Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Integrating Machine Learning and Portfolio Optimization Methods for Enhancing Baltic Stock Market Portfolio Composition Analysis

Masters's Final Degree Project

Mantas Kasimovas

Project author

Prof. dr. Audrius Kabašinskas

Supervisor

Assoc. prof. dr. Lina Sinevičienė

Supervisor

Kaunas, 2024

Kaunas University of Technology
Faculty of Mathematics and Natural Sciences

Integrating Machine Learning and Portfolio Optimization Methods for Enhancing Baltic Stock Market Portfolio Composition Analysis

Masters's Final Degree Project

Business Big Data Analytics (6213AX001)

Mantas Kasimovas
Project author

Prof. dr. Audrius Kabašinskas
Supervisor

Assoc. prof. dr. Lina Sinevičienė
Supervisor

Assoc. prof. dr. Mindaugas Šnipas
Reviewer

Assoc. prof. dr. Arvydas Jadevičius
Reviewer

Kaunas, 2024

Kaunas University of Technology
Faculty of Mathematics and Natural Sciences
Mantas Kasimovas

Integrating Machine Learning and Portfolio Optimization Methods for Enhancing Baltic Stock Market Portfolio Composition Analysis

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarized from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Mantas Kasimovas

Confirmed electronically

Kasimovas, Mantas. Integrating Machine Learning and Portfolio Optimization Methods for Enhancing Baltic Stock Market Portfolio Composition Analysis. Master's Final Degree Project / supervisors prof. dr. Audrius Kabašinskas, assoc. prof. dr. Lina Sinevičienė; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Mathematics, Applied mathematics.

Keywords: portfolio optimization, stock price prediction, machine learning, random forest, ARIMA, Long Short-Term Memory, transformers.
Kaunas, 2024. 45p.

Summary

The effectiveness of combining machine learning methods with Modern Portfolio Theory (MPT) to forecast stock prices and enhance investment portfolios is investigated in this thesis. In particular, it looks into how well the Random Forest, ARIMA, LSTM, and Transformers machine learning models perform when it comes to predicting the stock prices of 20 components of the OMXBBGI index. In accordance with the forecasts, the thesis builds a portfolio with the goal of maximizing returns for a specific degree of risk using Markowitz's Modern Portfolio Theory. Using historical stock data, the machine learning models were trained and verified. Based on these forecasts, a portfolio was subsequently constructed using MPT principles, with an emphasis on maximizing the risk-return trade-off. Important financial measures including Jensen's Alpha, Value at Risk (VaR), and the Sharpe Ratio were used to evaluate the performance of the portfolio. The results show that portfolios built using machine learning models' forecasts performed better than the benchmark index, offering higher returns for the same amount of risk. This enhancement demonstrates how machine learning may be used to improve investment strategies and stock market analytics. By demonstrating the useful advantages of fusing analytical methods with traditional financial theories to enhance risk management and investment choices, the thesis adds to the body of knowledge in the field of finance.

Kasimovas, Mantas. Mašininio mokymosi ir portfelio optimizavimo metodų naudojimas sudarant Baltijos šalių vertybinių popierių portfelį. Magistro baigiamasis projektas / vadovai prof. dr. Audrius Kabašinskas, assoc. prof. dr. Lina Sinevičienė; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Matematikos mokslai, taikomoji matematika.

Reikšminiai žodžiai: portfelio optimizavimas, akcijų kainų prognozavimas, mašininis mokymasis, atsitiktiniai miškai, ARIMA, LSTM, dėmesio modeliai.

Kaunas, 2024. 45p.

Santrauka

Šiame darbe tiriamas mašininio mokymosi metodų derinimo su Moderniosios portfelio teorija (MPT) efektyvumas prognozuojant akcijų kainas ir sudarant investicinį portfelį. Nagrinėjama, kaip atsitiktinių miškų, ARIMA, LSTM ir Transformers mašininio mokymosi modeliai pasirodo prognozuojant 20 OMXBBGI indekso komponentų akcijų kainas. Atsižvelgiant į prognozes, darbe konstruojamas portfelis, siekiant maksimalių grąžų esant konkrečiam rizikos lygiui, naudojant Markowitz Moderniosios portfelio teorijos principus. Mašininio mokymosi modeliai buvo mokomi ir patikrinti naudojant istorinius akcijų duomenis. Remiantis šiomis prognozėmis, vėliau buvo sukonstruotas portfelis pagal MPT principus, pabrėžiant rizikos ir grąžos santykio maksimizavimą. Portfelio veiklos vertinimui buvo naudojami svarbūs finansiniai rodikliai, tokie kaip Jensen Alpha, Rizikos vertė (VaR) ir Sharpe santykis. Rezultatai rodo, kad portfelis, sukurtas naudojant mašininio mokymosi modelių prognozes, pasirodė geriau nei etaloninis indeksas, siūlydamas didesnę grąžą esant tokiam pačiam rizikos lygiui. Šis pagerinimas parodo, kaip mašininis mokymasis gali būti naudojamas gerinant investicijų strategijas ir akcijų rinkos analizę. Demonstruodamas praktiškus analitinių metodų ir tradicinių finansų teorijų sujungimo privalumus gerinant rizikos valdymą ir investicinius sprendimus, darbas prisideda prie finansų srities žinių bazės plėtros.

Table of contents

List of figures

List of tables

List of abbreviations and terms

Abbreviations:

ANN – Artificial Neural Network

ARIMA – Autoregressive Integrated Moving Average

CAPM – Capital Asset Pricing Model

DT – Decision Trees

ETF – Exchange Traded Fund

GA – Genetic Algorithm

LSTM – Long Short-Term Memory

MA – Moving Average

MAE – Mean Absolute Error

MAPE – Mean Absolute Percentage Error

MAR – Minimum Acceptable Return

ML – Machine Learning

MPT – Modern Portfolio Theory

MSE – Mean Squared Error

MVO – Mean Variance Optimization

PMPT – Post Modern Portfolio Theory

RF – Random Forest

RMSE – Rooted Mean Squared Error

RNN – Recurrent Neural Network

ROI – Return on Investment

SVM – Support Vector Machines

VaR – Value at Risk

Introduction

The investment landscape has seen a substantial upheaval in recent years. The financial markets are now more accessible to a wider range of people because to the growth of digital platforms with the likes of Robinhood, Stash, M1 Finance, Public, Webull, majority of whom were founded in the past 10 years. Additionally, close to zero transaction costs have made investing more appealing to the masses. According to Chang (2019), many broker firms including Interactive Brokers, Charles Schwab, TD Ameritrade, E-Trade, Ally Invest, and Fidelity have offered to their clients zero-commission and ETF trading or zero fees across all stock trading back in 2019. Retail investment has increased dramatically as a result of this phenomenon, with people actively looking for ways to manage their portfolios. Over six million trading apps were downloaded in the US in January 2021, setting new records for average daily volumes of stock and options trades at retail brokerages (*The Rise of Retail Investing | United Fintech*, 2021). However, predicting stock prices accurately is one of the main issues in portfolio management. The intricacies of the market dynamics are frequently too complicated for traditional tools to fully capture. In response, machine learning algorithms have gained popularity among researchers and practitioners as a means of making predictions that are both more accurate and efficient.

The purpose of this master's thesis is to investigate how well different machine learning models predict stock prices and how to use those predictions to build the best possible portfolio. The research specifically focuses on four different algorithms: Random Forest, ARIMA (autoregressive integrated moving average), LSTM (long short-term memory), and LSTM with attention mechanism.

The increasing interest in applying machine learning techniques to improve investing strategies is the driving force behind this research. Through the effective use of ML algorithms, investors want to maximize returns while minimizing risks. The newly created portfolio's performance will be assessed against the 20-member OMXBBGI index, which acts as a benchmark. Index was chosen in order to hopefully boost additional interest towards investing as the number of retail investors is still small in Lithuania. In 2022 there were 53.8 thousands of retail investors in Lithuania, or 1.9% from total population (*Investuotojo Portretas: Aktyviausiai Investuoja 33-44 Metų Žmonės*, 2023). By providing research on how to employ machine learning models in creating optimal portfolio it is hoped that the number of retail investors would grow faster in Lithuania.

Objectives:

1. Conduct a comprehensive review of literature on portfolio optimization methods, highlighting both traditional and modern approaches.
2. Explore existing literature on machine learning algorithms for stock price prediction, emphasizing their strengths and limitations.
3. Compare the practical performance of Random Forest, ARIMA, LSTM, and LSTM + Transformers models in predicting stock prices.
4. Utilize the predicted stock prices to construct an optimal portfolio comprising 20 constituents of the OMXBBGI index.
5. Evaluate the performance of the newly created portfolio using portfolio efficiency indicators such as Sharpe ratio, Jensen's alpha, and Treynor ratio.

6. Discuss the feasibility and practical implications of implementing the proposed portfolio strategy in real-world investment scenarios.

Research Question:

The central research question driving this thesis is: Can an optimal portfolio created using machine learning algorithms for stock price predictions surpass the OMXBBGI index in terms of generating greater returns with the same level of risk?

This study has important ramifications for academics and professionals working in the financial and investment fields. Through establishing a connection between portfolio management and machine learning approaches, the research hopes to offer insightful information on how algorithmic trading tactics might improve investment results.

This is how the rest of the thesis is structured: Chapter 1 reviews the literature on machine learning algorithms for stock price prediction, and offers an analysis of the literature on portfolio optimization techniques. The methods used to compare and assess the performance of the chosen machine learning models are presented in Chapter 2. The results of the empirical investigation and their implications for portfolio construction are covered in Chapter 3. Chapter 4 brings the thesis to a close by reviewing the main conclusions, outlining the limits, and outlining potential directions for further study.

1. Literature Review.

Over the years, a great deal of research on portfolio optimization has been and continues to be published. Even though finance is not a brand-new discipline, new methods and strategies are constantly being developed to identify the ideal portfolio. Therefore, in order to present new research findings and angles, it would be helpful for this work to analyze and evaluate many articles and papers published on the issue of portfolio optimization in order to shed light on the lengths and depths that have not yet been explored in the academic community.

1.1. Modern Portfolio Theory (MPT)

The idea of portfolio optimization is older than 70 years when Harry Markowitz first introduced it to the world. The essential idea of optimal portfolio lied in the trade-off between risk and return – greater the risk, the more possibly rewarding return (Markowitz, 1952). Markowitz in his paper described a mean-variance model which outlines the best financial assets' combination to achieve the lowest risk possible for the desired rate of return. At the time the trade-off concept between risk and return was groundbreaking for two reasons (Kolm et al., 2014). On one hand, it suggested the possibility of assessing both the return and risk of a portfolio through a quantitative approach that takes into account the returns of individual securities as well as their movements in relation to each other. On the other hand, the theory presented a problem formulation called mean-variance optimization (MVO) which stated that from an infinite number of portfolios that can achieve a specific return goal, the optimal choice for the investor is the portfolio with the least amount of variance (Markowitz, 1952). Any other portfolios are deemed incompetent due to their higher variance, resulting in increased risk. The portfolio optimization idea made investors realize how diversification in combination with adequate risk management could in turn lead to safer and more profitable investment choices.

However, the MPT had a few imperfections at first as one of its biggest drawbacks was portfolio assessment fixated on variance rather than downside risk (Team, 2023). According to the MPT, two portfolios with the same level of variance and return would be equally regarded by investors. Yet both portfolios might have the same variance by one having periodic tiny declines in its value while the other might only rarely experience significant plummets in its worth. Since investors are believed to be risk averse, they will always prefer the one with frequent but small losses to the one that poses a risk to lose double digits in one day. Another shortcoming the MTP did not consider then was transactional costs (Lydenberg, 2016). Investors experience costs when they are buying or selling stocks, bonds, options, futures or forwards or any other financial instruments. Those costs can be bid ask spread, exchange fees, brokerage fees or regulatory fees, taxes. If an investor is buying or selling infrequently, costs will not amount to anything meaningful or impact the investment strategy significantly. However, if a trader performs consistent trades trying to profit from short-term market tendencies or public speculations it will undoubtedly mean more frequent trades and in turn larger transactional costs incurred. Although portfolio optimization may be done frequently and weights of the constituents adjusted periodically, the whole idea of portfolio holding rather than separate stocks lies in the long-term investment strategy. It means that people who are interested in holding a portfolio of stocks will hold it for a very long time and only recalibrate the weights to adjust for increased risk. Nonetheless, even holding only one portfolio could mean holding at least 10, 50 or 500 different stocks depending on one's investment strategy. Even if an investor holding a portfolio of 500 stocks was recalibrating the portfolio every year, it would mean buying or selling quite a few of the constituents. This could make a meaningful impact on the investment strategy and make investor

reconsider of the idea of building their own portfolio instead of simply buying regularly shares of ETF that tracks S&P 500.

One more unrealistic assumption that is built in MPT is unconstrained market liquidity (Lydenberg, 2016). This might hold in a highly developed market like NYSE or NASDAQ in the US, however in smaller markets there might be considerable liquidity constraints. However, in smaller markets like Baltics it might be hard to find a buyer to liquidate your position at a reasonable price when the market turmoil starts. Hence when constructing a portfolio in less liquid markets one must consider their abilities to liquidate the position in the unfortunate time of events.

Furthermore, Green and Hollifield (1992) added that not necessarily but rather contrary – the negative relationship between mean-variance portfolios and high diversification factor exists pointing out that portfolios which are well optimized based on the MVO approach are poorly diversified. Hence, was proposed modern portfolio theory (MPT) which expands on MVO by stressing diversification as a means of lowering portfolio risk. It proposes that investors might get a more efficient frontier, or better risk-return trade-offs, by combining assets with low or negative correlations. Although there are challenges that modern portfolio theory faces, there are also additional developments which help to improve the model (Kolm et al., 2014). In their paper authors describe 4 extensions:

- Taking into account expenses related to transactions (such as costs associated with market impact) and the impact of taxes.

- Incorporating diverse constraints that consider particular investment guidelines and institutional characteristics.

- Using Bayesian techniques, stochastic optimization, or robust optimization methods to model and quantify the influence of estimation errors in risk and return forecasts on portfolios.

- Expanding the Mean-Variance Optimization (MVO) framework over multiple periods to encompass intertemporal factors like hedging requirements, evolving market conditions, market impact expenses, and the decay of alpha.

Though overall the MPT on its own was a groundbreaking theory then, there has been quite a few improvements to be made before it could be used as a robust portfolio optimization strategy.

1.2. Post Modern Portfolio Theory (PMPT)

To improve the MPT and consider the criticism it has faced over the years a more progressive approach was invented at Pension Research Institute in the USA (Todoni, 2015). Before the PMPT, it was believed that returns on financial assets followed a normal distribution, that variance and standard deviation were trustworthy risk indicators, and that investors had uniform expectations. The necessity to construct PMPT as an extended return-risk paradigm has been imposed by the incompatibility of these assumptions with market realities.

Every investor has a separate minimum acceptable return (MAR), which they establish as a target, according to PMPT. The rate of return an investor should make in order to prevent a negative outcome is known as the investor's target rate of return, or MAR. When assessing the outcomes, MAR acts as the investor's own benchmark. Compared to MPT, PMPT is more individually tailored to each investor due to the MAR selection option. Risk is defined by the MPT as the total return volatility

around the mean value, and it may be quantified using either the standard deviation of return or variance. All uncertainties are handled equally by MPT: deviations above and below the mean value are handled similarly. PMPT asserts that investment risk should be connected to each investor's unique objective, and that returns above this target do not constitute an economic or financial risk, in contrast to MPT, which links risk with obtaining an average return. Only volatility below the investor's intended return is deemed risk, according to PMPT. Uncertainty created by a return over the target is nothing more than a risk-free opportunity to obtain an unexpectedly big return (Rom and Ferguson, 1993).

Moreover, variance and standard deviation are inappropriate risk measures, according to PMPT. This is because those two metrics do not mirror the reality of the market and are not good risk indicators. The explanation for this stems from the assumption that all deviations from the mean are valued the same, the negative and the positive. Additionally, the assumption states that standard deviation is symmetrical to risk, though in reality this does not hold true as investors always seek their return to be greater than the average return (Dumbliauskienė and Paužuolis, 2015). Hence, PMPT's risk is calculated using downside risk instead of standard deviation or variance.

1.3. Black-Litterman Model

MPT idea is taken one step forward by Black and Litterman who stated that mean-variance optimization has the potential to generate disproportionate or counterintuitive allocations for certain assets within the portfolio (Black and Litterman, 1991; Black and Litterman, 1992). The concept underlying Black and Litterman's (BL) model is the estimation and forecasting of the asset parameters through the integration of two information sources: the past performance of the asset returns and expert forecasts that are subjective in nature (Stoilov et al., 2020). As a result, subjective expert opinions have an impact on the covariance matrix and the portfolio parameters for mean returns. When objective perspectives and actual history are combined, portfolios not having a clear separation of asset classes are created as is the case with Markowitz portfolios. Because multiple points of view will result in distinct portfolio solutions that lack common ground for comparison, the BL model is criticized for its perceived lack of efficiency. The incapacity of the Black-Litterman model to guarantee portfolio optimization is another of its limitations (Kenton, 2023). Rather, it creates portfolios and modifies them based on the opinions of investors or portfolio managers about the market. Its dependence on assumptions, which leaves it vulnerable to the subtleties of investor opinions, is another significant disadvantage. This suggests that the model works on the premise that various points of view act independently of one another and have an uneven impact on the model's output. Nonetheless, because of the increased degree of diversification it provides, investors find the portfolio structure provided by the BL model to be more acceptable than the one suggested by MPT.

1.4. Risk Parity Portfolio Optimization

Another approach to create an optimal portfolio is using risk parity – a strategy that aligns with modern portfolio theory by making an effort to optimize returns while continuously monitoring the investor's risk tolerance. On the other hand, the risk parity strategy permits leverage and short selling (Chen, 2022). Risk parity is a sophisticated portfolio method that is frequently employed by professional investors and hedge funds. Its allocations are more sophisticated than those of simpler allocation systems since they necessitate a sophisticated quantitative methodology. To achieve the best possible return at the desired risk level is the aim of risk parity investing (Chen, 2022). Risk

parity techniques provide the utilization of leverage and alternative diversification in addition to enabling short selling within funds and portfolios. Portfolio managers are free to utilize any combination of assets while using this strategy. However, risk parity strategies base their investment decisions on the optimal risk target level rather than creating allocations to various asset classes to arrive at an optimal risk target. Leverage is frequently used to accomplish this goal by distributing risk evenly across various asset classes while maintaining the ideal risk target level. When using leverage in a risk parity approach, assets must be regularly rebalanced (Edwards, 2022). It could be necessary to balance out the leveraged investments to maintain the amount of volatility exposure for each asset type. Derivatives may be used in risk parity strategies; thus, active management is necessary for these positions. Another important concept when creating a risk parity portfolio to consider is correlation (Edwards, 2022. In general, it is challenging to find perfect positive and negative correlations in finance. Nevertheless, adding assets to a portfolio that are negatively correlated with one another increases its diversification. There is no assurance that the relationships that have been calculated based on previous data will hold true going forward. One of the primary objections to risk parity and contemporary portfolio theory is this.

1.5. Review of Machine Learning Techniques in Finance

1.5.1. Random forest

Leo Breiman and Adele Cutler are the trademark holders of the popular machine learning technique known as "random forest," which aggregates the output of several decision trees to produce a single outcome. Since random forest algorithm is based on the ensemble of decision trees (DT), it would be beneficial to shortly remember what DTs are. Plainly, decision trees can be imagined as branches (hence the name) of questions for which answers are leading to more questions, therefore more branches and this “tree” develops and grows branches until the final – leaf – node is reached. The main usage of such method is to determine the optimal split to limit the data, and DTs are usually trained using the Classification and Regression Tree (CART) algorithm. Despite being a popular supervised learning algorithm, decision trees have many drawbacks, including bias and overfitting (*What Is Random Forest? | IBM*, n.d.). However, if enough of decision trees are used as an ensemble in the RT algorithm, the analysis could yield more precise and accurate predictions, especially when individual trees are zero-intercorrelated.

