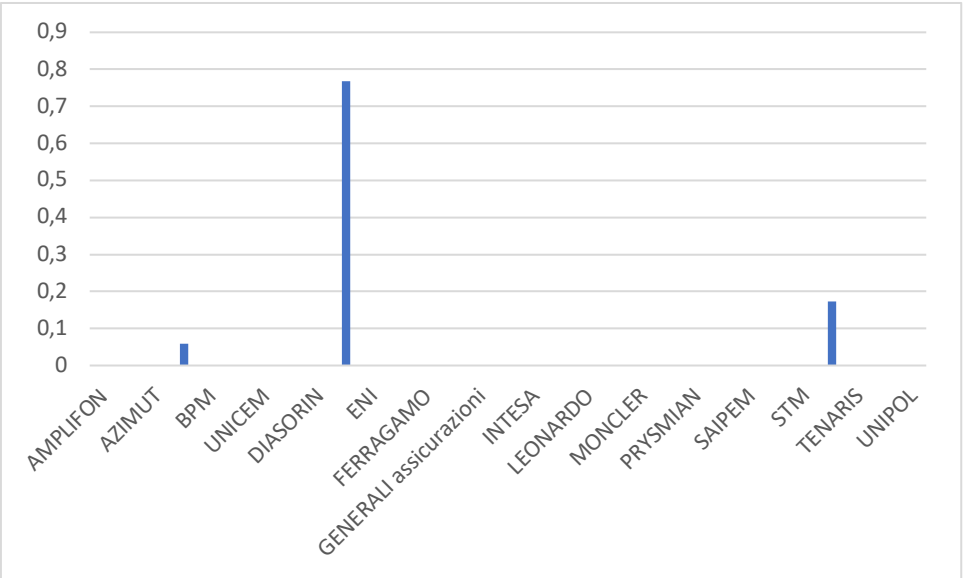

Figure A.2 graphical representation of the fourth machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

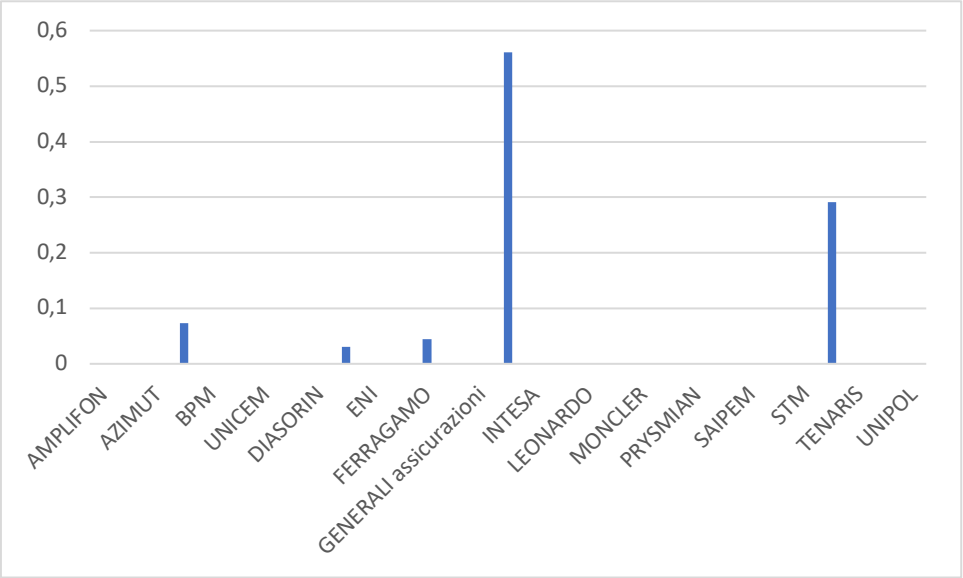

Figure A.3 graphical representation of the fifth machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