Node size, the number of trees, and the number of features sampled conclude the essential three hypermeters used in the RF algorithm that will need to be determined before the training phase. Then each tree in a RT ensemble is made from a bootstrap sample, or data sample, which is taken from a training set with replacement. A third of that training sample – referred to as the out-of-bag sample – is reserved for testing purposes. Feature bagging is then used to introduce yet another randomization, increasing dataset variety and decreasing decision tree correlation. The individual decision trees in a regression job will be averaged, and in a classification work, the predicted class will be determined by a majority vote, or the most common categorical variable. Lastly, that prediction is confirmed using cross-validation using the out-of-bag sample.

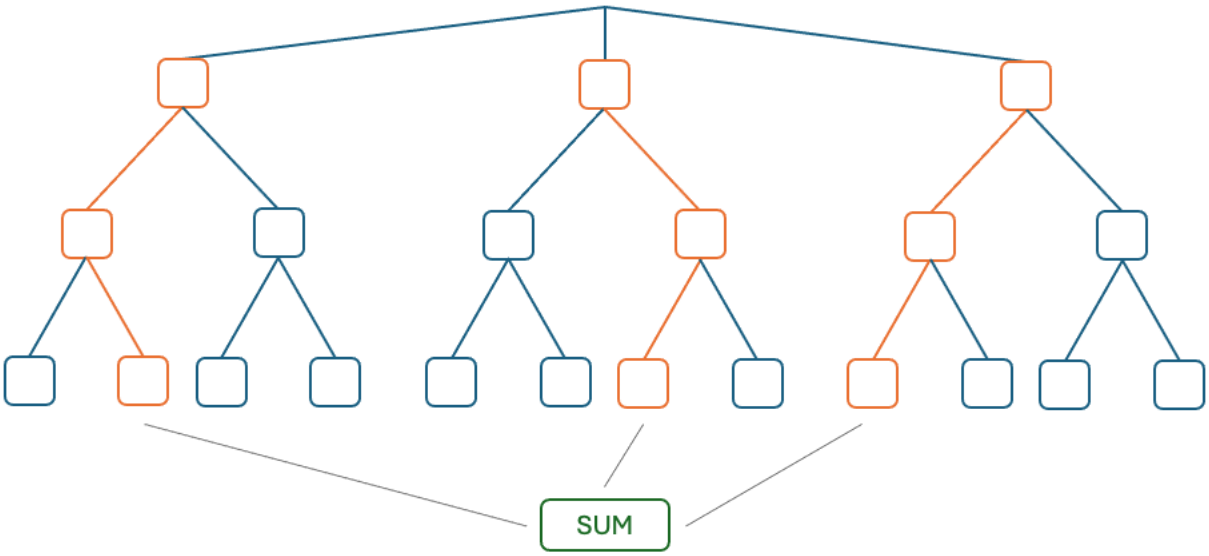


Figure 1. Decision tree diagram

Since the RT algorithm is great for handling non-linear data it has been also applied in stock price prediction experiments. For instance, Vijh et al. (2020) employed RF to predict closing prices of JP Morgan, Nike, Johnson and Johnson, Goldman Sachs and Pfizer. They used RMSE, MAPE and MBE indicators to compare the prediction results versus the result presented by an alternative – Artificial Neural Network (ANN). Even though, the outcome of comparison favored the ANN model, the MAPE values for all five stock predictions using the RF method varied from 0.75% to 1.14% which according to Moreno et al. (2013) means highly accurate forecasting since MAPE value is below 10%.

1.5.2. Autoregressive integrated moving average (ARIMA)

A statistical analysis technique called autoregressive integrated moving average, or ARIMA, makes use of time series data to forecast future trends or to get a deeper understanding of the data set. ARIMA is a type of regression analysis that assesses the strength of a single dependent variable in relation to other changing variables (Hayes, 2024). Instead of using actual values, the model looks at discrepancies between values in the series to forecast future movements in securities or the financial markets. An ARIMA model is composed of three parts: autoregression (AR), integrated (I) and moving average (MA). AR section of the model describes how an observation and a predetermined number of lagged observations – that is, earlier values – relate. Integrated part is used for differencing the time series data one or more times. It is needed to achieve stationarity – the constancy of statistical features like mean and variance throughout time. The last piece of the equation is responsible for modeling the error term that is represented as a linear mixture of error terms that happened both simultaneously and at different points in the past.

As ARIMA is adopted to predict time series data, it is a potentially useful tool for stock price predictions, as stock prices are indeed time series data. Funde and Damani (2023) employed ARIMA model to forecast stock prices of 15 selected constituents from NIFTY 50 index for the period 2016 April 1 to 2021 March 31. The authors applied historical price data of 1239 observations to predict the price of 1240th day and used up to date data to retrain and forecast the next day-ahead price for the consecutive 38 days. The model's results were estimated using RSME and MAPE metrics. The latter value ranged from 11.53% to 29.31% depending on the stock for which the price was being

forecasted. Following Moreno et al. (2013) methodology on forecasting accuracy, the results provided by the authors would fall under "good" (MAPE between 10 and 20 percent) and "reasonable" (MAPE between 20 and 50 percent) predictions. Ma (2020) in their article analyzed ARIMA's potential in predicting stock prices in comparison to ANN and Long Short-Term Memory (LSTM) network models. The author used ARIMA (1, 0, 0) model to arrive at a conclusion that ANN model was superior than that of the ARIMA, and LSTM a better consideration than ANN. Adebiyi et al. (2014) also performed a comparison analysis of ARIMA and ANN models for stock price prediction using Dell stock index. The authors as well used ARIMA (1, 0, 0) model for stock price prediction as it was concluded to be the best fit after a few trials. Number 1 in the ARIMA model means that the best predictor of the series is its immediate past value, adjusted by a constant and a stochastic error term. Employing forecasting error formula to evaluate the accuracy of predictions, they concluded that ARIMA model performed quite well as the forecast error was very small. Another study by Almaafi et al. (2023) assessed and compared effectiveness of the ARIMA and XGBoost models in forecasting the weekly closing stock prices of Saudi Telecom Company. They used ARIMA (0, 1, 1) model as it showed to have the lowest AIC score. Nonetheless, the ARIMA model proved to be inferior to XGBoost method as it was providing more linear results, while the latter method was capable of picking up the downward trend better. The authors also used MAE, RSME and MAPE scores to compare the accuracy of model's prediction capabilities. Even though, the ARIMA model was slightly worse than XGBoost, it still managed to achieve 2.1% MAPE which is considered to be a sign of a very accurate forecasting model.

ARIMA models can be very useful in forecasting time-series data, however there are some points to consider when implementing them. Though models are trustworthy when it comes to predicting short-term outcomes, they lack stability when forecasting longer periods (Hayes, 2024). Additionally, since ARIMA is a linear model, it struggles to capture shocks in predicted values, though it's great at picking up on seasonality and trends (Grogan, 2021; Petrică et al., 2016).

1.5.3. Long Short-Term Memory (LSTM) Networks

A popular recurrent neural network (RNN) architecture in deep learning is called LSTM (Long Short-Term Memory). It was created by Hochreiter and Schmidhuber to address the issue brought about by conventional RNNs and machine learning methods. An input layer, one or more hidden layers, and an output layer make up LSTM networks. The number of explanatory variables and the number of neurons in the input layer are the same. The primary feature of LSTM networks resides in the buried layer or layers that comprise so-called memory cells (Fischer & Krauß, 2018). Those cells consist of three parts: the forget gate, the input gate and the output gate (see Figure 1). The first part is responsible for determining whether the information passed on from the previous timestamp is important or has to be dismissed. In the middle part, the cell attempts to learn new information from the input to this cell. Finally, the cell transfers the changed data from the current timestamp to the subsequent timestamp in the third section. This whole cycle of LSTM can be treated as a single-time step (Saxena, 2024).

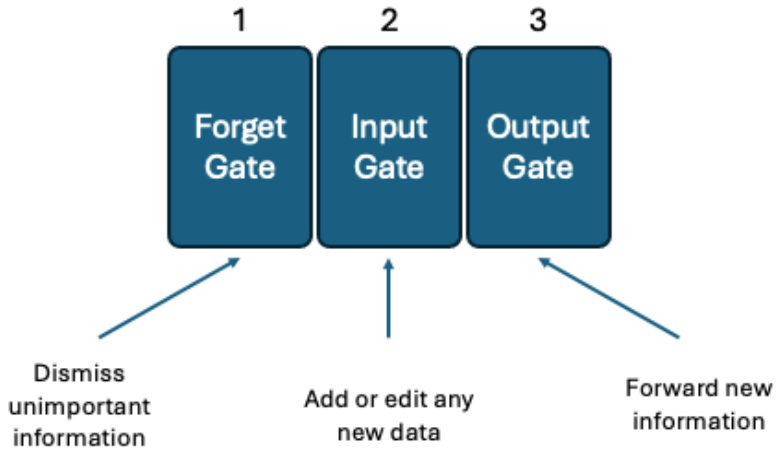


Figure 2. Simplified architecture of LSTM network's cell.

Because LSTM has feedback connections, as opposed to standard neural networks, it can handle complete data sequences as opposed to simply single data points. Since LSTM can extract meaningful insights from sequential data, it has developed into a potent tool in deep learning and artificial intelligence that is facilitating advances across a wide range of domains. One of those domains is finance and stock price predictions. Chung and Shin (2018) employed genetic algorithms (GA) to calculate the LSTM network's temporal window size and architectural components. To validate the approach, they took daily Korea Stock Price Index data. They found that the best 1-day ahead predictions can be obtained by using past 10 trading days. The same outcome was reported by Chen et al. (2021) who were trying 5-, 10-, 15- and 20-day time stamps to conclude that the lowest errors were calculated when they used 10 prior days' information to predict the 11th day's price. Furthermore, Chung and Shin (2018) have determined that the optimal number of LSTM units – that is, the number that makes up two hidden layers—is 15 and 7, respectively. Finally, according to the performance measures, for which they used mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE), the new hybrid model proved to be superior to the benchmark. In another article by Liu et al. (2021), authors experimented by combining long short-term memory model with online social networks to predict the close prices of the SSE 50 constituent stocks. They used two layers of neural networks with 16 neurons in each layer. Batch size was chosen to be 32 and time window was set to be 5 days which means that past 5 days were used to predict the price of the next day. Authors used 100, 200 and 300 epochs for model training and learning rate of 0.001. The model results were validated using RMSE and MAPE indicators, both of which showed a trend that the more learning epochs there were, the better RMSE and MAPE results were obtained. One more paper by Zheng et al. (2021) proposed the use of evolutional bidirectional LSTM model for price prediction of 3 indexes: Shanghai Securities Composite Index, A-Share Index and S&P 500 Index. They used the past 20 days' closing prices to predict the next day's price with learning rate of 0.00001 and 150 epochs. The outcome of their analysis measured by MAE, MAPE and MSE indicators, all of which supported the superiority of their proposed method. In addition to already reviewed articles, (Fischer & Krauß, 2018) also used LSTM for financial market predictions. Namely, they used the method to predict price changes for the S&P 500's component stocks between 1992 and 2015. They used an input layer with 1 feature and 240 timesteps, an LSTM layer with 25 neurons, and a dropout rate of 0.1. The output layer consisted of 2 neurons with a "softmax" activation function. The authors employed their LSTM model and evaluated it against random forest, deep neural network, and logistic regression to find out that regardless of the size of the portfolio, LSTM

demonstrated better performance indicators than other methods. For instance, the daily returns for LSTM model reached 0.46 percent, when for random forest, deep neural network, and logistic regression they stretched to 0.43, 0.32 and 0.26 percent respectively. Nonetheless, Fischer and Krauß (2018) provided some criticism for practical implementation of LSTM over time. While LSTMs initially brought strong returns, their edge diminished over time as their techniques became more widespread and understood within the trading industry. This diffusion among professionals gradually eroded profitability, illustrating a common trend in financial markets where novel strategies can give an advantage initially, but tend to lose effectiveness as they become mainstream. The RAF's ability to capitalize on specific market conditions (like the 2008 financial crisis) highlights the importance of contextual factors in algorithmic trading performance.

1.5.4. Transformers

Back in 2017, the Google team proposed transformers which is a classic natural language processing (NLP) model that is believed to be better in every aspect compared to RNNs and CNNs. Model's novelty lies in attention mechanism that enables parallelization and global information in the model, instead of employing the RNN sequential structure (Wang, 2023). In contrast to recurrent networks, the transformer may access any point in the past, independent of the distance between words, and does not experience gradient disappearance. There is a decoder section and an encoder section in a transformer. The latter section has a stack of encoders. In order to produce the desired output, it encodes the input data in accordance with a certain mode, and the decoder section decodes the output in accordance with the encoded input. In the encoder section, the multi-head self-attention mechanism plays a crucial role in enabling the transformer to identify both short- and long-term dependencies. To capture additional feature information, different parts of the temporal pattern receive varying amounts of attention. Transformers are more adept at anticipating time series issues since they comprehend the context better.

Mian (2023) employed a Transformer sequential-based approach to forecast the next day's closing prices using ten day rolling window. They used RMSE, MAE, and MAPE error terms to evaluate model's predictive abilities for Yahoo Finance, Facebook and JPMorgan stock price forecasting. In addition, they ran ARIMA, LSTM, and Random Forest models to give a better understanding of the proposed methods' capabilities compared to other popular machine learning methods for stock price prediction. Their study proved that Transformer based model outperformed all other methods for all three stocks according to all error terms. Another research done by Zhang et al. (2022) used Transformer inspired Attention models and tweets with 5 days look back window to forecast one day ahead stock prices. A comparison with baseline models such as ARIMA, CH-RNN, Adv-LSTM and others revealed that Transformer enhanced method was superior in 2 datasets of stocks in terms of accuraccy, falling second in the other two.

1.6. Previous Integrations of Machine Learning with Portfolio Optimization

Although portfolio optimization theory was created in 1950s, it is still widely used in the 21st century due to its reliability in core fundamentals (McClure, 2022). Nonetheless, as the time passed on, more techniques to find the best portfolio optimization strategies were analyzed. Tola, Lillo, Gallegati and Mantegna (2008) used cluster analysis to demonstrate increased model's accuracy based on forecasted and factual risk (Tola et al., 2008). Though the authors had strict assumptions of the market conditions including short selling and flawless prediction capability for future gain and volatility of

stocks, they also showed that clustering algorithms led to some of the results being confirmed under the more rational market conditions. Another impressive method for portfolio optimization was the particle swarm optimization, a heuristic approach which aims to mimic the communal interaction comparable to swarm of ants or flock of fish (Cura, 2009). The paper describes every representative of the swarm as a distinct portfolio. Then each representative recalls any other representative's best preceding location visited including its own and, in a way, creating a massive swarm whose every bit is moving closer to its best previous location and in turn moving closer to the best representative. More recently, with the advancements in technology, it has become feasible to analyze extensive historical price databases using computational systems (Chiang et al., 2016). The extensive utilization of intelligent predictive models through intensive computational methods is often referred to as machine learning in the research community. Hence, the next section will focus on machine learning approaches in particular to seeking optimal portfolio strategy.

Machine learning revolutionizes the optimization of financial portfolios by offering a fresh perspective. Unlike traditional approaches that emphasize quantitative analysis and hedging mechanisms, machine learning introduces several distinct advantages (ElectrifAi | Portfolio Optimization With Machine Learning, n.d.). Firstly, it enables the processing of vast amounts of data and the extraction of patterns that surpass the limitations of conventional mathematical methods. Secondly, machine learning effortlessly captures non-linear relationships and effectively reduces dimensionality, a feat typically unattainable through alternative means. Thirdly, within a machine learning algorithm, the intricate connection between risk and return, often involving numerous factors, can be comprehensively processed, and identified. Ultimately, reinforcement learning empowers machines to learn and continuously enhance their performance, surpassing human capabilities in portfolio optimization. In empirical studies employing machine learning, typically two primary stages are involved (Henrique et al., 2019). The initial phase focuses on selecting significant variables and models for prediction, where a portion of the data is set aside for training and validating the models to optimize their performance. In the subsequent phase, the optimized models are applied to separate data specifically designated for testing, enabling the assessment of their predictive capabilities. The fundamental techniques commonly employed in the literature encompass artificial neural networks (ANNs), support vector machines (SVMs), and random forests (RFs).

Conlon, Cotter and Kynigakis (2021) use machine learning and its dimensionality contraction method to analyze if factor-based covariance matrix established on this feature can improve minimum-variance portfolios made of separate securities (Conlon et al., 2021). The authors conclude that their used factors which were established on principal component analysis and partial least squares experienced a more solid dynamic with generally adopted factor proxies in comparison to autoencoders. In addition to that, they found that portfolios created from autonomous learning methods tend to need much less asset reallocation with their weights being more diversified and less volatile at the same time compared to not so autonomous analogues or recognized factors. The study achieved a statistically significant decrease of annual portfolio volatility by 3.3% when using the suggested models as well as a 25% boost in Sharpe ratio. Even more so, according to the paper, the average return for an investor who has medium or low risk appetite, was found to be statistically significantly higher by 2.5% to 4.5% when compared to equal-weighted allocation.

Although mean-variance method for portfolio selection is one of the most frequently used approaches, it may yield unseen results when established together with stock price prediction (Chen et al., 2021). The authors use eXtreme Gradient Boosting (XGBoost) and an improved firefly algorithm (IFA)

whose sole purpose is boosting the hyperparameters of the XGBoost to forecast the value of securities. The second step covers the mean-variance method application when selected stocks with higher probable yield are used for optimal portfolio creation. Particularly, for the portfolio optimization, authors randomly allocated a set of weights of various securities in different portfolios and measured the variance and mean return of such portfolios, constituting a Monte Carlo approach in use. In this process, 50 thousand random portfolios were created from which only one was selected based on mean-variance model, and namely, a Sharpe ratio to distinguish the best trade-off between risk and return. The best one selected had the greatest Sharpe ratio. The paper concluded that the created XGBoost model improved by IFA generated better portfolio in comparison to randomly selected stocks with MV applied for weight allocation and randomly selected stocks distributed equally. It was reported as well that stock price prediction models with applied MV gave better results than if stocks were allocated with equal weights concluding that MV method was necessary to create an optimal portfolio. And final finding showed that only 7 securities were enough for the investor to form a well-balanced portfolio.

Ma, Han, and Wang (2021) proposed to optimize the portfolio using machine learning and deep learning (Ma et al., 2021). Particularly, they suggested using random forest (RF) and support vector regression (SVR) models for machine learning methods, as well as LSTM neural network, deep multilayer perceptron (DMLP), and convolutional neural network as deep learning methods to forecast stock price movement in order to select the best candidates for optimal portfolio creation. As benchmarks to compete against, were created portfolios combining suggested stocks by autoregressive integrated moving average model. The paper analyzed China's 100 securities index from 2007 to 2015. It was found that random forest stock price prediction capabilities to select best stocks for the portfolio, combined with mean-variance with forecast method for portfolio creation, resulted in the best optimal portfolio. Once both parts of the equation were obtained – the GA and the stock candidates – MatLAB software was used to find the most optimal solution. The optimization outcome revealed that for the one year which was used as proxy investment horizon in the paper, the GA with selected stocks astonishingly outperformed the market result where the weights of four stocks were determined by deterministic programming approach. Although the systematic risk was higher for the GA model compared to the deterministic one, the Jensen's alpha was much more in favor for the GA approach, making it the preferable choice.

Although, here are presented only a few papers on machine learning utilization while solving portfolio optimization problem there are numerous other studies which due to a limited space were not looked at. Nevertheless, an elaborate review of many more studies on machine learning and its use on portfolio optimization was presented by Henrique, Sobreiro and Kimura (Henrique et al., 2019). In addition, they revise different markets, assets, predictive variables, predictions, main methods and performance measures used by other papers. From 57 papers reviewed, 17 of them used ML to predict prices which also will be the main predictor of this paper. Moreover, the majority of the studies used neural networks however implementing neural networks whether as a standalone procedure or an addition to genetic algorithms and support vector machines. Hence, this paper will also apply the best practice and will use some variation of neural network framework. Finally, the performance indicators were repeated by all papers, but the main three were singled out to be MSE, MAPE and MAE, therefore at same measurements will be used to benchmark created models in this paper as well.

2. Methodology

In this section, we investigate the predictive power of three advanced modelling approaches on stock price forecasting: Random Forest, ARIMA, and LSTM. Whereas the ARIMA model offers insights based on autocorrelation within time series data, the Random Forest model uses an ensemble learning strategy to improve prediction accuracy. Furthermore, the recurrent neural network model known as the LSTM model is applied because of its capacity to identify long-term dependencies in sequential data. These models have been carefully chosen to offer a thorough examination of price movements and patterns in the stock market. Meanwhile, other parts of this section explore other metrics for portfolio efficiency and methodologies for portfolio optimization, looking at how well they work to improve portfolio performance. This dual approach makes it possible to evaluate investment strategies and predictive models in the financial realm holistically.

The data used for analysis and the methods for creating and fine-tuning the models are covered in this section as well. The technical details of the models utilized, and the performance metrics utilized to evaluate their efficacy will be covered in the middle portion. The formulation of the optimization problem will come next. A number of criteria will be offered in the last section to assess the caliber of the optimized portfolio. The analysis was partially run on Google Colabs servers and a MacBook Pro with the following specifications – CPU: 2.4 GHz 8-Core Intel Core i9, GPU: Radeon Pro 560X 4GB, Intel UHD Graphics 630 1536MB, Memory: 32GB 2400 MHz DDR4.

2.1. Data Collection

The data for the practical experiment was collected from Yahoo Finance from 2014 to 2023 for each of the 20 constituents of OMX Baltic Benchmark Index. The trend of daily prices can be seen in the appendix, as well as the annualized returns of the stocks. For the majority of the constituents of mentioned index the trend was moving upwards. There were no missing values in the data sample. The Python libraries for common tasks among all models included analysis and models were executed using Python with prebuilt libraries such as *numpy* – used for working with arrays, *pandas* – used for data handling and creating lagged values, *matplotlib* – for plotting results, *yfinance* – used for downloading data from Yahoo Finance. The data for all three models was split the same way – 80% of it was used for training the model and the rest for testing it. As majority of the stocks had historical price data for the analyzed 10-year period – a training set included price data from 2014 to 2021 and the testing sample was constructed from price data from 2022 to 2023.

2.2. Evaluation of Machine Learning Models

As the aim of this analysis will be to determine the best prediction power having model identical comparison indicators must be employed. For this task standard accuracy metrics will used, most of which already were applied in the articles examined in literature review section. All three of the metrics will use the error term to determine the best model. If the model had the lowest error term of all three, it would mean it has the greatest accuracy. The error terms used for comparison will be the mean absolute error (MAE), the rooted squared mean error (RSME) and the means absolute percentage error (MAPE). The formulas for computing these indicators are as follows:

$MAE = \frac{1}{n}\sum_{t=m+1}^{n}|y_t - \hat{y}_t|$ for the mean absolute error, (1)

$RMSE = \sqrt{\frac{1}{n}\sum_{t=m+1}^{n}(y_t - \hat{y}_t)^2}$ for the rooted squared mean error, (2)

$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| * 100$ for the means absolute percentage error. (3)

Here *n* is the sample size, *y* is the actual values and $\hat{y}$ is the predicted values.

In addition to error term evaluation, a base model with moving averages (MA) will be created for forecasting and will be evaluated using the same error terms. The moving averages of 3, 5, and 10 days will be used to predict the next day's price. Those MA models will serve as a good benchmark against which, hopefully, other models will perform better. The three error terms will be calculated for the moving average models as well to have the same foundation for comparison.

2.3. Random Forest

The underlying idea behind the random forest model is the combination of multiple decision trees, which should yield better results and have greater predictive power compared to individual decision trees. In addition to the libraries mentioned in "Data Collection" part, a few others were employed for the random forest model: *sklearn.ensemble.RandomForestRegressor* to call the random forest and *sklearn.model_selection.RandomizedSearchCV* for hyperparameters' optimization. Before training the random forest model, data preprocessing is inevitable to avoid any potential errors later on in the analysis. Since there were no missing values in the historical stock price datasets for any constituent of the index, the data processing step will only focus on scaling the data. This is needed to ensure that the input characteristics have a similar scale and lessen bias towards variables with higher magnitude. After data processing, further step includes feature engineering. In this part it is typical to use technical indicators to provide the model with more information about historical trends and patterns. However, due to the simplicity and aim of this analysis—to demonstrate that with the simplest assumptions and ease of replication, retail investors can find inspiration to achieve better results than the market index. As selection of appropriate technical indicators requires advanced knowledge of financial economics, this part would not be easily replicable. Nonetheless, feature engineering will include lagged values of 30 days of stock prices prior to predicted day's price to forecast day ahead stock price. After that a model is developed using 80% of data as training data and 20% as training data. Before launching random search method, a base model with no parameters is trained and used to predict the future stock prices. The results of base model are measured using MAE, RMSE, and MAPE metrics. Then a selection of hyperparameters is entrusted to the random search method to determine the best values for the number of trees, depth of the trees, minimum samples split. Then the model will predict the future stock prices using 30-day look back period to predict price of one day ahead – similarly as it was done when predicting with base trained model. Finally, the tuned models will be evaluated using MAE, RMSE, and MAPE indicators. Additionally, graphs of testing data and predicted prices will be presented for visual comparison of prediction accuracy while a summary table with ticker, training time for each of the three models and their error terms will serve as a technical approach to compare the three methods.

2.4. Autoregressive Integrated Moving Average (ARIMA)

Successfully employing ARIMA model for stock price predictions requires going through the steps provided in the figure below (see Figure 3). First of all, the data will be checked for stationarity using the Augmented Dickey-Fuller (ADF) test. The test output shows the ADF score and p-value, which

if lower than 0.05 implies that the data is stationary. If, however, the p-value is greater than 0.05, then the null hypothesis that data is not stationary, cannot be rejected and it means there are additional steps to be taken to make the data stationary. These steps would include logarithmic transformation or differencing.

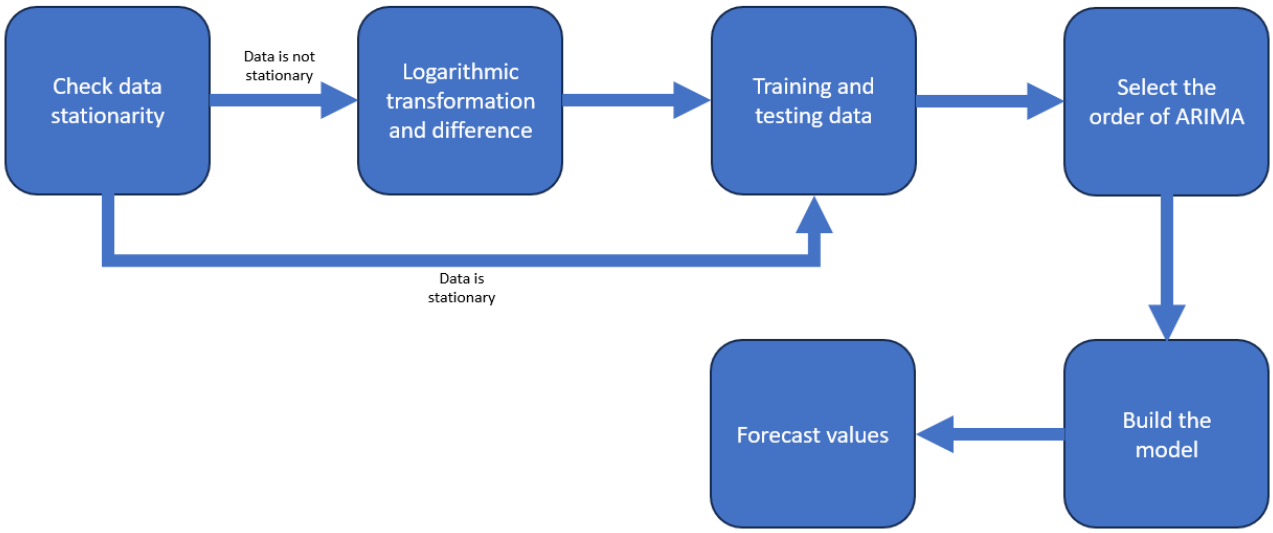


Figure 3. Process of ARIMA model preparations

After ensuring that the data is stationary, it is split to training and testing datasets at a ratio of 80:20. The selection of the ARIMA order is done afterwards. A formula for ARIMA can be expressed as follows:

$$y_t = I + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_p y_{t-p} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} \quad (4)$$

It is also referred to as ARIMA(p, d, q) model. The lagged p data points for the autoregressive component of the equation and the lagged q errors for the moving average part, which are all differenced, are the predictors. During training period, ARIMA calculates coefficients α and θ for the provided values of *p*, *d*, and *q* (Kotu & Deshpande, 2019). Though it might be hard to pick the most efficient such values manually, using *pmdarima* library in Python and function *auto_arima* will save time. This function evaluates different ARIMA orders by AIC score and suggests the one with the lowest AIC as the best fit.

Once the most appropriate ARIMA order is known, a model is built in reference to the best fit ARIMA order. When the model has trained, it will be used to predict one step at a time using all available historical data. The predicted stock prices will be forecasted for the whole test period and then compared to the actual stock prices in the test dataset. The predicted and actual prices will be displayed visually as well as the three metrics of error term will be calculated to later compare the model's accuracy with other machine learning algorithms. A summary table of each stock with their ARIMA model's order and the three main indicators will be provided to conclude the potential of ARIMA for the stock price forecasting in the Baltic market.

2.5. Long Short-Term Memory Networks

LSTM networks have been introduced back in 1997 as an improvement to then existing recurrent neural networks RNN to solve an underlying issue in latter's methodology. RNNs were not capable

to address the problem of erupting or fading gradients (Sak et al. 2014). However, the LSTMs not only were able to solve this issue of RNNs but were primarily great at learning long-term dependencies.

In literature review there was presented a structure of LSTM memory cell, containing three gates: forget gate, input gate, and output gate. Those gates are provided with the input x_t at every timestep t. Additionally, the output of a memory cell – h_{t-1} – is provided at timestep $t-1$. The cell states s_t and outputs h_t of the LSTM layer can be calculated through a process of 4 steps. During the initial phase, the LSTM layer decides which data from its prior cell states s_{t-1} should be deleted. As a result, bias terms b_f of the forget gates together with the current input x_t and the outputs h_{t-1} of the memory cells at the previous timestep $(t-1)$ are used to calculate the activation values f_t of the forget gates at timestep t. After that scaling is performed by sigmoid function into range from 0 to 1, where 0 means to completely forget and 1 to completely remember:

$$f_t = sigmoid(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f)\ . \tag{5}$$

The following phase describes how the LSTM layer chooses which data to add to the network's cell states (s_t) in the second step. There are two steps to this process: first, candidate values $\tilde{s}_t$ that might be added to the cell states are calculated. Then, the input gates' activation values i_t are computed:

$$\tilde{s}_t = \tanh\,(W_{\tilde{s},x}x_t + W_{\tilde{s},h}h_{t-1} + b_{\tilde{s}}), \tag{6}$$

$$i_t = sigmoid(W_{i,x}x_t + W_{i,h}h_{t-1} + b_i). \tag{7}$$

The Hadamard (elementwise) product is shown by ◦ in the third step, which calculates the new cell states s_t depending on the outcomes of the preceding two steps:

$$s_t = f_t \circ s_{t-1} + i_t \circ \tilde{s}_t. \tag{8}$$

Final phase portrays two equations which explain how the output h_t of the memory cells is obtained:

$$o_t = sigmoid(W_{o,x}x_t + W_{o,h}h_{t-1} + b_o), \tag{9}$$

$$h_t = o_t \circ \tanh\,(s_t). \tag{10}$$

The notation for the vectorized equations mentioned above is explained below:

- The input vector at timestep t is denoted by x_t.
- Weigh measures are $W_{f,x}$, $W_{f,h}$, $W_{\tilde{s},x}$, $W_{\tilde{s},h}$, $W_{i,x}$, $W_{i,h}$, $W_{o,x}$, and $W_{o,h}$.
- The bias vectors are b_f, $b_{\tilde{s}}$, b_i, and b_o.
- The activation values of the corresponding gates are represented by the vectors f_t, i_t, and o_t.
- The vectors for the cell states and potential values are denoted by s_t and $\tilde{s}_t$.
- A vector representing the LSTM layer's output is h_t.

In addition to libraries mentioned in “Data Collection” part, a handful of others were used in LSTM analysis, such as *Keras* and *TensorFlow*. Keras is an advanced neural network API that can be used with TensorFlow. Because of its modular design and ease of use, experimenting with various neural network designs is a breeze. Layers such as LSTM, Dense, Dropout, Input, LayerNormalization, MultiHeadAttention will be used for building layers of a deep learning model. Tensorflow is an open-source machine learning platform that provides extensive tools, libraries, and community resources

to enable academics to push the boundaries of machine learning while making it simple for developers to create and implement ML-powered apps.

Contrary to the previously used models like RF and ARIMA, where an automated solution was employed to find the best hyperparameter set, for LSTM model the optimal parameters will be selected through trial-and-error process. The base LSTM model will be constructed likewise: the first LSTM layer with return sequence turned on will consist of 128 neurons which can be seen as a dimensionality of the output space. Initializing the return sequence in the first layer instructs it to provide each sample's complete output sequence. This is required because the sequence's subsequent layer—an additional LSTM layer—requires a sequence input. Then an initial dropout layer will be set at 0.2 indicating that 20% of the input units are randomly set to 0 at each update during training time. This is done in order to reduce the probability of overfitting the model by limiting its sensitivity to specific weights of neurons. After that a second LSTM layer with 128 neurons is used, however this time with return sequence turned off. This change serves as an indication that the output sequence will be reduced to a single vector because the layer will only return the final output. This configuration is common in many LSTM architectures, when prediction requires only the final context vector. After that another dropout layer of 0.2 is placed, again to help against overfitting the model. Then a dense layer is defined with 32 neurons and a ReLU (rectified linear unit) activation function which is frequently used to add non-linearities to the model that are required in order to learn more intricate functions. It is followed by the last layer of the model which is a completely linked layer that contains just one neuron. A linear activation function is employed which is useful for regression problems (e.g., forecasting a continuous value) because it enables the model to produce values within a problem-appropriate range. Finally, the model is compiled using the Adam optimizer, a well-liked option that combines the benefits of two further extensions of stochastic gradient descent and a loss function for mean squared errors. This is needed to calculate the average squared difference between the estimated and actual values, or the average of the squares of the errors. The LSTM model will be run for 50 epochs with a batch size of 32. The number of epochs means that the model will go through the entire training dataset 50 times. During each epoch, it will update its weights iteratively based on the training data, aiming to minimize the loss function. When the batch size is set to 32, the LSTM model will process 32 samples at a time during training before updating its weights. This configuration affects training performance, memory consumption, and gradient estimate quality. To train LSTM models effectively and efficiently, the proper batch size must be determined. A validation split parameter of 20% will be used meaning that twenty percent of the training data should be reserved as validation data. The model will utilize this data to assess the loss and any model metrics at the conclusion of each epoch, but it will not use it for training. This aids in tracking the model's effectiveness and detecting overfitting. Finally, the base model will use 30 days of a look back window, meaning that it will use past 30 days' information on stock's price to predict the next day's price. From all hyperparameters described only the following will be changed:

- The number of neurons in the first and second layers.
- The number of epochs.
- The batch size.
- The number of days for look back window.

As an alternative to the best LSTM model an LSTM with transformers model will be trained and used to predict stock prices. Similarly to the best LSTM model, it will take all of its features, which means

that the LSTM with transformers will have the same number of neurons in the first two layers as the best LSTM model. In addition, there will be the same dropout layers each equal to 0.2, again to help address the overfitting problem of the model. However, the novelty to the LSTM with transformers model comes from the implementation of a multi-head attention layer that enables the model to concurrently process data from many representation subspaces. Two parameters are used in this layer: the number of heads and the size of each head. The parameter, *key_dim* which determines the size of each head will be set to 256 while the parameter *num_heads* will be set to 4 heads indicating that the attention mechanism is split into 4 heads. The layer is able to identify the most significant portions of the input sequence since it has access to *x* as both its input and its context. All other parameters and layers are kept identical to the best LSTM model used.

2.6. Evaluation Indicators for Portfolio Optimization

2.6.1. Return on Investment (ROI).

A ratio known as return on investment (ROI) compares an investment's gain or loss to its cost in order to determine how profitable it was. It aids in determining the possible return on investments made in items like stocks or company endeavors. ROI can be determined using a certain formula and is typically expressed as a percentage:

$$ROI = \frac{Final\ Value\ of\ Investment - Initial\ Value\ of\ Investment}{Initial\ Value\ of\ Investment} * 100\% \quad (11)$$

Return on Investment (ROI) is one of the simplest methods to compare two investment options. For instance, if an investor places money in an actively managed fund with a risk level similar to that of a market index for an equivalent period, they can determine whether the actively managed fund has outperformed or underperformed the market. However, basing the evaluation of an investment's superiority or inferiority solely on ROI, particularly over shorter periods (less than five years), may lead to incorrect conclusions about the validity of the investment decision. Nevertheless, under straightforward assumptions, ROI can quickly provide insight into whether the investment was effectively good or bad during the selected timeframe compared to the decision to invest into exchange traded fund which replicates market index.

2.6.2. Standard Deviation.

The degree to which the investment returns vary from the mean of the probability distribution of investments is indicated by the standard deviation of the portfolio. In other words, it informs investors of the amount that the investment will diverge from the anticipated return. The standard deviation of portfolio consisting of only two assets can be expressed as follows:

$$\sigma_p = \sqrt{(w_1^2 * \sigma_1^2 + w_2^2 * \sigma_2^2 + 2 * w_1 * w_2 * \sigma_1 * \sigma_2 * \rho_{12})}, \quad (12)$$

where:

- w_1 is weight of asset no. 1 in the portfolio.
- w_2 is weight of asset no. 2 in the portfolio.
- σ_1 is standard deviation of asset no. 1.
- σ_2 is standard deviation of asset no. 2.
- ρ_{12} is correlation of asset no. 1 and asset no. 2.

A low standard deviation indicates a portfolio with a lengthy history of reliable results (Banton, 2022). A fund focused on growth, or emerging markets is probably going to be more volatile and have a higher standard deviation. It is therefore more dangerous by nature. However, this relationship between risk and return does not indicate that that with high risk comes great return, though high return will never be made out from a low-risk investment. Standard deviation also has some setbacks to consider, one of them being incapability to show how the portfolio or a fund is performing against a benchmark – it only shows stability or instability of portfolio's return. Presumption that data values follow a normal distribution is another drawback of standard deviation as this tendency is not present in many portfolios. Hence, similarly to ROI, standard deviation as a standalone performance indicator is a poor choice, but in usage with others can help to perform an extensive and informative analysis of a portfolio.