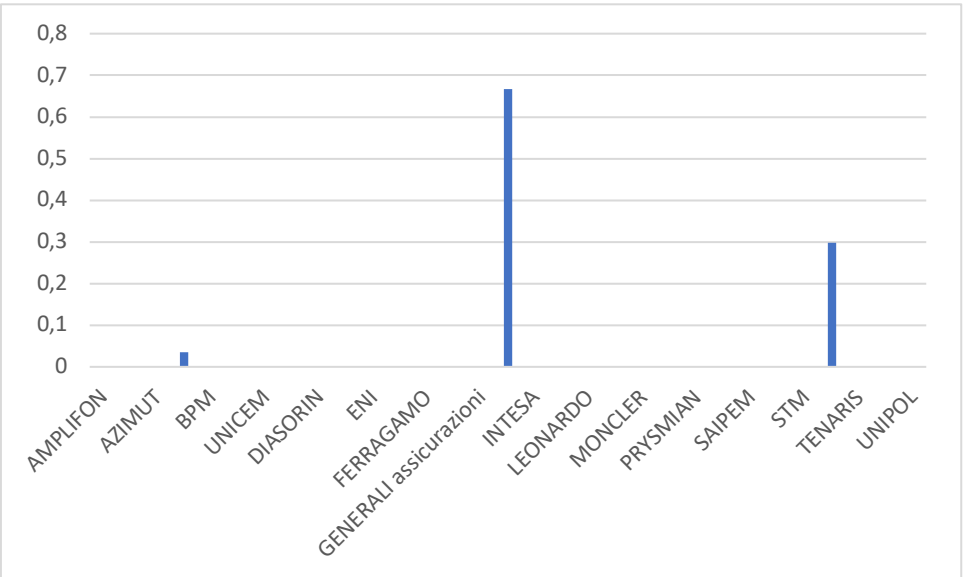

Figure A.4 graphical representation of the fifth machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

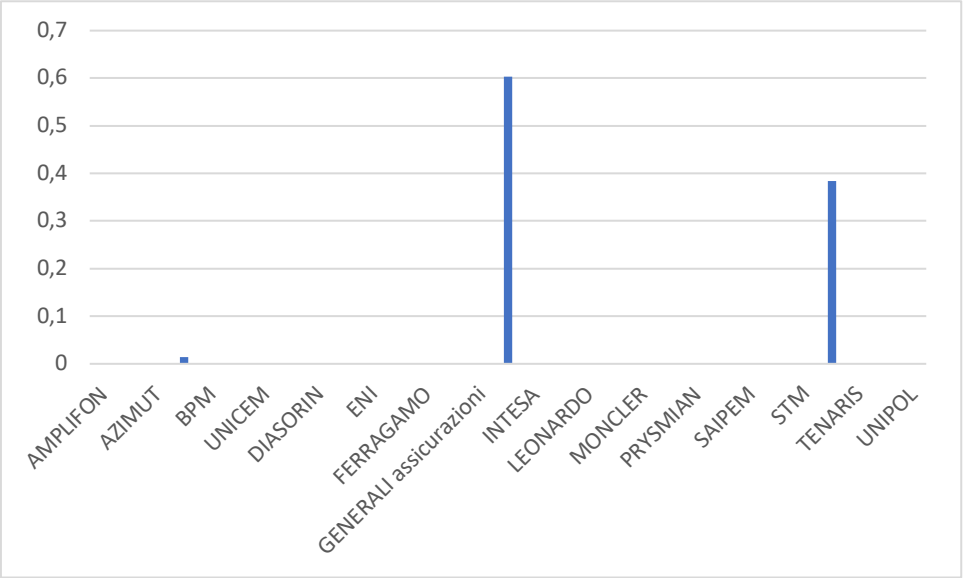

Figure A.5 graphical representation of the seventh machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