2.6.3. Jensen's Alpha.

The extra return that an investment portfolio achieves over its projected performance, as predicted by the Capital Asset Pricing Model (CAPM), is measured by Jensen's Index, commonly referred to as alpha. The method used to create this index is to compare the portfolio's actual returns to the expected returns given its risk (measured by beta), where the predicted returns are derived from the market premium multiplied by the risk-free rate and the portfolio's beta (Leković, 2017; Jobson & Korkie, 1984).

$$\alpha = R_p - (R_f + \beta(R_m - R_f)), \tag{13}$$

where:

- R_p is the actual return of the portfolio.
- R_f is the risk-free rate of return.
- β is the portfolio's beta, which measures the sensitivity of the portfolio's returns to the returns of the market.
- R_m is the expected market return.

Utilizing Jensen's Index to assess investment portfolios has several benefits. First of all, by expressing the degree to which the management has increased or decreased value in comparison to the market, it provides a simple and transparent measure for evaluating a manager's success. Compared to straightforward return comparisons, it is a more comprehensive risk-adjusted performance indicator since it incorporates the risk of the portfolio through the beta measure. Additionally, this function makes it possible to compare several portfolios, which helps investors choose funds or portfolios with confidence.

Jensen's Index does, however, have a number of drawbacks. The accuracy of the index is dependent on the Capital Asset Pricing Model (CAPM) being a suitable model for market returns, which is one of its main limitations. In the event that the assumptions behind the CAPM are incorrect, Jensen's Alpha may not fairly depict the manager's performance. If the market benchmark is chosen incorrectly, it can potentially produce deceptive findings as it has a substantial impact on the estimated alpha. Furthermore, because it places so much emphasis on performance during a predetermined time frame, Jensen's Index may incite managers to prioritize short-term rewards above long-term strategy and risk management. Finally, it could oversimplify the evaluation of a manager's

performance by ignoring additional elements that can affect returns, like changes in market volatility or macroeconomic developments.

In general, Jensen's Index is a helpful tool for assessing portfolio performance, but how well it works depends on how the CAPM is used and which benchmark is selected. In order to have a more comprehensive understanding of a manager's performance, analysts and investors should combine it with additional tools and measurements.

2.6.4. Sharpe Ratio.

William F. Sharpe created the Sharpe Ratio in 1966 to evaluate an investment's performance to a risk-free asset once its risk has been taken into account. It is calculated by dividing the difference between the investment returns and the risk-free return by the risk-representing standard deviation of the investment returns. It is used a useful metric for determining how well an asset's return covers an investor's risk. The asset with a greater Sharpe Ratio offers a better return for the same risk when two assets are compared to the same benchmark. The Sharpe Ratio formula is provided below:

$$S_p = \frac{R_p - R_f}{\sigma_p}, \tag{14}$$

where:

- R_p is the return of the portfolio.
- R_f is the risk-free rate of return.
- σ_p is the standard deviation of the portfolio's excess returns.

The popularity behind Sharpe Ratio among many lies in its simplicity as it is easy to calculate and widely understood and used in the financial industry. This simplicity makes it a go-to metric for performance evaluation among both academics and practitioners (Levy, 2017). Additionally, the ratio is praised for its standardization and performance comparison. It provides a standardized way of comparing investments across different types and sectors by normalizing their returns against their risks (Brinză, Ioan, & Lazarescu, 2023). This standardization allows investors and managers to compare different investments on a like-for-like basis, irrespective of their risk levels (Levy, 2017).

However, the simplicity to use has its own cost as the Sharpe Ratio considers only the standard deviation to measure risk, ignoring other factors like the shape of the return distribution, which can lead to underestimation or overestimation of true economic risk (Levy, 2017). Moreover, the ratio assumes that returns are normally distributed, which may not always be the case. This can lead to misrepresentations of risk in assets with skewed or kurtotic return distributions (Levy, 2017).

Though overall the Sharpe Ratio is a useful tool for assessing risk-adjusted performance, depending entirely on this metric might cause one to miss out on significant aspects of investing risk, especially in portfolios that have highly asymmetric or non-normal return distributions. For a complete understanding of investment risks and returns, it is imperative to utilize the Sharpe Ratio in conjunction with other metrics and qualitative evaluations.

2.6.5. Value at Risk (VaR).

A metric known as value at risk, or VaR, measures the potential extent of financial losses in a position, portfolio, or company over a given period of time (Kenton, 2024). The metric is used primarily by

financial institutions like investment or commercial banks in order to evaluate the likelihood and magnitude of possible losses in their institutional portfolios. There are three key aspects that VaR helps to understand about the exposure of financial impact: the prospective loss amount, the likelihood that the loss will occur, and the time horizon. For instance, if a portfolio was concluded to have 2% one-month VaR of 5%, it means that there is a 2% probability for the portfolio value to lose 5% of its value during one-month horizon.

2.7. Portfolio Optimization

After determining the best machine learning approach for the price prediction, a portfolio optimization technique following the Markowitz's Modern Portfolio Theory will be employed. It means that for a set amount of risk a model will try to maximize returns. The portfolio optimization task will be carried out with the help of three new functions defined. The first function defines buying the stocks by splitting the money available for investment according to the weights of different stocks. Then the buying process is initiated by buying as many stocks as possible with allocated amount of money for that stock. Buying fractional shares is allowed. The second function defines the process of selling stocks. It calculates the amount of money available after selling all the stocks in the portfolio at last day's closing prices. The final and most important function is responsible for portfolio optimization. In this function stock weights will be rebalanced every day before the hypothetical trading commences. This implies that the entire portfolio will be sold at the beginning of each day based on closing prices of the last day. Then using predicted values by the ARIMA model a new set of weights is determined. After that a new portfolio with newly computed weights is constructed by buying the stocks at the last day's closing price. The constraints of this optimization method are:

- The sum of the weights of the constituents of the portfolio cannot exceed 1.
- The minimum weight of one stock should not be lower than 0.1% and cannot exceed 25%. This ensures that portfolio is diversified.
- The penalty factor is introduced to discourage stock concentration with high weights. This additionally improves diversification.
- The set risk for the optimal portfolio is set to daily volatility of OMXBBGI index from January 1st, 2014, to November 30th, 2023. It means that when rebalancing the portfolio based on the predicted prices, the risk of the newly constructed portfolio cannot exceed historical daily volatility of the index.

After setting the constraints for the optimization problem, the target of this task will be to maximize daily returns for the portfolio. Then a calculated result should be provided to show the portfolio's value after daily rebalancing at the end of December. Additionally, a graph will be provided to display the differences in portfolio values every day for the simulation month.

3. Experimental Setup and Results

The results of applying the Random Forest, ARIMA, and LSTM models to the task of the stock price prediction are displayed in this section. Evaluation of each model's predicted performance in comparison to the benchmark MA models and discussion about the advantages and disadvantages of each one in relation to the provided dataset takes place. This section also juxtaposes the consequences of these discoveries with the performance measures of the different portfolio strategies that were previously examined. The purpose is to determine which model best captures market trends and provides dependable predictions through a thorough examination of the data, ultimately assisting in the making of the best possible investment decisions. This thorough analysis contributes to the integration of theoretical modelling and real-world investment applications.

The benchmark models were created using three different moving averages – 3-day, 5-day, and 10-day MA models – to predict the next day's price. The summary results of MAPE values for each model is presented in Table 2. The average MAPE values for these models are 0.90, 1.12 and 1.60 respectively which means that MA could also be used as a high forecasting accuracy yielding method keeping in mind that such model does not require any time to train.

	3-day MA	5-day MA	10-day MA
Ticker	**MAPE**	**MAPE**	**MAPE**
IGN1L.VS	0,39	0,48	0,64
TAL1T.TL	1,02	1,26	1,81
EGR1T.TL	1,06	1,29	1,69
SAB1L.VS	1,09	1,37	2,05
LHV1T.TL	1,01	1,28	1,78
TKM1T.TL	0,56	0,70	0,98
TEL1L.VS	0,51	0,63	0,88
TSM1T.TL	0,54	0,72	1,10
CPA1T.TL	0,78	0,95	1,29
MRK1T.TL	0,91	1,18	1,70
TVE1T.TL	0,60	0,73	1,03
HAE1T.TL	1,04	1,29	1,77
GRG1L.VS	1,11	1,39	1,99
AKO1L.VS	1,12	1,40	2,05
APG1L.VS	1,11	1,42	2,16
PKG1T.TL	1,51	1,78	2,61
VLP1L.VS	0,98	1,21	1,75
NTU1L.VS	1,06	1,28	1,73
AUG1L.VS	0,93	1,19	1,81
HPR1T.TL	0,66	0,84	1,22
Average	**0,90**	**1,12**	**1,60**

Table 1. Summary table of MA models' average MAPE values.

3.1. Machine Learning Model Performance

3.1.1. Results of Random Forest

The random forest forecasting required defining two functions that are later used for the analysis. The first one's purpose was to calculate MAE, RMSE, and MAPE error terms. The other one was used to process the dataset to format it into a structure suitable for training the random forest models. Additionally, look back window parameter is used that will be utilized to create feature sets. Each feature set, or model input sequence, will contain 30 consecutive data points from the dataset because the set size is 30. This means that the model will use past 30 days to predict price of a stock one day

ahead of the input sample. After defining the two functions a loop with following steps is run for every stock:

- Download available historical data from Yahoo Finance from 2014 to 2023.
- Data scaling to have a similar scale and lessen bias towards variables with higher magnitude.
- Split the data into training and testing datasets at a ratio of 80:20.
- Train and predict the next day's prices for all three models: the base model, the random search enhanced model and the Bayesian search enhanced model.
- Evaluate the results of all three models using MAE, RMSE, and MAPE indicators.
- Visualize training data, testing data, and predicted prices.
- Save results to a summary table for easier comparison of different RF models.

The summary results of each model are presented in Table 3 while the result graphs are provided in Appendix 3. The random forest base model was created to be used as a benchmark to compare random forest with random search and random forest with Bayesian search for parameter optimization. It took only 37.5 seconds in total to train the base model with longest training time being 4.6 seconds for "Telia Group AB". The MAPE values for the random forest base model varied from 2.47% to 77.6% averaging 26.6%. The high MAPE values could be expected from the base model since no additional parameters were added to the model.

The random forest with random search algorithm to find optimal hyperparameters was tried next. Compared to an exhaustive search, this method finds a decent collection of hyperparameters more quickly in a potentially huge space, especially when there are a lot of possible combinations. The number of trees in the forest for random selection was set from 100 to 1000, while the depth of each tree was set to be randomly selected from a range of 5 to 100. The random search then went through 50 iterations of randomly selected number of trees in the forest and tree's depth to choose the most optimal set to use for training. It took almost 31 hours to train all random forest models with random search optimal hyperparameters for all stocks. The MAPE values ranged from 2.38% to 87.6% with an average of 25.38% showing only slight improvement over the base model. The result graphs are provided in Appendix 3 displaying quite poor predictive capabilities of this approach. Taking into consideration the time needed to train the models, the conclusion is clear that RF with random search hyperparameter tuning is not a reasonable choice.

The third random forest model employed Bayesian search for hyperparameter tuning. Because it creates a probabilistic model of the function translating hyperparameter values to the target evaluated on a validation set and uses that model to choose which hyperparameters to evaluate next, this method is especially helpful. This method may be more efficient—especially in high-dimensional spaces – than grid search and random search since it can identify better parameters in fewer stages. The efficiency compared to random search is evidently visible as the total amount of time required for the Bayesian search random forest models to train was 21.5 hours, 30% less compared to the random search training time. The MAPE values ranged from 2.99% to 86.2% resulting in an average value of 25.65%. The error term is still large and improved by only 1pp from the base model leading to the same conclusion as for the random search hyperparameter tuning: this path to forecast prices is not worth the time when the base model can yield very similar results for practically no training time required.

Ticker	Base Model Train Time (min)	MAE Base Model	RMSE Base Model	MAPE Base Model	Random Search Time (min)	Random Search Train Time (min)	MAE Random Search	RMSE Random Search	MAPE Random Search	Bayesian Search Time (min)	Bayesian Train Time (min)	MAE Bayesian Search	RMSE Bayesian Search	MAPE Bayesian Search
IGN1L.VS	0,01	0,06	0,09	77,64	0,00	14,08	0,07	0,09	87,63	0,00	18,15	0,07	0,09	86,19
TAL1T.TL	0,05	0,12	0,19	8,51	0,00	49,05	0,12	0,19	8,43	0,00	48,04	0,12	0,19	8,55
EGR1T.TL	0,01	0,15	0,22	72,93	0,00	13,44	0,16	0,22	70,51	0,00	12,82	0,15	0,22	73,18
SAB1L.VS	0,05	0,14	0,18	16,30	0,00	44,15	0,14	0,18	16,31	0,00	436,47	0,14	0,18	16,30
LHV1T.TL	0,03	0,06	0,10	5,90	0,00	30,13	0,06	0,10	5,96	0,00	27,25	0,06	0,10	6,02
TKM1T.TL	0,04	0,06	0,08	6,20	0,00	38,67	0,06	0,08	6,16	0,00	39,31	0,06	0,08	6,19
TEL1L.VS	0,08	0,12	0,16	11,90	0,00	295,62	0,13	0,17	12,11	0,00	188,33	0,12	0,17	11,92
TSM1T.TL	0,02	0,28	0,45	14,16	0,00	39,05	0,28	0,45	14,17	0,00	16,64	0,28	0,45	14,15
CPA1T.TL	0,02	0,03	0,04	6,02	0,00	18,72	0,03	0,04	6,06	0,00	13,18	0,03	0,04	6,52
MRK1T.TL	0,04	0,12	0,18	11,68	0,00	38,09	0,12	0,18	11,51	0,00	224,40	0,12	0,19	11,88
TVE1T.TL	0,02	0,06	0,08	55,80	0,00	25,51	0,05	0,07	45,43	0,00	23,81	0,05	0,07	45,46
HAE1T.TL	0,03	0,05	0,09	7,93	0,00	485,40	0,05	0,09	7,27	0,00	24,76	0,06	0,10	8,23
GRG1L.VS	0,03	0,10	0,16	20,28	0,00	561,19	0,10	0,15	18,43	0,00	32,68	0,10	0,15	18,71
AKO1L.VS	0,03	1,21	1,42	59,65	0,00	31,80	1,21	1,42	60,00	0,00	29,49	1,21	1,42	59,60
APG1L.VS	0,04	0,06	0,11	13,15	0,00	33,37	0,06	0,11	13,11	0,00	27,55	0,06	0,11	12,28
PKG1T.TL	0,03	0,07	0,09	5,56	0,00	31,47	0,06	0,08	5,29	0,00	33,00	0,06	0,08	5,45
VLP1L.VS	0,03	0,73	0,81	40,56	0,00	34,64	0,72	0,80	39,82	0,00	31,22	0,73	0,81	40,41
NTU1L.VS	0,02	0,02	0,02	2,47	0,00	22,52	0,01	0,02	2,38	0,00	21,83	0,02	0,02	2,99
AUG1L.VS	0,04	0,05	0,07	64,87	0,00	40,85	0,04	0,07	46,02	0,00	35,00	0,04	0,07	47,94
HPR1T.TL	0,01	0,42	0,51	31,03	0,00	7,87	0,42	0,51	30,91	0,00	7,08	0,42	0,51	31,03

Table 2. Summary table of Random Forest models‘ training times and error terms.

Certainly, the random forest models can be used to predict stock prices more accurately, however due to the purpose of keeping it simple and replicable no technical indicators or more sophisticated feature engineering took place.

3.1.2. Results of ARIMA

Checking the stock price data for stationarity is the first step following the methodology. The ADF test was performed for all 20 constituents of the OMXBBGI index. Summary results can be examined in Table 1. Though there were two stocks – “Enefit Green AS” and “Novaturas AB” which price data came relatively close to stationarity, since the p-values were not lower than 0.05, the null hypothesis that data is not stationary could not be rejected. It means that historical stock price data for all securities in the index is non-stationary. The next step would include taking the first difference in the price data to ensure stationarity. If the data would still be non-stationary, then second differences would be employed or logarithmic transformation.

Ticker	ADF Statistic	p-value
IGN1L.VS	-1,81	0,38
TAL1T.TL	-1,58	0,50
EGR1T.TL	-2,28	0,18
SAB1L.VS	-1,20	0,67
LHV1T.TL	-0,85	0,80
TKM1T.TL	-1,67	0,45
TEL1L.VS	-1,02	0,74
TSM1T.TL	-0,21	0,94
CPA1T.TL	-1,65	0,46
MRK1T.TL	-1,11	0,71
TVE1T.TL	-1,75	0,41
HAE1T.TL	-1,48	0,54
GRG1L.VS	-0,89	0,79
AKO1L.VS	-0,60	0,87
APG1L.VS	-1,93	0,32
PKG1T.TL	-1,16	0,69
VLP1L.VS	0,02	0,96
NTU1L.VS	-2,29	0,17
AUG1L.VS	-2,05	0,27
HPR1T.TL	-2,08	0,25

Table 3. Stationarity check for the constituents of the index.

However, manually differencing and looking for the best difference order with 20 different stocks might be a time-consuming task. Therefore, an *auto_arima* function is employed to look for the best autoregressive, difference and moving average orders or *p*, *d*, and *q*. The process was looped in Python to make the analysis more efficient. The loop included the following steps:

- Download available historical data from Yahoo Finance from 2014 to 2023.
- Select the best ARIMA model using *auto_arima* function.
- Split the data into training and testing datasets at a ratio of 80:20.
- Predict the next day's prices based on all available data prior to the forecasted day's price.
- Evaluate the results using MAE, RMSE, MAPE metrics.
- Save the results to a data frame for the ease of comparison, including the following columns – *Ticker* to identify the stock, *ARIMA Order* to show the best ARIMA model selected by *auto_arima* function used for future stock price predictions, *Train Time* to understand how long it took for the model to train, *MAE* for mean absolute error, *RMSE* for the root mean squared error, and *MAPE* for the mean absolute percentage error.
- Plot and save charts with the train stock price data, the test stock price data, and the predicted stock prices.
- Export results data frame to excel file for easier result examination.

The results of looped ARIMA forecasts are presented in Table 2. For the majority of stocks, the training time for ARIMA model took under 3 minutes (15 out 20 stocks) while the total amount of time required for all trainings to complete was 55 minutes and 37 seconds. For 9 out of 20 securities the autoregressive order was selected to be 0 as the best fit. This means that the model does not use any lagged terms of the series itself in the forecasting equation. It rather relies solely on other components, like the moving average part instead of predicting future values based on past values directly. The stock price data for all constituents of the index except "Novaturas AB" was stationary after the first differences, while for the latter it needed second differences to achieve stationarity. The last part of the ARIMA model – the moving average part – that indicates how many lagged forecast errors the model uses in the prediction equation. For 5 stocks this order was equal to 0 implying the model, relying solely on autoregression and differencing, is well-suited for cases where the stock prices are believed to follow a "random walk with drift" or some other autoregressive process. For other 5 securities it was 1 and for other 6 it was 2, meaning that this model is particularly good at adjusting predictions based on the most recent unexpected changes (errors), which can be useful in volatile markets like stock trading where recent events can significantly impact future prices.

Ticker	ARIMA Order	Train Time (min)	MAE	RMSE	MAPE
IGN1L.VS	(1, 1, 1)	0,26	0,06	0,09	0,30
TAL1T.TL	(0, 1, 2)	2,78	0,00	0,01	0,78
EGR1T.TL	(0, 1, 1)	0,16	0,03	0,04	0,73
SAB1L.VS	(0, 1, 0)	0,82	0,01	0,01	0,83
LHV1T.TL	(0, 1, 2)	0,64	0,03	0,04	0,75
TKM1T.TL	(0, 1, 3)	2,59	0,04	0,07	0,41
TEL1L.VS	(2, 1, 0)	1,48	0,01	0,01	0,40
TSM1T.TL	(2, 1, 2)	0,94	0,01	0,01	0,43
CPA1T.TL	(0, 1, 1)	0,37	0,02	0,02	0,61
MRK1T.TL	(2, 1, 4)	15,63	0,09	0,15	0,62
TVE1T.TL	(2, 1, 2)	7,53	0,06	0,09	0,46
HAE1T.TL	(2, 1, 2)	5,29	0,04	0,09	0,77
GRG1L.VS	(0, 1, 0)	0,7	0,01	0,01	0,80
AKO1L.VS	(3, 1, 3)	4,36	0,01	0,02	0,88
APG1L.VS	(0, 1, 0)	0,65	0,02	0,04	0,77
PKG1T.TL	(1, 1, 1)	1,34	0,01	0,02	1,22
VLP1L.VS	(0, 1, 0)	0,73	0,03	0,05	0,74
NTU1L.VS	(1, 2, 5)	5,99	0,03	0,04	0,81
AUG1L.VS	(2, 1, 2)	2,96	0,00	0,01	0,66
HPR1T.TL	(3, 1, 1)	0,31	0,04	0,06	0,58

Table 4. Summary table of best ARIMA models and error terms.