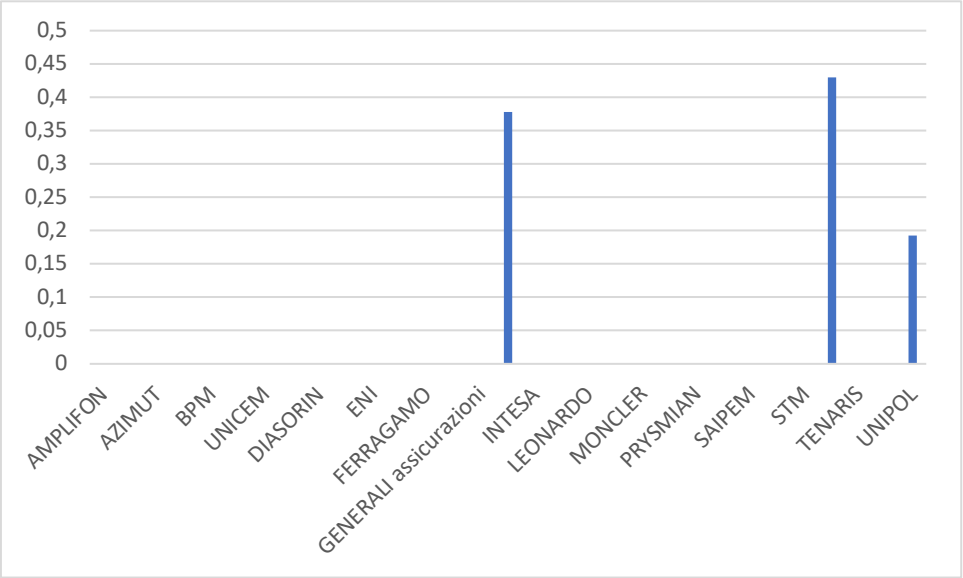

Figure A.6 graphical representation of the eighth machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

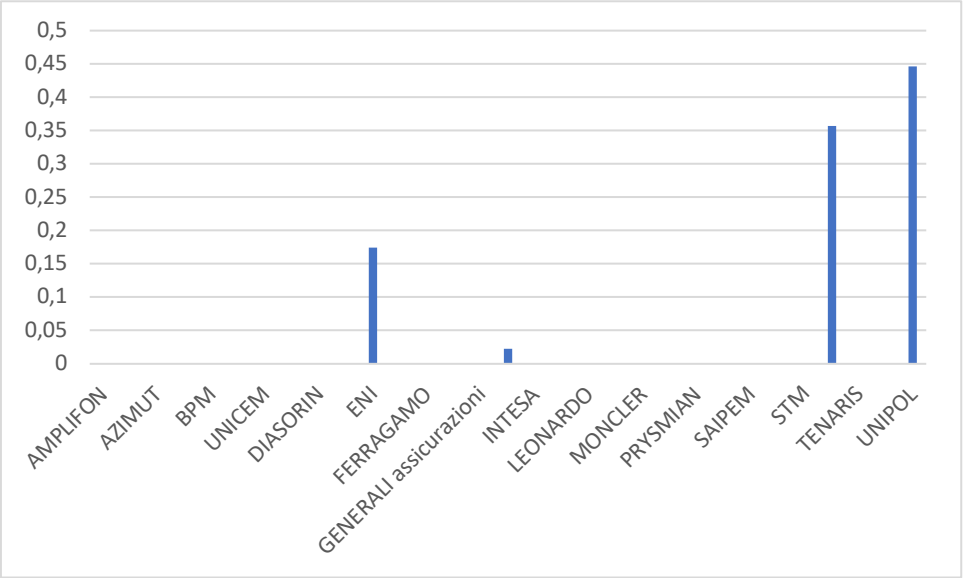

Figure A.7 graphical representation of the ninth machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