The other part of the ARIMA model forecasts is to analyze error terms to evaluate the accuracy of this model's predictive capabilities. To check if there is a relationship between training times and the MAPE error terms, correlation was computed. If training times indeed were highly correlated with error terms, a strong negative relationship would be observed, where increasing training time would lead to lower error terms. The calculated correlation results were -0.03676, implying almost negligible negative correlation between training times and MAPE. Additionally, for all of the constituents of the index, the MAPE value was below 1%, "Merko Ehitus AS" being an exception in this case with MAPE value reaching 1.22%. Nonetheless, following Moreno et al. (2013) methodology, a MAPE of under 10% means a highly accurate forecasting. In addition, comparing ARIMA MAPE values to the benchmark models, it can be concluded that for each stock ARIMA model produced smaller mean absolute percentage error terms.

3.1.3. Results of LSTM and LSTM with Transformers

Before the training of the models could take place four functions were created to help with the analysis. The first function was used to create a dataset that is later used with LSTM and LSTM with transformers models. The function's goal was to convert an input dataset into target values (Y) and input sequences (X) that can be used to train the models. In this function a look back window parameter was defined by the amount of prior time steps to use as input variables to forecast the subsequent time step. Since the default value was 30, there were 30 time steps in every input sequence, however this number was changed to 60, 10 and 5 to check if model's accuracy changes. The second function was created to evaluate the model by MAE, RMSE, and MAPE error terms. The last two functions defined LSTM and LSTM with transformers models respectively with the parameter detailing provided in the methodology. Then a loop is executed to perform the following steps:

- Download available historical data from Yahoo Finance from 2014 to 2023.

- Generate the dataset for LSTM and LSTM + Transformers models. Split data into training and testing datasets at a ratio of 80:20.
- Train the LSTM model and measure time required for the task. Predict the prices and estimate the error terms.
- Train the LSTM + Transformers model and measure time required for the task. Predict the prices and estimate error terms.
- Visualize the results in graphs.
- Save the results to summary data frame.

The general LSTM model was run under 12 different parameter combination – the scheme of parameter changes can be found in Figure 4 while the summary statistics of training time required and MAPE can be analyzed in Tables 5 and 6 and the result graphs can be examined in Appendix 5.

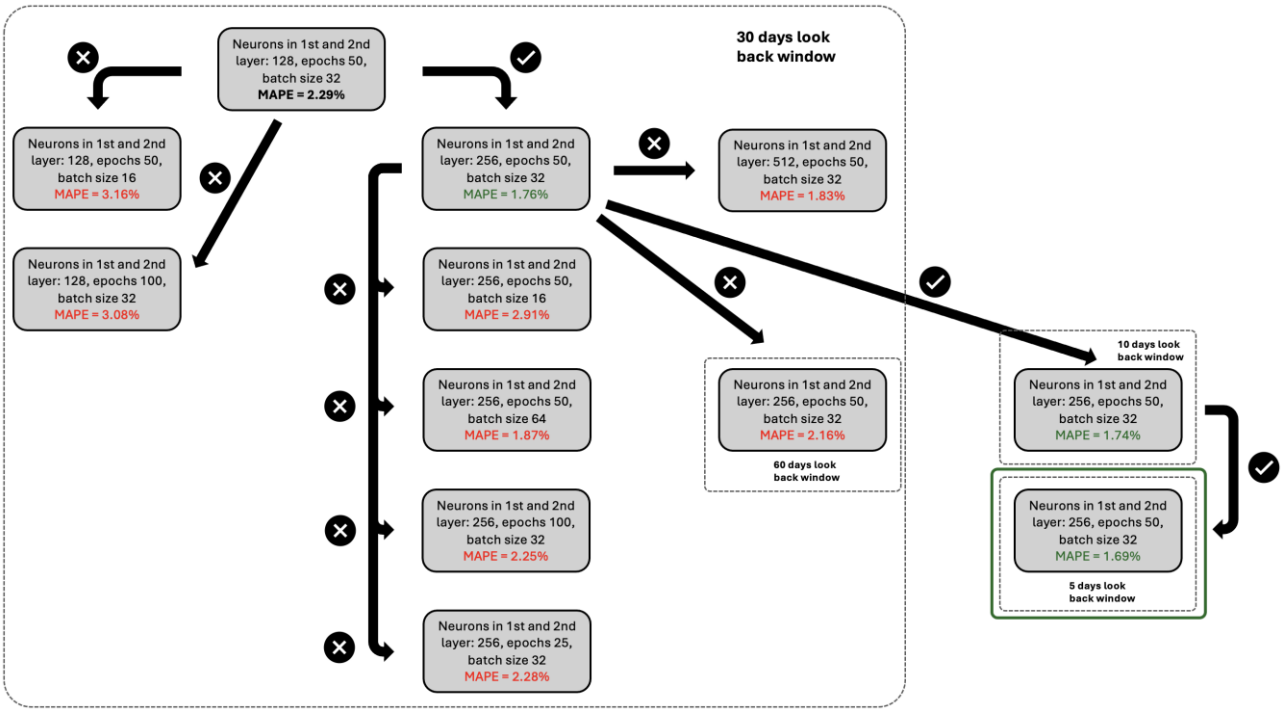


Figure 4. LSTM trial-and-error process.

The initial LSTM model was run with 128 neurons in the first layer and the same number of neurons in the second layer. The model ran through 50 epochs with a batch size of 32 and produced a MAPE equal to 2.29% over almost 51 minute of training time. The first trial was to change the number of batch size to 16, however the outcome was an increased error term to 3.16%. Then it was decided to change the number of epochs to 100 as it was believed that more iteration would lead to better weight allocation. Nonetheless, the result again showed an increased MAPE of 3.08%. After that, a different angle was tried – increasing the number of neurons in the first and second layers to 256. The hypothesis behind increasing the number of neurons relied on the idea that this change should enhance the model's capacity to learn and represent more complex patterns and capture long-term dependencies. The outcome showed a decrease of 0.53pp in MAPE to 1.76% which is a significant improvement. Following this logic a model with 512 neurons in each layer was run, however resulting in a slightly worse MAPE of 1.83%. The train time for this model was the longest of all LSTM models running for more than 6.5 hours becoming a result inefficient attempt. As 256 neurons seemed to be the most optimal number of neurons it was decided to stick with this option in the upcoming trials.

Unfortunately, neither the changing the batch size to 16, nor to 64, nor changing the number of epochs to 100 or to 25 brought any improvements to the MAPE of 1.76%. Finally, it was decided to change the number of days for used as a look back window from 30 to 60 to see if this would provide better results as the model would have more days and might capture more complex patterns in how the stock's price moves. However, increasing the look back window produced a worse MAPE of 2.16%. Then contrary to increasing the number of look back days, it was reduced to 10. The lower number of days could help the model capture the most recent changes and trends in stock's price. The outcome supported this hypothesis as MAPE was 1.74%, however only by 0.02pp better than the one using 30 days look back window. This difference could easily be interpreted as the noise in neural network forecasting; hence it was chosen to decrease the look back window even more – to only 5 days. This change yielded the best results for LSTM model with a MAPE of 1.69%. Additionally, it took just over 24 minutes to train the models for all 20 stocks which was the best result observed from all LSTM models. For none of the stocks model required more than 2 minutes to train, for 7 out of 20, the model needed less than a minute to train. The LSTM model had the best MAPE value of 0.67% for "Tallina Vesi AS" while the worst error term was for "Pro Kapital Grupp AS" with almost four percent. A correlation between training time and MAPE values was calculated to check if longer training times cause lower MAPE values, however an output of 0.09 shows negligible positive correlation, though for the statement to be true there should have been a strong negative correlation observed. Nonetheless, even the best LSTM model could not outperform a simple 10-day MA model, let alone 5- or 3-day MA models in terms of the mean absolute percentage error.

	30 days look back window											
	Neurons in 1st and 2nd layer: 128, epochs 50, batch size 32		Neurons in 1st and 2nd layer: 128, epochs 50, batch size 16		Neurons in 1st and 2nd layer: 128, epochs 100, batch size 32		Neurons in 1st and 2nd layer: 256, epochs 50, batch size 32		Neurons in 1st and 2nd layer: 512, epochs 50, batch size 32		Neurons in 1st and 2nd layer: 256, epochs 50, batch size 16	
Ticker	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**
IGN1L.VS	1,08	0,58	0,52	0,65	2,10	0,38	3,42	0,75	9,70	0,49	3,19	0,39
TAL1T.TL	4,09	7,71	1,64	7,41	7,57	8,81	9,17	4,44	31,13	1,60	12,24	7,94
EGR1T.TL	0,88	1,25	0,43	1,12	1,56	1,12	1,69	1,20	5,49	1,74	2,81	1,01
SAB1L.VS	4,42	1,45	1,87	3,80	4,95	1,53	8,73	1,61	25,52	3,11	12,56	3,23
LHV1T.TL	4,17	5,20	1,64	7,23	4,54	9,97	6,07	0,94	19,41	3,72	8,87	4,90
TKM1T.TL	4,87	0,76	2,12	0,53	9,10	0,67	6,98	1,20	26,28	0,71	11,02	0,86
TEL1L.VS	2,47	0,51	1,85	2,06	8,55	2,20	7,98	3,43	24,46	0,55	10,29	1,00
TSM1T.TL	1,80	3,62	0,90	5,41	5,91	5,42	3,72	1,89	14,03	1,90	5,85	5,74
CPA1T.TL	1,55	1,39	0,89	4,79	4,65	0,71	2,76	0,94	9,94	1,82	3,82	1,78
MRK1T.TL	3,03	2,31	1,80	3,35	12,55	0,96	7,48	1,64	25,39	1,69	10,56	1,69
TVE1T.TL	2,41	0,56	1,62	0,59	5,41	0,65	6,94	0,53	23,54	0,53	7,36	0,77
HAE1T.TL	1,98	0,93	1,61	1,05	11,21	1,00	5,20	1,14	23,84	1,00	7,57	0,85
GRG1L.VS	3,54	1,08	1,64	1,14	9,50	2,16	5,99	1,12	24,13	1,48	8,01	1,15
AKO1L.VS	4,34	1,42	1,48	4,61	4,24	6,52	7,33	1,99	24,53	2,59	8,71	10,91
APG1L.VS	1,87	1,55	1,63	1,07	7,24	1,79	7,97	1,48	21,19	1,31	9,10	1,05
PKG1T.TL	2,29	7,58	1,68	7,60	8,63	9,02	8,82	5,48	23,15	5,80	10,00	8,21
VLP1L.VS	3,25	1,01	1,90	1,88	12,76	2,49	8,92	0,94	25,73	2,19	11,67	2,49
NTU1L.VS	1,34	2,56	1,30	5,99	5,84	2,23	4,13	1,48	13,76	1,13	5,95	1,68
AUG1L.VS	1,30	1,08	1,94	0,79	3,89	0,81	6,94	0,80	22,03	1,13	9,20	0,95
HPR1T.TL	0,30	3,31	0,31	2,17	1,16	3,22	1,37	2,23	3,73	2,09	1,83	1,66
	50,97	**2,29**	**28,76**	**3,16**	**131,37**	**3,08**	**121,63**	**1,76**	**396,99**	**1,83**	**160,59**	**2,91**
	Total	**Average**	**Total**	**Average**	**Total**	**Average**	**Total**	**Average**	**Total**	**Average**	**Total**	**Average**

Table 5. Summary results of LSTM models.

	30 days look back window						60 days look back window		10 days look back window		5 days look back window	
	Neurons in 1st and 2nd layer: 256, epochs 50, batch size 64		Neurons in 1st and 2nd layer: 256, epochs 100, batch size 32		Neurons in 1st and 2nd layer: 256, epochs 25, batch size 32		Neurons in 1st and 2nd layer: 256, epochs 50, batch size 32		Neurons in 1st and 2nd layer: 256, epochs 50, batch size 32		Neurons in 1st and 2nd layer: 256, epochs 50, batch size 32	
Ticker	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**	**Training time**	**MAPE**
IGN1L.VS	1,79	0,50	6,13	0,38	1,09	0,85	4,20	0,67	0,90	0,39	0,51	1,00
TAL1T.TL	6,47	1,49	20,86	8,16	4,31	1,85	14,24	3,73	3,49	4,07	1,77	3,25
EGR1T.TL	1,50	1,47	4,33	0,92	0,91	1,60	3,28	1,67	0,81	1,64	0,39	1,52
SAB1L.VS	6,97	3,18	19,52	1,28	4,23	5,54	13,34	1,21	3,10	2,04	1,38	1,13
LHV1T.TL	5,41	2,32	15,71	3,21	3,79	2,33	11,51	6,36	2,54	2,70	1,03	1,24
TKM1T.TL	7,11	1,04	20,24	0,96	5,01	0,73	14,61	1,09	3,70	0,92	1,55	1,26
TEL1L.VS	5,84	1,13	16,85	1,93	5,06	2,78	13,91	0,74	3,34	0,59	0,99	0,89
TSM1T.TL	3,51	2,78	10,22	1,44	2,33	1,88	8,21	3,27	2,09	2,49	0,70	2,05
CPA1T.TL	2,77	1,16	7,96	1,93	1,71	2,05	5,81	1,10	1,62	3,08	0,61	2,80
MRK1T.TL	7,29	1,23	18,98	0,95	4,04	1,03	18,30	0,81	3,63	1,03	1,68	1,62
TVE1T.TL	7,05	0,61	20,17	0,57	4,40	0,72	18,52	0,66	3,58	0,52	1,42	0,67
HAE1T.TL	6,14	1,32	17,83	1,44	4,85	1,67	15,99	1,57	3,53	1,07	1,64	0,94
GRG1L.VS	5,88	1,54	18,03	1,67	5,13	1,26	15,51	1,19	4,10	1,17	1,61	1,73
AKO1L.VS	5,60	3,76	16,43	3,83	4,62	5,19	12,78	4,74	3,51	1,69	1,45	0,98
APG1L.VS	6,21	1,65	20,20	1,21	5,64	1,66	12,99	1,20	3,86	1,27	1,62	1,09
PKG1T.TL	5,98	4,38	18,79	8,55	5,48	6,79	14,54	6,38	4,12	3,95	1,74	3,96
VLP1L.VS	6,24	1,29	19,36	0,96	5,27	2,51	13,38	1,60	3,60	0,93	1,59	2,28
NTU1L.VS	4,13	2,41	8,85	3,27	3,15	2,08	7,21	1,62	2,33	1,33	0,93	2,99
AUG1L.VS	5,54	1,49	19,35	0,97	3,77	1,06	14,58	0,89	3,36	1,22	1,29	0,82
HPR1T.TL	1,15	2,59	3,79	1,45	0,88	2,02	2,77	2,65	0,71	2,79	0,36	1,60
	102,59	**1,87**	**303,59**	**2,25**	**75,70**	**2,28**	**235,69**	**2,16**	**57,93**	**1,74**	**24,27**	**1,69**
	Total	**Average**	**Total**	**Average**	**Total**	**Average**	**Total**	**Average**	**Total**	**Average**	**Total**	**Average**

Table 6. Summary results of LSTM models.

The training time for the LSTM with transformers model using the same parameters as the best LSTM model took almost 76 minutes in total to train or on average close to 4 minutes per stock – almost 3 times longer when compared to the LSTM model. A correlation between training time and MAPE was calculated to see if there was any relationship between longer training time and lower error term, however a result of -0.26 proved that only weak negative correlation exists. Analyzing MAPE values for transformers models further, it was obvious that this alternative is less stable as MAPE values range from 0.56% to 40% compared to general LSTM MAPE range of 0.67% to 3.96%. Even though that for majority of shares the error term was higher than the one observed in general LSTM, there were four stocks where LSTM with transformers model managed to increase forecasting accuracy by reducing MAPE by 0.43pp for “Ignitis Grupe AB”, 1.93pp for “Coop Pank AS”, 0.02pp for “Tallina Vesi AS”, and 0.28pp for “Pro Kapital Grupp AS” which had the highest MAPE under general LSTM model. However, the stock price predictions require continuous stability and error terms to be under appropriate level of variance, hence LSTM with transformers with current parameter would not qualify as a good approach for stock price forecasting. The findings coincide with similar doubts about transformers’ stability and predictability are shared in an article by May (2023) where they as well observed that though both models had good MAPE values, the one that used transformers had greater range of the error term then the general LSTM model.

Ticker	Time to train LSTM_T	MAE	RMSE	MAPE
IGN1L.VS	1,14	0,11	0,15	0,56
TAL1T.TL	5,29	0,03	0,04	5,10
EGR1T.TL	1,10	0,10	0,11	2,59
SAB1L.VS	4,90	0,06	0,07	9,51
LHV1T.TL	3,74	0,45	0,47	12,79
TKM1T.TL	4,98	0,17	0,20	1,72
TEL1L.VS	4,68	0,30	0,31	15,37
TSM1T.TL	2,68	0,12	0,13	9,16
CPA1T.TL	2,18	0,02	0,03	0,88
MRK1T.TL	4,33	0,99	1,07	6,55
TVE1T.TL	4,50	0,08	0,11	0,65
HAE1T.TL	4,91	0,17	0,22	2,98
GRG1L.VS	5,02	0,05	0,06	5,46
AKO1L.VS	4,13	0,36	0,40	27,27
APG1L.VS	4,66	0,04	0,06	1,64
PKG1T.TL	4,82	0,03	0,04	3,69
VLP1L.VS	4,59	0,33	0,39	7,02
NTU1L.VS	2,38	0,19	0,20	5,97
AUG1L.VS	4,77	0,00	0,01	1,08
HPR1T.TL	1,01	3,03	3,24	40,47

Table 7. Summary table of LSTM with transformers training times and error terms.

There were 6 different machine learning models (random forest base, random forest with random search, random forest with Bayesian search, ARIMA, LSTM, and LSTM with transformers) created and tested for predicting prices of the constituents of the OMXBBGI index. While some of them performed quite poorly with an average MAPE reaching over 25% in case of random forest models, others like ARIMA or general LSTM managed to prove their abilities to successfully forecast stock prices. However, in relation to providing the lowest error terms, ARIMA model demonstrated its superiority by having smaller MAPE errors for all stocks with an average of 0.68% compared to 1.69% of the LSTM model. Hence, for further portfolio optimization task the ARIMA model will be employed.

3.2. Portfolio Optimization Results

Markowitz's Modern Portfolio Theory was integrated into the forecasts provided by the ARIMA model for portfolio optimization and find the best asset allocation to maximize returns at a given risk level. It was determined as the daily volatility for the portfolio which was calculated to be 2.63%. The optimization solution worked through every business day for December 2023 to calculate the maximum return for the given risk level. While the model for the majority of days favored always 4 stocks (see Figure 6) allocating the largest weights to them, it did not violate constraint of daily volatility for the constructed portfolio. Though it might not seem like a very balanced portfolio, however if an investor is seeking a portfolio that can replicate the same risk as the benchmark index, this simulated portfolio required only 4 stocks to do that.

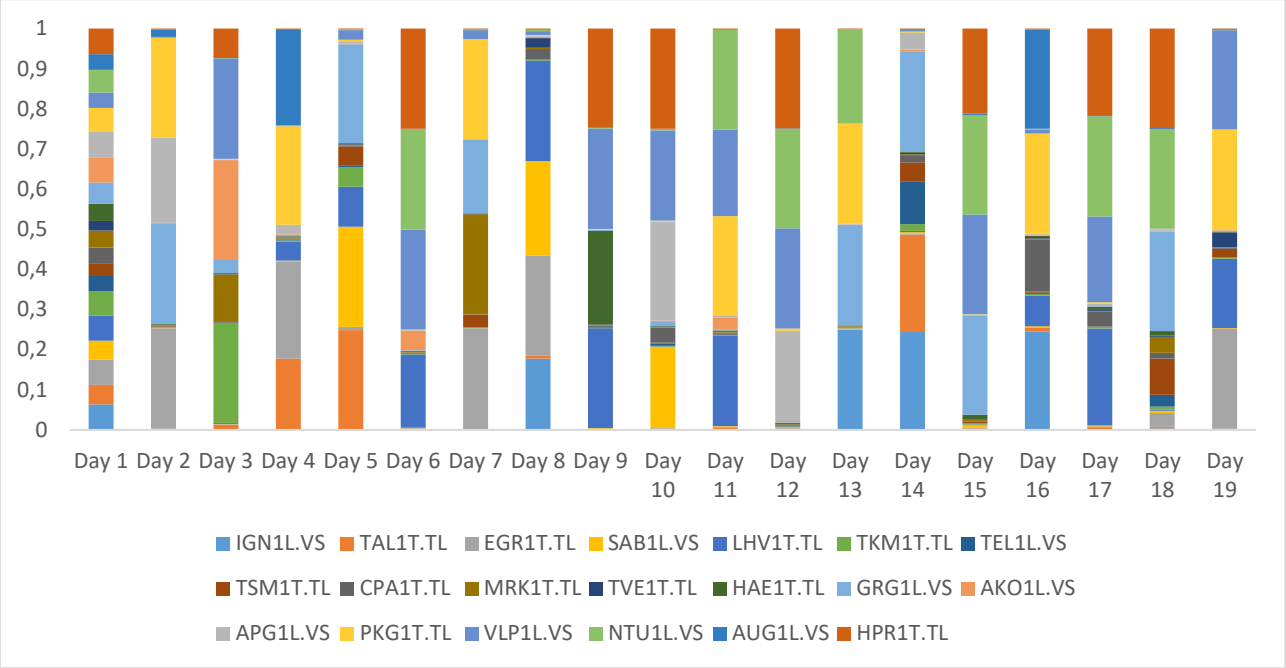


Figure 5. Daily stock weights for optimal portfolio.

The total value of the portfolio with daily rebalances at the end of the month was equal to 1015.15 EUR, yielding a monthly return of 1.515%. In comparison if the market index was bought in the beginning of December 2023 with 1000 EUR, at the end of the month the value of such portfolio would have been equal to 1002.51 EUR giving 0.251% return on the month. The daily portfolio value movements are displayed in Figure 7. This shows that optimized portfolio with daily rebalancing strategy can outperform a benchmark index.

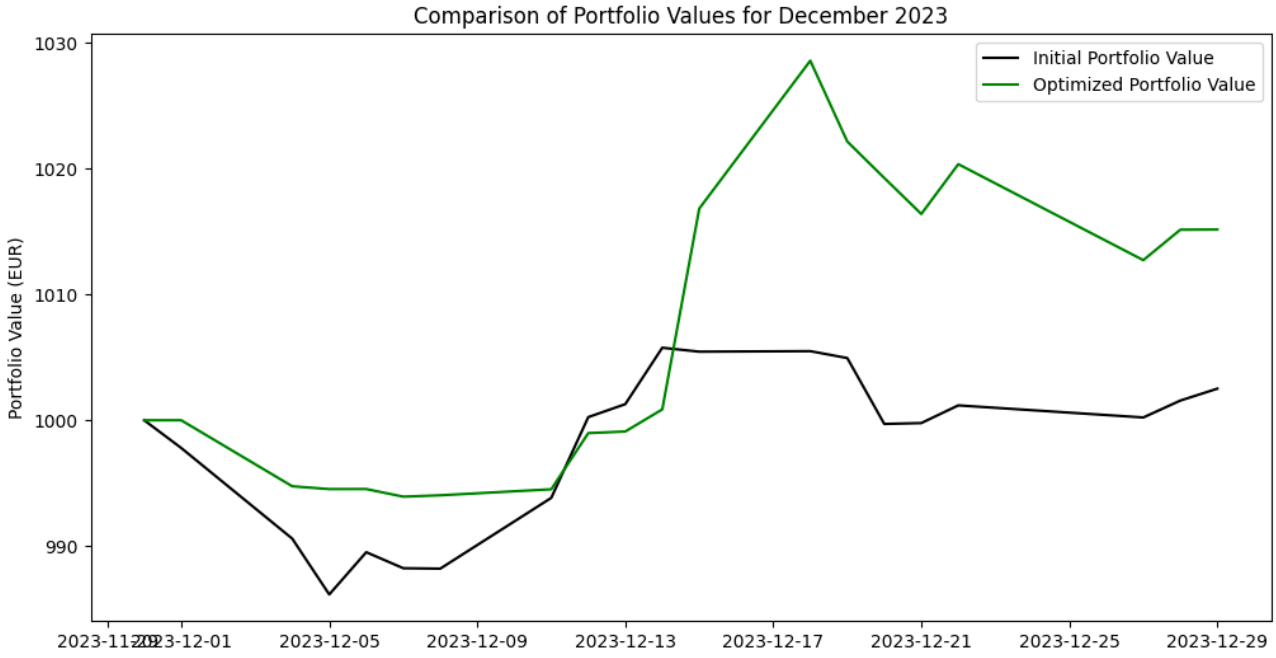


Figure 6. Comparison of Portfolio Values for December 2023.

Additionally, annualized Sharpe ratio is calculated for the optimal portfolio and the benchmark index. The former with annualized Sharpe ratio of 2.02 is significantly more efficient in terms of risk-adjusted returns compared to the benchmark one with a Sharpe Ratio of 0.1. It indicates that the

investment is providing a significantly higher return compared to the risk taken. The risk-free rate of return was used to be 3% as the current 1-year return for a term deposit. Another metric calculated to measure portfolio's efficiency was Jensen's Alpha. The results obtained from the calculations were equal to 0.0108 which means, that over the analyzed time, the investment has produced a return that is 1.08% (annualized) more than the expected return projected by its risk level (as indicated by its beta relative to the market). One more indicator on optimal portfolio to measure its efficiency was calculated Value at Risk (VaR) indicator. The output provides the highest anticipated loss over a 24-hour period that, 95% of the time, won't be exceeded. The results of 1-day VaR at 95% confidence interval for optimal portfolio showed a 0.75% loss while for the benchmark index it was 0.72%. In terms of VaR metric, the optimal portfolio would seem to be slightly worse off compared to the benchmark, however the difference is rather small and probably not impactful.

The newly created portfolio demonstrated its superiority over the market index while keeping the same level of risk. However, performing daily model retraining and stock reweighting might be a challenging task. The limitation of predicting only day ahead price leads to at least a two-fold problem. First of all, running intense machine learning models might take a lot of time and machine learning itself is a resource-intensive task that requires increased power consumption for efficiently completing the task. Even though the model might have a very high accuracy in predicting the next day's price, the overall change in the portfolio weights according to the forecasted prices might result in a minor change in the portfolio's value. This leads to questioning whether daily price forecasting is really that profitable, given the time and power consumed for the retraining of ML models. Another part of the problem from forecasting the next day's prices stems from the ability to quickly capitalize on gains. Since models employed here do not consider transaction costs because some of the Baltic banks allow buying and selling Baltic stocks for 0 commission, the process itself might not be as fast as trading on other brokerage firms. Hence, this could neglect all the benefits of machine learning for predicting only one day ahead prices. Instead, predicting the price of 5 or 10 days ahead might be more beneficial and easier implemented in realistic trading conditions. Finally, the weights of the optimal portfolio and return were calculated only for one month, whereas calculating for longer periods might reveal additional drawbacks to the optimization method proposed.

Conclusions

This thesis investigated the use of machine learning algorithms, such as LSTM, ARIMA, and RF, to forecast stock prices. Through the integration of these predictive models with Modern Portfolio Theory, the goal was to create investment strategy that maximizes returns while maintaining a predetermined degree of risk.

The results of optimized portfolio proved that such portfolio might be superior to the benchmark index with the total return over one month's period of 1.515% compared to a passive "buy and hold" strategy's result of 0.251%. This finding helped answering the research question of this thesis that indeed an optimal portfolio created using machine learning algorithms for stock price predictions can surpass the OMXBBGI index in terms of generating greater returns with the same level of risk. In addition to returns and standard deviations, metrics such as Sharpe Ratio, Jensen's Alpha, and Value at Risk were calculated providing useful information about the optimal portfolio constructed.

The study did, however, have a few imperfections leaving it to future studies to build upon. For instance, the random forest models could benefit more from feature engineering and introduction of technical indicators to improve its forecasting accuracy. More complex LSTM features could be introduced as well to help boost the accuracy as in other studies the LSTM models often proved to be significantly more accurate than ARIMA or other benchmark models. Another global limitation of the experiment which could be analyzed in later studies is forecasting not one single day at the time, but 5 or 10 days in the future.

In summary, this thesis emphasizes the potential of machine learning algorithms to improve portfolio management and stock market analytics, but it also emphasizes the difficulties and complexities involved in putting these technologies into practice. Nonetheless, the study managed to portray that even when implementing comparatively simple stock price forecasting models and a straightforward portfolio optimization method, the simulated portfolio's results are better compared to the benchmark index.

List of references

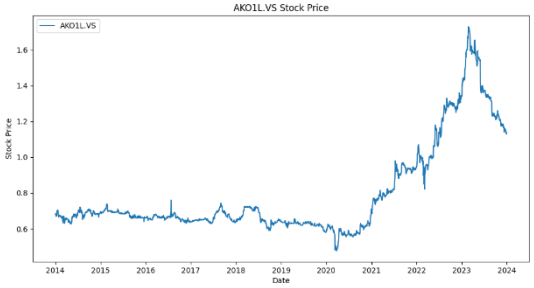
1. Adebiyi, A. A., Adediran, A., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, *2014*, 1–7. https://doi.org/10.1155/2014/614342
2. Almaafi, A., Bajaba, S., & Alnori, F. (2023). Stock price prediction using ARIMA versus XGBoost models: the case of the largest telecommunication company in the Middle East. *International Journal of Information Technology*, *15*(4), 1813–1818. https://doi.org/10.1007/s41870-023-01260-4
3. Banton, C. (2022, July 9). *What does standard deviation measure in a portfolio?* Investopedia. https://www.investopedia.com/ask/answers/022015/what-does-standard-deviation-measure-portfolio.asp
4. Black, F., & Litterman, R. (1991). Asset equilibrium: Combining investor views with market equilibrium. *Journal of Fixed Income*, 1, 7–18.
5. Black, F., & Litterman, R. (1992). Global portfolio optimization. *Financial Analysts Journal*, 48, 28–43.
6. Brinză, A., Ioan, V., & Lazarescu, I. (2023). Critical Analysis of the Sharpe Ratio: Assessing Performance and Risk in Financial Portfolio Management. *Ovidius University Annals, Economic Sciences Series, 23*(2).
7. Chen, J. (2022, January 1). *Risk Parity: Definition, Strategies, example*. Investopedia. https://www.investopedia.com/terms/r/risk-parity.asp#:~:text=The%20risk%20parity%20approach%20builds,advanced%20than%20simplified%20allocation%20strategies.
8. Chen, W., Zhang, H., Mehlawat, M. K., & Jia, L. (2021). Mean–variance portfolio optimization using machine learning-based stock price prediction. *Applied Soft Computing*, 100, 106943.
9. Chiang, W. C., Enke, D., Wu, T., & Wang, R. (2016). An adaptive stock index trading decision support system. *Expert Systems with Applications*, 59, 195-207.
10. Chung, H., & Shin, K. S. (2018). Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability*, *10*(10), 3765.
11. Conlon, T., Cotter, J., & Kynigakis, I. (2021). Machine learning and factor-based portfolio optimization. *arXiv preprint arXiv*:2107.13866.
12. Cura, T. (2009). Particle swarm optimization approach to portfolio optimization. *Nonlinear analysis: Real world applications*, 10(4), 2396-2406.
13. Dubinskas, P., & Urbšienė, L. (2017). Investment portfolio optimization by applying a genetic algorithm-based approach. *Ekonomika*, *96*, 66-78.
14. Edwards, J. (2022, January 22). *How to create a risk parity Portfolio*. Investopedia. https://www.investopedia.com/articles/active-trading/091715/how-create-risk-parity-portfolio.asp.
15. *ElectrifAi | Portfolio Optimization with Machine Learning*. (n.d.). https://www.electrifai.com/blog/portfolio-optimization-with-machine-learning.
16. Fischer, T., & Krauß, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

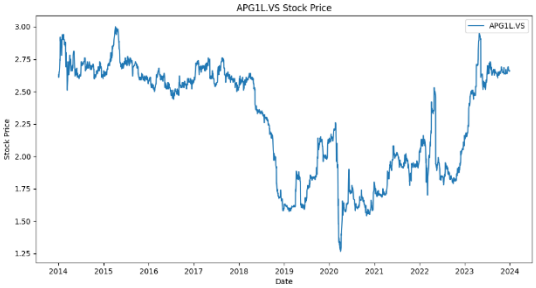
17. Funde, Y., & Damani, A. (2023). Comparison of ARIMA and exponential smoothing models in prediction of stock prices. *The Journal of Prediction Markets*, *17*(1), 21–38. https://doi.org/10.5750/jpm.v17i1.2017
18. Green, R. C., & Hollifield, B. (1992). When will mean-variance efficient portfolios be well diversified?. *The Journal of Finance*, *47*(5), 1785-1809.
19. Grogan, M. (2021, December 15). Limitations of ARIMA: Dealing with Outliers - Towards Data Science. *Medium*. https://towardsdatascience.com/limitations-of-arima-dealing-with-outliers-30cc0c6ddf33
20. Hayes, A. (2024, April 6). *Autoregressive Integrated Moving Average (ARIMA) Prediction Model*. Investopedia. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp
21. Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, *124*, 226-251.
22. *Investuotojo portretas: aktyviausiai investuoja 33-44 metų žmonės*. (2023, March 31). Lietuvos Bankas. Retrieved May 5, 2024, from https://www.lb.lt/lt/naujienos/investuotojo-portretas-aktyviausiai-investuoja-35-44-metu-zmones#:~:text=Investuotojų%20skaičius%20Lietuvoje%202022%20m,aktyviai%20investavo%2034%2C4%20tūkst.
23. Jobson, J. D., & Korkie, B. (1984). On the Jensen Measure and Marginal Improvements in Portfolio Performance: A Note. *The Journal of Finance, 39*(1), 245-251.
24. Jurkonytė-Dumbliauskienė, E., & Paužuolis, V. (2015). Moderniosios, postmoderniosios portfelio teorijų ir Black-Litterman modelio palyginimas. *Mokslo taikomieji tyrimai Lietuvos kolegijose*, *1*(11), 96-102.
25. Kenton, W. (2023, December 1). *Black-Litterman Model: Definition, Basics, and example*. Investopedia. https://www.investopedia.com/terms/b/black-litterman_model.asp
26. Kenton, W. (2024, February 19). *Understanding value at risk (VAR) and how it's computed*. Investopedia. https://www.investopedia.com/terms/v/var.asp#:~:text=Value%20at%20risk%20(VaR)%20is%20a%20measure%20of%20the%20potential,much%20returns%20vary%20over%20time.
27. Kolm, P. N., Tütüncü, R., & Fabozzi, F. J. (2014). 60 Years of portfolio optimization: Practical challenges and current trends. *European Journal of Operational Research*, *234*(2), 356-371.
28. Kotu, V., & Deshpande, B. (2019). Time series forecasting. In *Elsevier eBooks* (pp. 395–445). https://doi.org/10.1016/b978-0-12-814761-0.00012-5
29. Leković, M. (2017). Mutual funds portfolio performance evaluation models: Sharpe, Treynor and Jensen index. *Bankarstvo*, *46*(4), 108-120.
30. Levy, M. (2017). Measuring Portfolio Performance: Sharpe, Alpha, or the Geometric Mean? Journal of Investment Management, 15(3), 6-22.
31. Lydenberg, S. (2016). Integrating Systemic Risk into Modern Portfolio Theory and Practice. *The Bank of America Journal of Applied Corporate Finance*, *28*(2), 56–61. https://doi.org/10.1111/jacf.12175
32. Ma, Q. (2020). Comparison of ARIMA, ANN and LSTM for stock price prediction. *E3S Web of Conferences*, *218*, 01026. https://doi.org/10.1051/e3sconf/202021801026

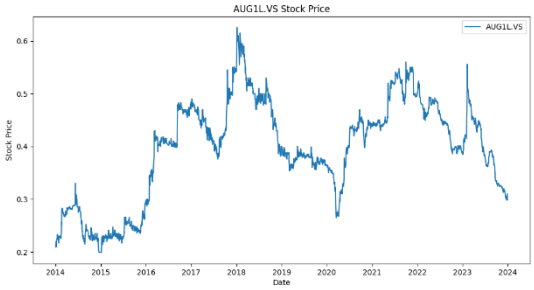
33. Ma, Y., Han, R., & Wang, W. (2021). Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications, 165*, 113973.
34. Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*, *7*(1), 77–91. https://doi.org/10.2307/2975974
35. May, M. (2023, July 19). Transformers vs. LSTM for Stock Price Time Series Prediction. *Medium*. https://medium.com/@mskmay66/transformers-vs-lstm-for-stock-price-time-series-prediction-3a26fcc1a782
36. McClure, B. (2022). Modern Portfolio Theory: Why It's Still Hip. *Investopedia*. https://www.investopedia.com/managing-wealth/modern-portfolio-theory-why-its-still-hip/
37. Mian, T. S. (2023). Evaluation of Stock Closing Prices using Transformer Learning. *Engineering, Technology and Applied Science Research/Engineering, Technology and Applied Science Research*, *13*(5), 11635–11642. https://doi.org/10.48084/etasr.6017
38. Moreno, J. J. M., Pol, A. L. P., Sesé, A., & Blasco, B. C. (2013). Using the R-MAPE index as a resistant measure of forecast accuracy. *PubMed*, *25*(4), 500–506. https://doi.org/10.7334/psicothema2013.23
39. Petrică, A., Stancu, S., & Tindeche, A. (2016). Limitation of ARIMA models in financial and monetary economics. *DOAJ (DOAJ: Directory of Open Access Journals)*. https://doaj.org/article/3b26dede3d584d9f8d3e52e0d0c5ae6e
40. Plikynas, D., Daniušis P., (2010), *Kompiuterinis intelektas ir daugiaagentės sistemos socialinių mokslų srityje*. Monografija. Vilnius: Verslo ir vadybos akademija.
41. Rom, B. M., & Ferguson, K. W. (1993). Post-modern portfolio theory comes of age. *The Journal of Investing*, *2*(4), 27-33.
42. Saxena, S. (2024, January 4). *What is LSTM? Introduction to Long Short-Term Memory*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/#
43. Stoilov, T., Stoilova, K., & Vladimirov, M. (2020). Analytical overview and applications of modified Black-Litterman model for portfolio optimization. *Cybernetics and Information Technologies*, 20(2), 30–49. https://doi.org/10.2478/cait-2020-0014.
44. Team, I. (2023, August 29). *Modern Portfolio Theory: What MPT is and how Investors use it*. Investopedia. https://www.investopedia.com/terms/m/modernportfoliotheory.asp
45. *The rise of retail investing | United Fintech*. (2021, November 15). United Fintech. https://unitedfintech.com/blog/the-rise-of-retail-investing/
46. Todoni, M. D. (2015). A post-modern portfolio management approach on CEE markets. *Procedia Economics and Finance*, *32*, 1362-1376.
47. Tola, V., Lillo, F., Gallegati, M., & Mantegna, R. N. (2008). Cluster analysis for portfolio optimization. *Journal of Economic Dynamics and Control*, 32(1), 235-258.
48. Wang, S. (2023). A stock price prediction method based on BILSTM and improved transformer. *IEEE Access*, *11*, 104211–104223. https://doi.org/10.1109/access.2023.3296308
49. *What is Random Forest? | IBM*. (n.d.). https://www.ibm.com/topics/random-forest
50. Zhang, Q., Qin, C., Zhang, Y., Bao, F., Zhang, C., & Лю, П. (2022). Transformer-based attention network for stock movement prediction. *Expert Systems With Applications*, *202*, 117239. https://doi.org/10.1016/j.eswa.2022.117239

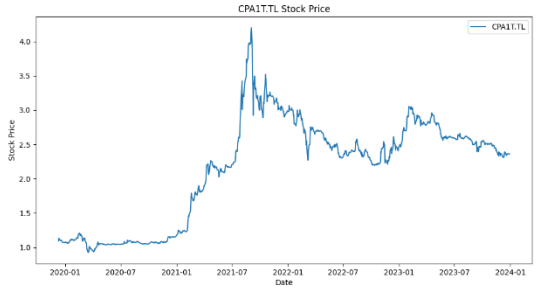
Appendices

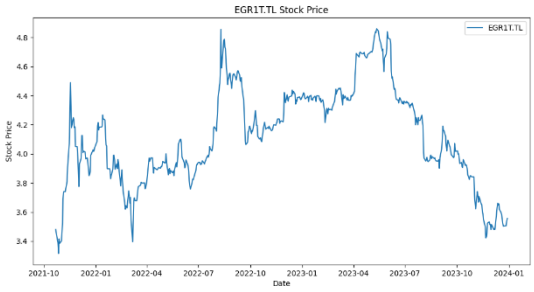
Appendix 1. Historical Prices of the Constituents of the OMXBBGI Index.