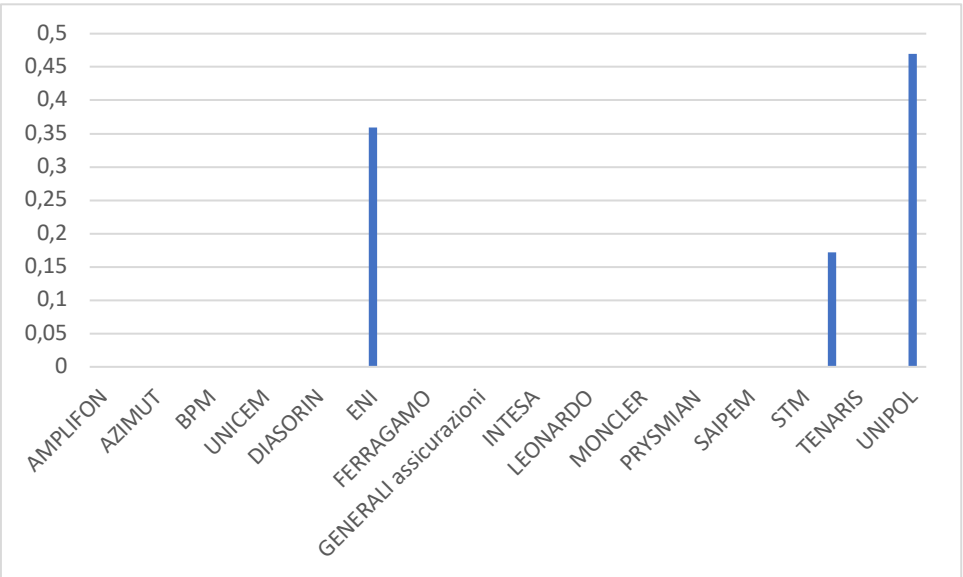

Figure A.8 graphical representation of the tenth machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

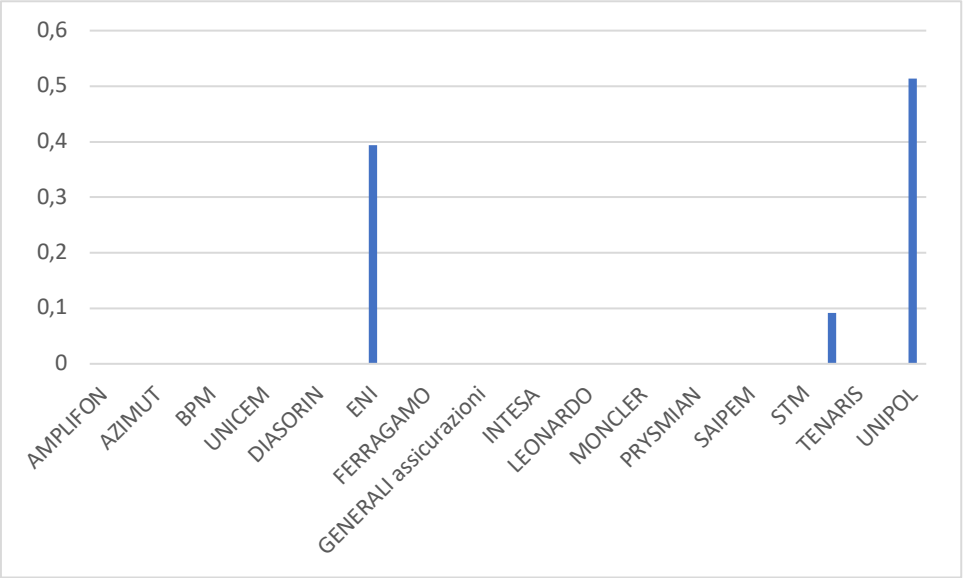

Figure A.9 graphical representation of the eleventh machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

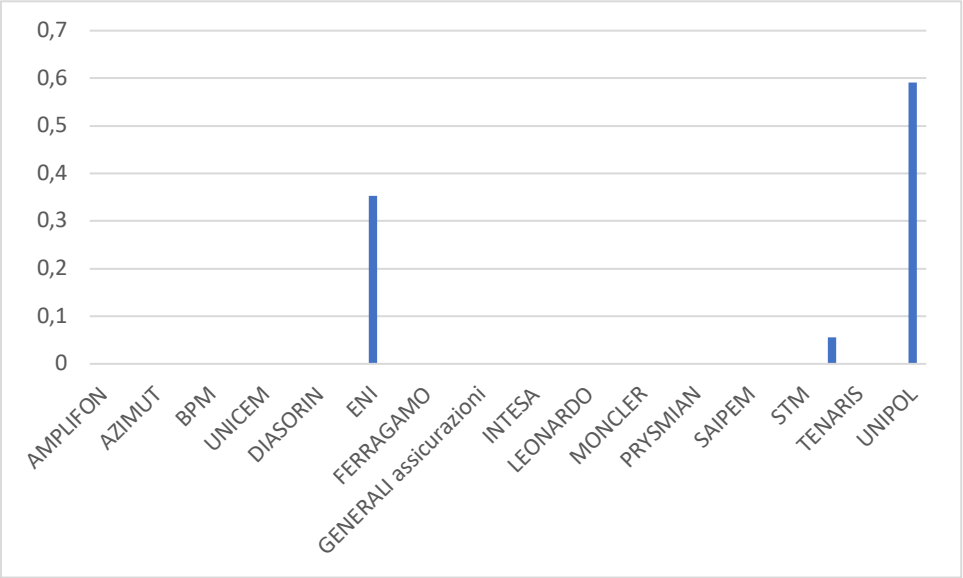

Figure A.10 graphical representation of the twelfth machine learning risky portfolio. In the horizontal axis there are the stocks, in the vertical axis the weight assigned to them.

REFERENCES

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

Carriero, A., Clark, T. E., Marcellino, M., & Mertens, E. (2022). Addressing COVID-19 Outliers in BVARs with Stochastic Volatility. *The Review of Economics and Statistics, 2022, 1-38*. https://doi.org/10.1162/rest_a_01213

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273-297. https://doi.org/10.1023/A:1022627411411

Di, X. (2014). Stock Trend Prediction with Technical Indicators using SVM, *Independent Work Report, Standford: Leland Stanford Junior University,* (2014).

Dunis, C. L., Likothanassis, S. D., Karathanasopoulos, A. S., Sermpinis, G. S., & Theofilatos, K. A. (2013). A hybrid genetic algorithm-support vector machine approach in the task of forecasting and trading. *Journal of Asset Management*, *14*(1), 52-71. https://doi.org/10.1057/jam.2013.2

Dunis, C. L., Rosillo, R., de la Fuente, D., & Pino, R. (2013). Forecasting IBEX-35 moves using support vector machines. *Neural Computing and Applications*, *23*(1), 229-236. https://doi.org/10.1007/s00521-012-0821-9

Elminejad, A., Havránek, T., & Havránková, Z. (2022). *People are less risk-averse than economists think*. Charles University in Prague, Institute of Economic Studies (IES) working paper no. 14/2022. http://hdl.handle.net/10419/265200

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2), 383-417. https://doi.org/10.2307/2325486

Fan, A., & Palaniswami, M. (2001). Stock selection using support vector machines. *International Joint Conference on Neural Networks. Proceedings,* 1793–1798 vol.3. https://doi.org/10.1109/IJCNN.2001.938434

Gilli, M., Maringer, D., & Schumann, E. (2019). Numerical methods and optimization in finance. In *Numerical Methods and Optimization in Finance*. https://doi.org/10.1016/C2017-0-01621-X

Goulet Coulombe, P., Marcellino, M., & Stevanović, D. (2021). CAN MACHINE LEARNING CATCH THE COVID-19 RECESSION? *National Institute Economic Review*, *256, 71-109*. https://doi.org/10.1017/nie.2021.10

Gupta, P., Mehlawat, M. K., & Mittal, G. (2012). Asset portfolio optimization using support vector machines and real-coded genetic algorithm. *Journal of Global Optimization*, *53*(2), 297-315. https://doi.org/10.1007/s10898-011-9692-3

Gururaj, V., & R, S. V. (2019). Stock Market Prediction using Linear Regression and Support Vector Machines. *International Journal of Applied Engineering Research* 14(8), 1931-1934.

Hargreaves, C. A., Reddy, V., & REDDY, R. V. (2017). Machine learning application in the financial markets industry. *Indian J Sci Res*, *17*(1), 253–256.

Hsu, S. H., Hsieh, J. P. A., Chih, T. C., & Hsu, K. C. (2009). A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression. *Expert Systems with Applications*, *36*(4), 7947-7951. https://doi.org/10.1016/j.eswa.2008.10.065

Hu, Z., Zhu, J., & Tse, K. (2013). Stocks market prediction using Support Vector Machine. *Proceedings of 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering,* 115-118 vol.*2*. https://doi.org/10.1109/ICIII.2013.6703096

Huang, W., Nakamori, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, *32*(10), 2513–2522.