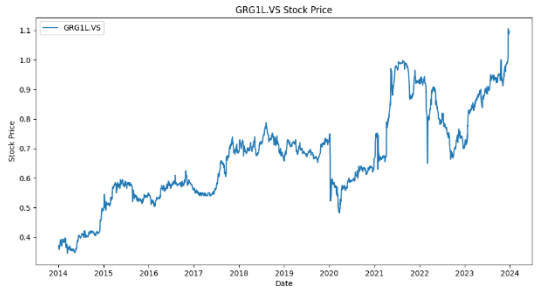


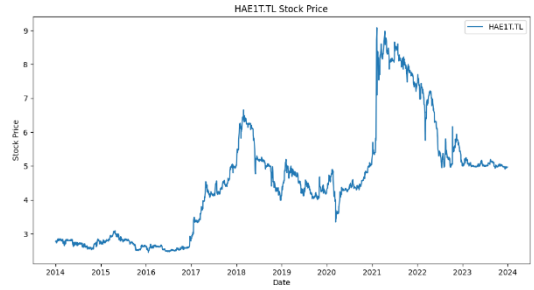


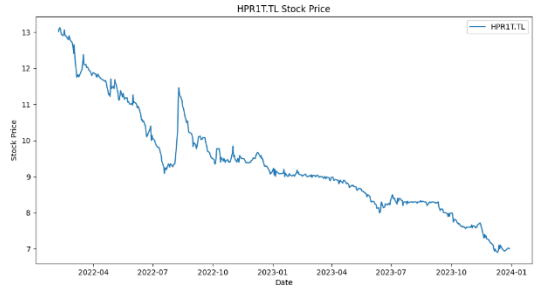


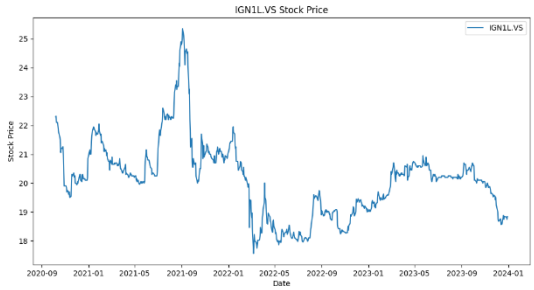


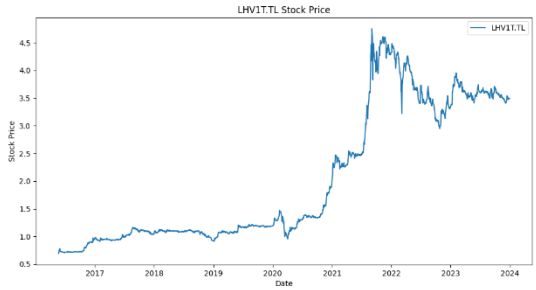


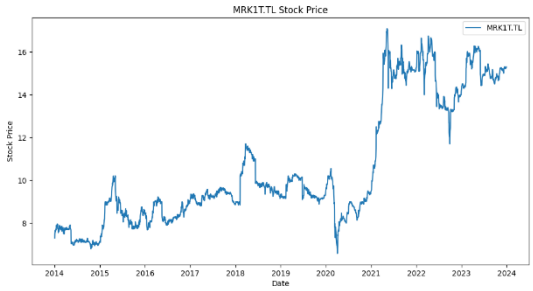


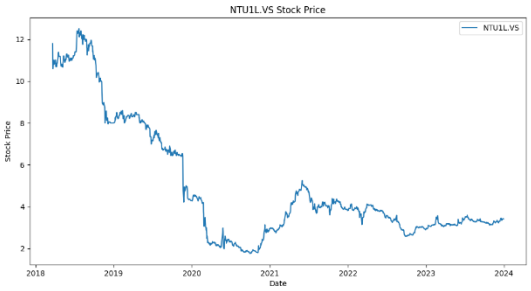


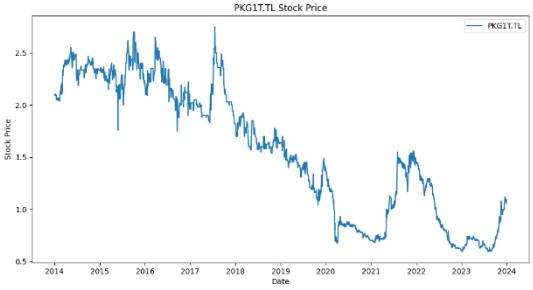


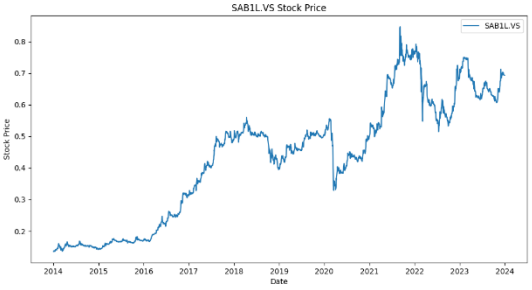


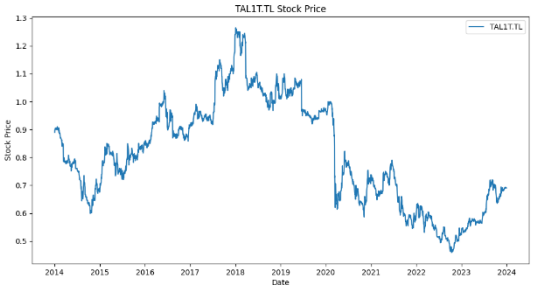


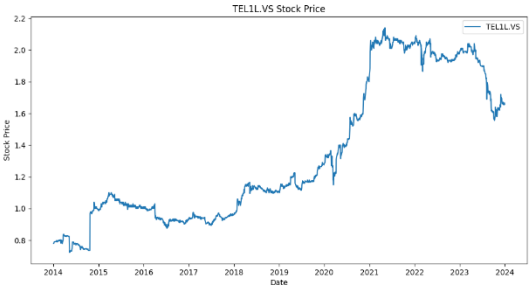

MRK1T.TL Stock Price
MRK1T.TL
Stock Price
Date

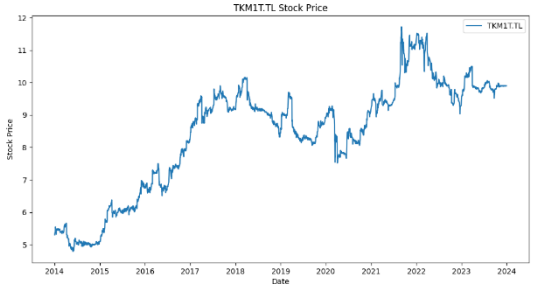

NTU1L.VS Stock Price
NTU1L.VS
Stock Price
Date

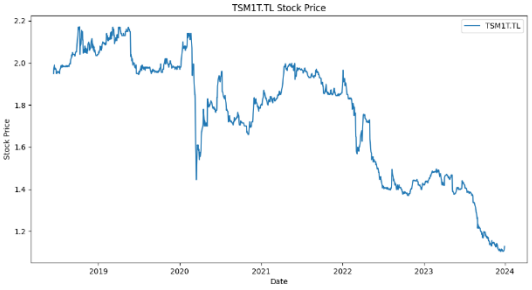

PKG1T.TL Stock Price
PKG1T.TL
Stock Price
Date

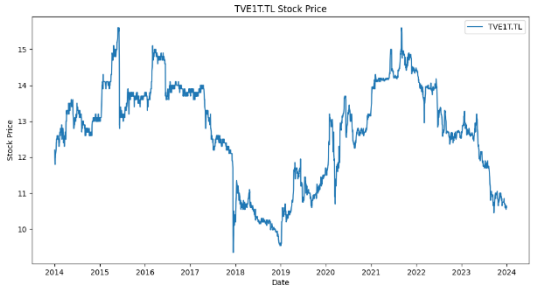

SAB1L.VS Stock Price
SAB1L.VS
Stock Price
Date

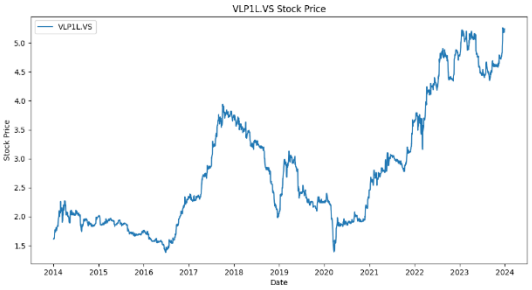

TAL1T.TL Stock Price
TAL1T.TL
Stock Price
Date


TEL1L.VS Stock Price
TEL1L.VS
Stock Price
Date


TKM1T.TL Stock Price
TKM1T.TL
Stock Price
Date


TSM1T.TL Stock Price
TSM1T.TL
Stock Price
Date

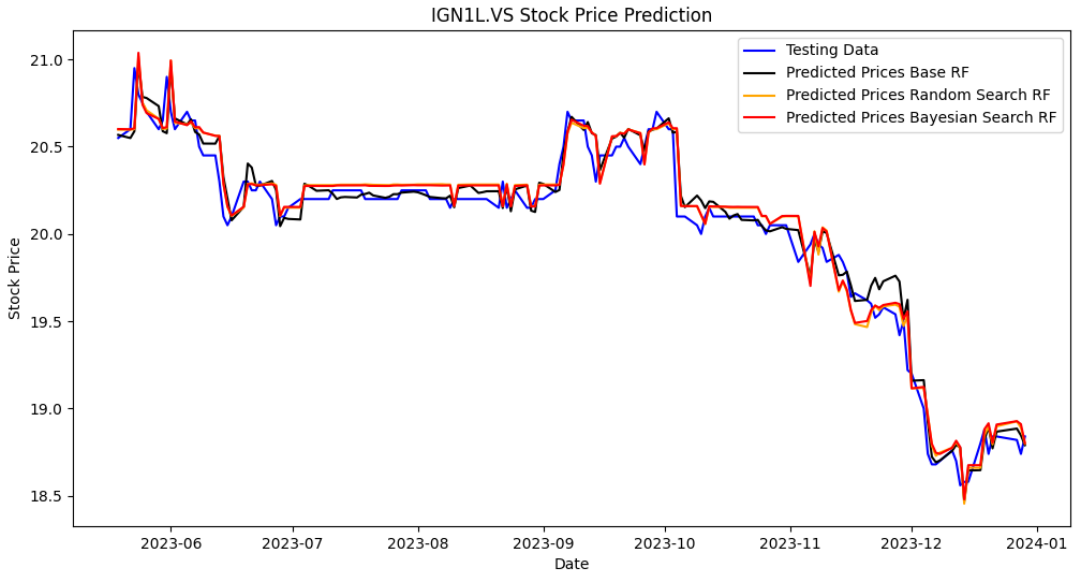

TVE1T.TL Stock Price
TVE1T.TL
Stock Price
Date

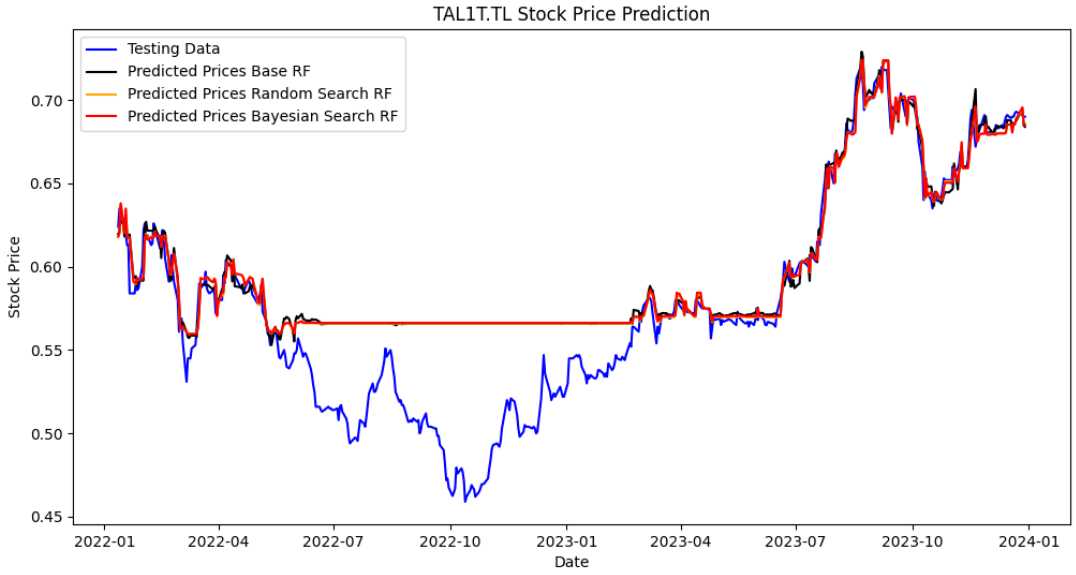

VLP1L.VS Stock Price
VLP1L.VS
Stock Price
Date

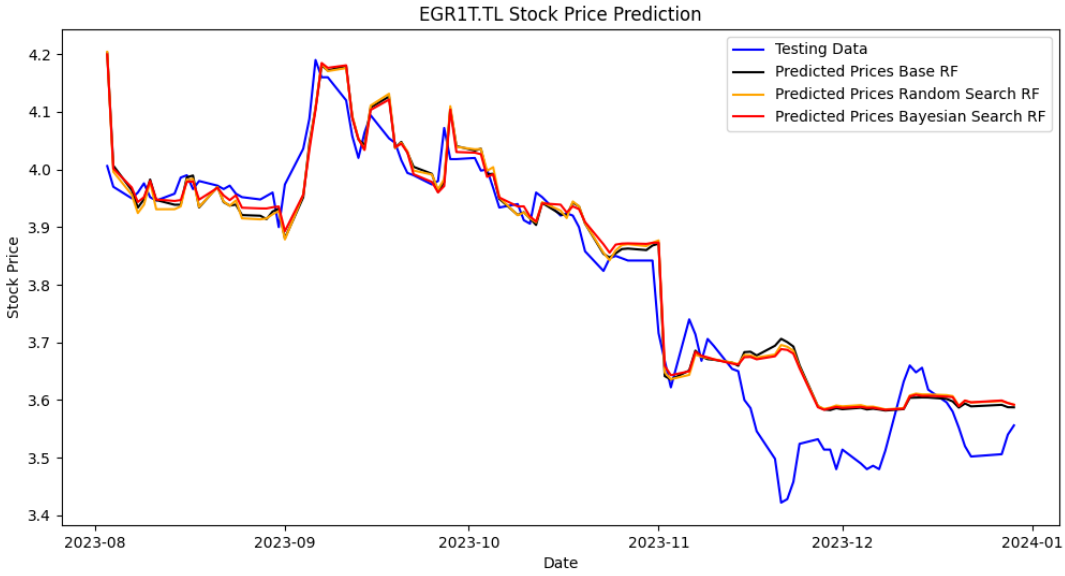
Appendix 2. Summary of Index Constituents.