Kara, M., Ulucan, A., & Atici, K. B. (2019). A hybrid approach for generating investor views in Black–Litterman model. *Expert Systems with Applications*, *128, 256-270*. https://doi.org/10.1016/j.eswa.2019.03.041

Karazmodeh, M., Nasiri, S., & Hashemi, S. M. (2013). Stock Price Forecasting using Support Vector Machines and Improved Particle Swarm Optimization. *Journal of Automation and Control Engineering*, *1*(2), 173-176. https://doi.org/10.12720/joace.1.2.173-176

Kim, K. J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1–2), 307-319. https://doi.org/10.1016/S0925-2312(03)00372-2

Kumar, M., & M., T. (2011a). Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest. *In Proceedings of ninth Indian institute of capital markets conference, Mumbai, India.* **https://doi.org/10.2139/ssrn.876544**

Kumar, M., & M., T. (2011b). Support Vector Machines Approach to Predict the S&P CNX NIFTY Index Returns. *In 10th Capital Markets Conference, Indian Institute of Capital Markets Paper.* https://doi.org/10.2139/ssrn.962833

Lee, M. C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, *36*(8) 10896-10904. https://doi.org/10.1016/j.eswa.2009.02.038

Lenza, M., & Primiceri, G. E. (2022). How to estimate a vector autoregression after March 2020. *Journal of Applied Econometrics*, *37*(4), 688-699. https://doi.org/10.1002/jae.2895

Ma, Y., Han, R., & Wang, W. (2021). Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, *165*. https://doi.org/10.1016/j.eswa.2020.113973

Moran, P. A., & Whittle, P. (1951). Hypothesis Testing in Time Series Analysis. *Journal of the Royal Statistical Society. Series A (General)*, *114*(4), 579-580. https://doi.org/10.2307/2981095

Pan, J., Zhuang, Y., & Fong, S. (2016). The impact of data normalization on stock market prediction: Using SVM and technical indicators. *Communications in Computer and Information Science*, *652, 72-88*. https://doi.org/10.1007/978-981-10-2777-2_7

Patil, S. S., Patidar, K., & Jain, M. (2016). Stock market trend prediction using support vector machine. *International Journal of Current Trends in Engineering & Technology*, *2*(1), 18-25.

Peachavanish, R. (2016). Stock selection and trading based on cluster analysis of trend and momentum indicators. *Lecture Notes in Engineering and Computer Science*, *1, 317-321*.

Sadia, K. H., Sharma, A., Paul, A., Padhi, S., & Sanyal, S. (2019). Stock market prediction using machine learning algorithms. *International Journal of Recent Technology and Engineering*, *8*(4), 25-31. https://doi.org/10.14201/adcaij20198497116

Sercu, P. (2009). International finance: Theory into practice. In *International Finance: Theory into Practice*.

Sharpe, W. F. (1964). CAPITAL ASSET PRICES: A THEORY OF MARKET EQUILIBRIUM UNDER CONDITIONS OF RISK. *The Journal of Finance*, *19*(3), 425-442. https://doi.org/10.1111/j.1540-6261.1964.tb02865.x

Shen, S., Jiang, H., & Zhang, T. (2012). Stock market forecasting using machine learning algorithms. *Department of Electrical Engineering, Stanford University, (2012)*.

Sims, C. A., Stock, J. H., & Watson, M. W. (1990). Inference in Linear Time Series Models with some Unit Roots. *Econometrica*, *58*(1), 113-144. https://doi.org/10.2307/2938337

Vapnik, V. N., & Chervonenkis, A. Ya. (1971). On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, *16*(2), 11-30. https://doi.org/10.1137/1116025

Wilder, J. (1978). New Concepts in Technical Trading Systems. In *New Concepts in Technical Trading Systems*.

Zbikowski, K. (2015). Using Volume Weighted Support Vector Machines with walk forward testing and feature selection for the purpose of creating stock trading strategy. *Expert Systems with Applications*, *42*(4), 1979-1805. https://doi.org/10.1016/j.eswa.2014.10.001

Zheng, A., & Jin, J. (2017). Using AI to Make Predictions on Stock Market. *Thesis, Stanford University (2017)*.