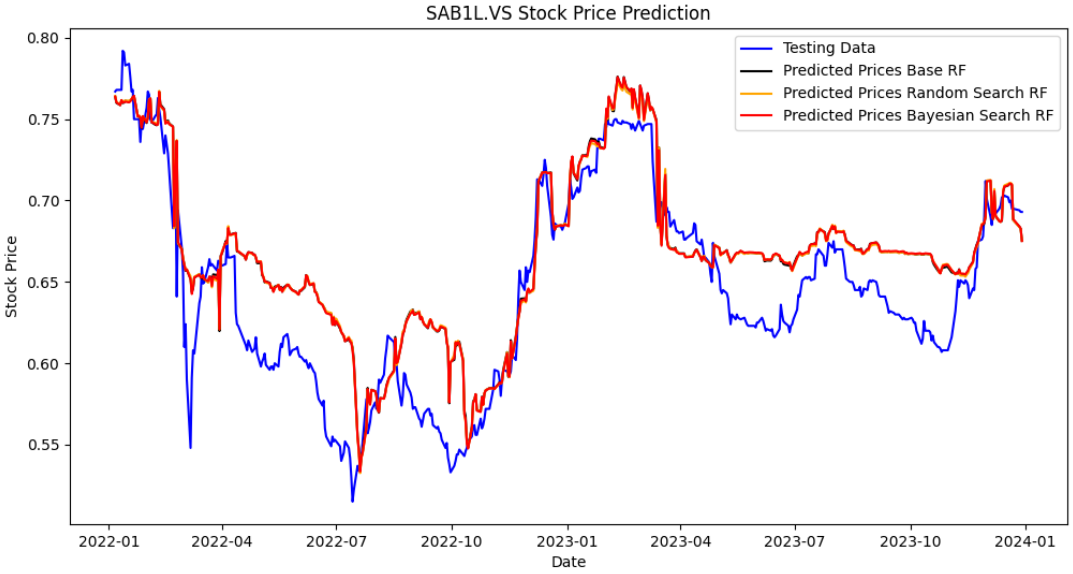
Name	Ticker	Weight	Annualized return [Close]	Based on
AB Akola Group	AKO1L.VS	0,98%	8,72%	10 years
APB Apranga	APG1L.VS	1,44%	-0,05%	10 years
Auga Group AB	AUG1L.VS	0,91%	10,77%	10 years
Coop Pank AS	CPA1T.TL	6,05%	30,33%	3,1 years
Enefit Green AS	EGR1T.TL	7,10%	21,31%	1,2 years
Grigeo AB	GRG1L.VS	2,09%	9,67%	10 years
AS Harju Elekter	HAE1T.TL	2,01%	6,62%	10 years
Hepsor AS	HPR1T.TL	0,20%	-33,06%	0,9 years
AB Ignitis grupe	IGN1L.VS	11,95%	-6,89%	2,2 years
AS LHV Group	LHV1T.TL	27,89%	26,88%	6,6 years
AS Merko Ehitus	MRK1T.TL	3,07%	9,16%	10 years
Novaturas AB	NTU1L.VS	0,48%	-25,44%	4,8 years
AS Pro Kapital Grupp	PKG1T.TL	1,89%	-11,31%	10 years
AB Siauliu Bankas	SAB1L.VS	10,30%	20,48%	10 years
AS Tallink Grupp	TAL1T.TL	8,10%	-4,92%	10 years
Telia Lietuva, AB	TEL1L.VS	4,25%	9,73%	10 years
TKM Grupp AS	TKM1T.TL	4,68%	5,51%	10 years
AS Tallinna Sadam	TSM1T.TL	3,77%	-6,68%	4,5 years
AS Tallinna Vesi	TVE1T.TL	1,91%	3,15%	10 years
AB Vilkyskiu pienine	VLP1L.VS	0,93%	14,39%	10 years

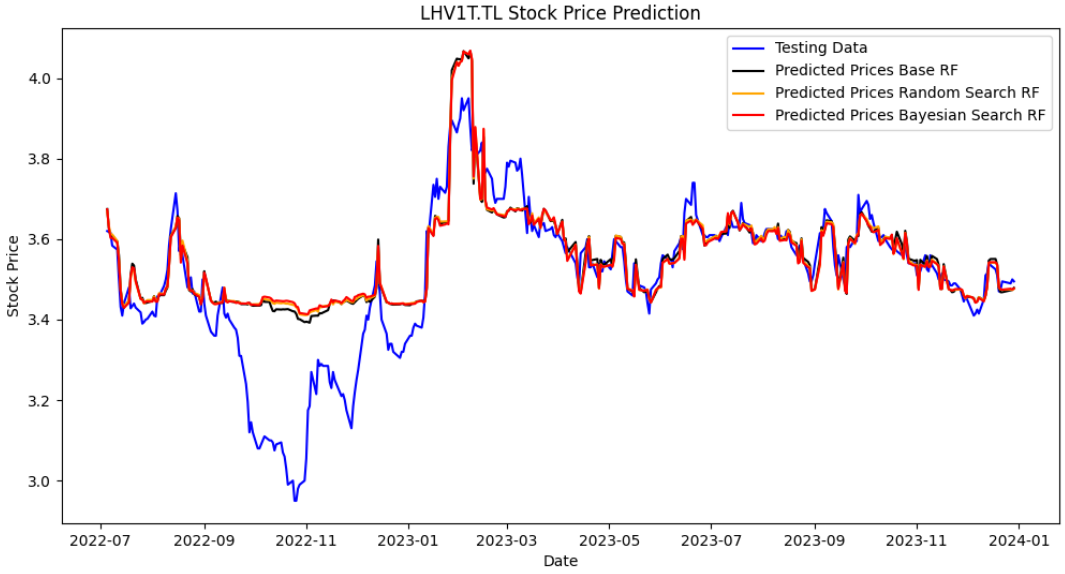
Appendix 3. Result Graphs of Random Forest, Random Forest with Random Search, Random Forest with Bayesian Search models.

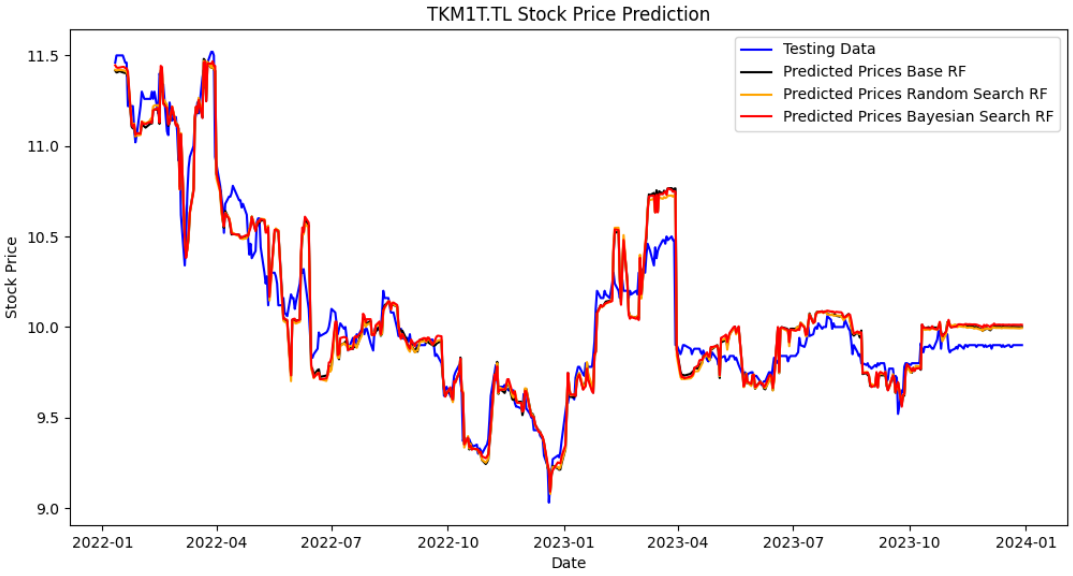


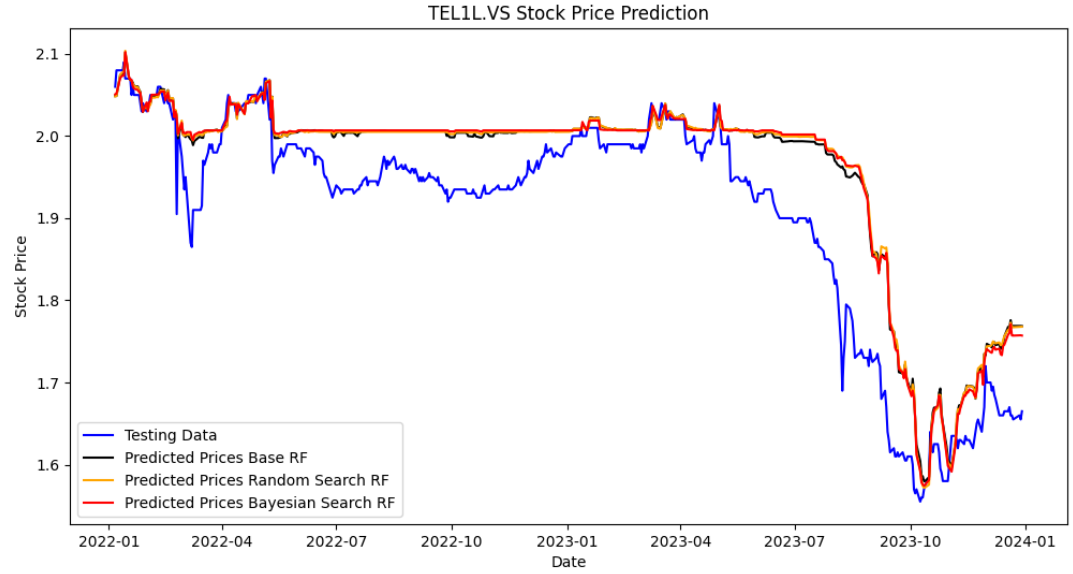


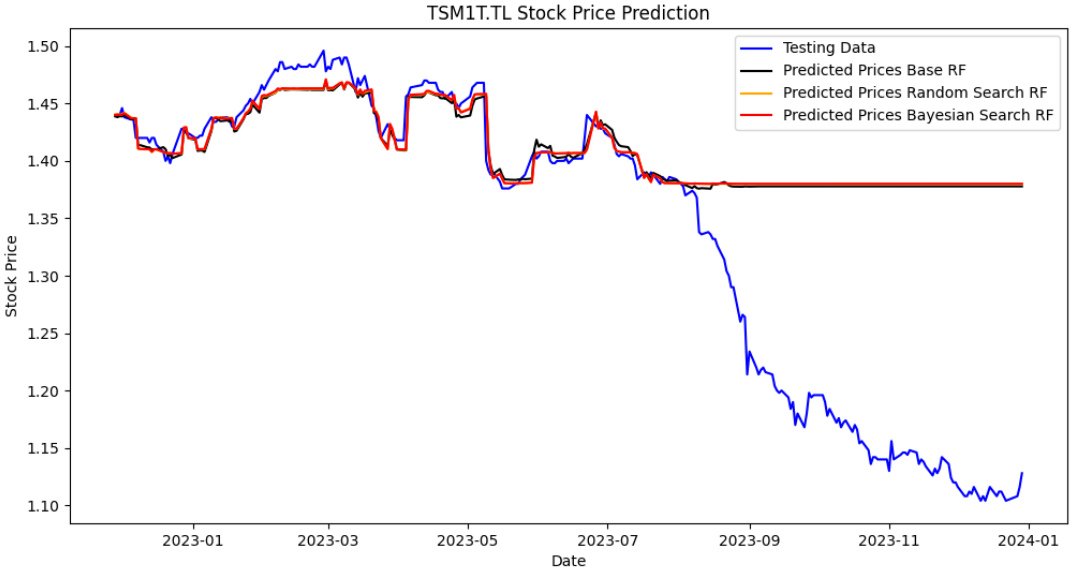

EGR1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
4.2
4.1
4.0
3.9
3.8
3.7
3.6
3.5
3.4
2023-08
2023-09
2023-10
2023-11
2023-12
2024-01

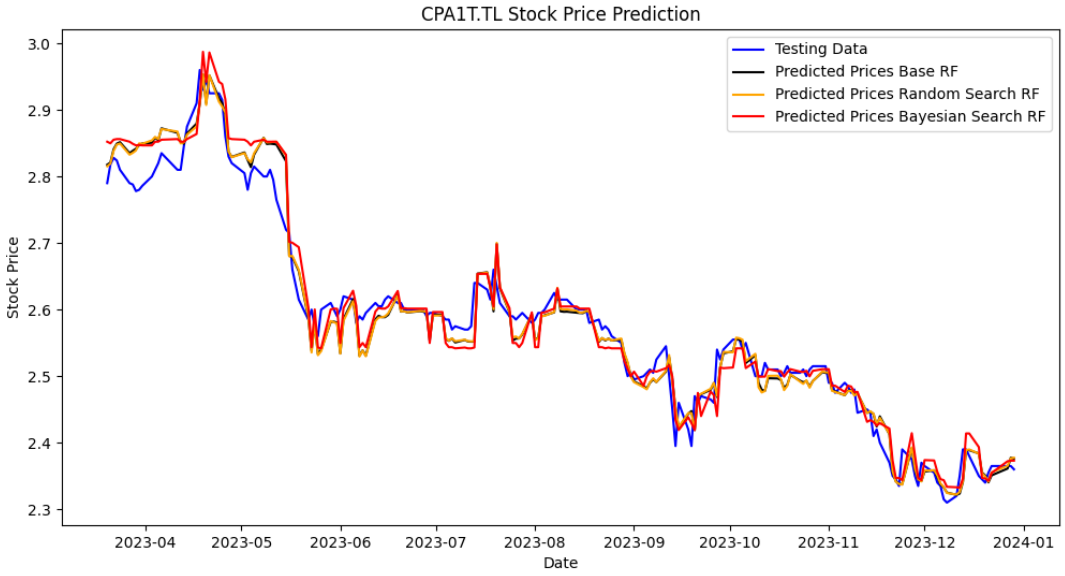

SAB1L.VS Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
0.80
0.75
0.70
0.65
0.60
0.55
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

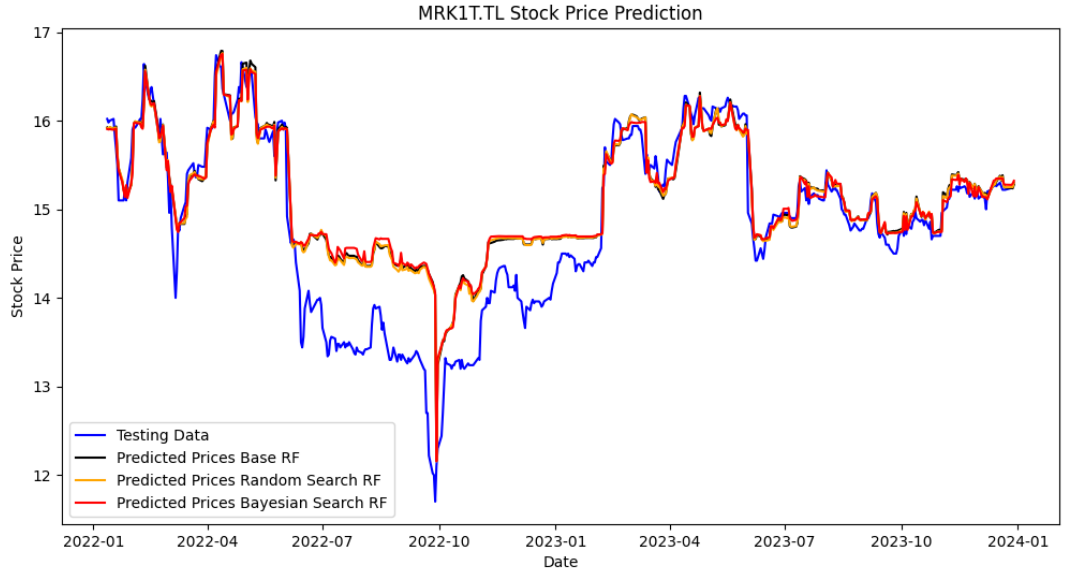

LHV1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
4.0
3.8
3.6
3.4
3.2
3.0
2022-07
2022-09
2022-11
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

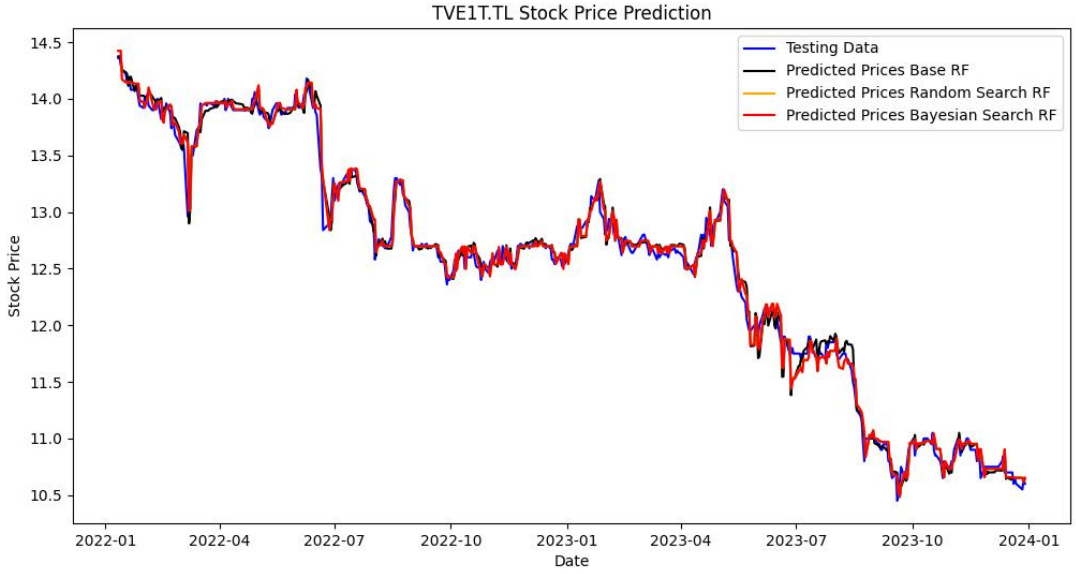

TKM1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
11.5
11.0
10.5
10.0
9.5
9.0
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

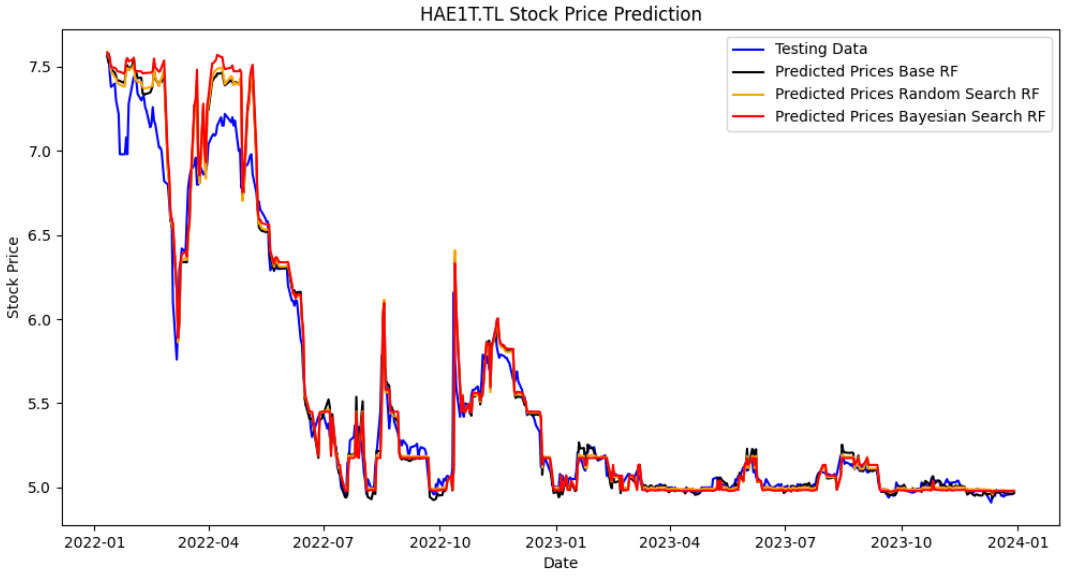

TEL1L.VS Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
2.1
2.0
1.9
1.8
1.7
1.6
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

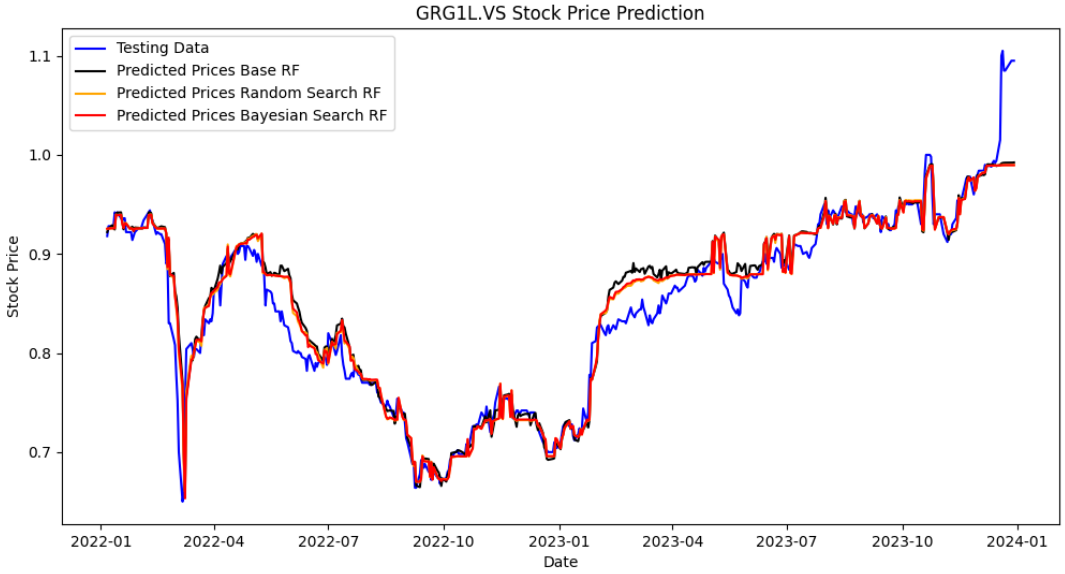

TSM1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
1.50
1.45
1.40
1.35
1.30
1.25
1.20
1.15
1.10
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01

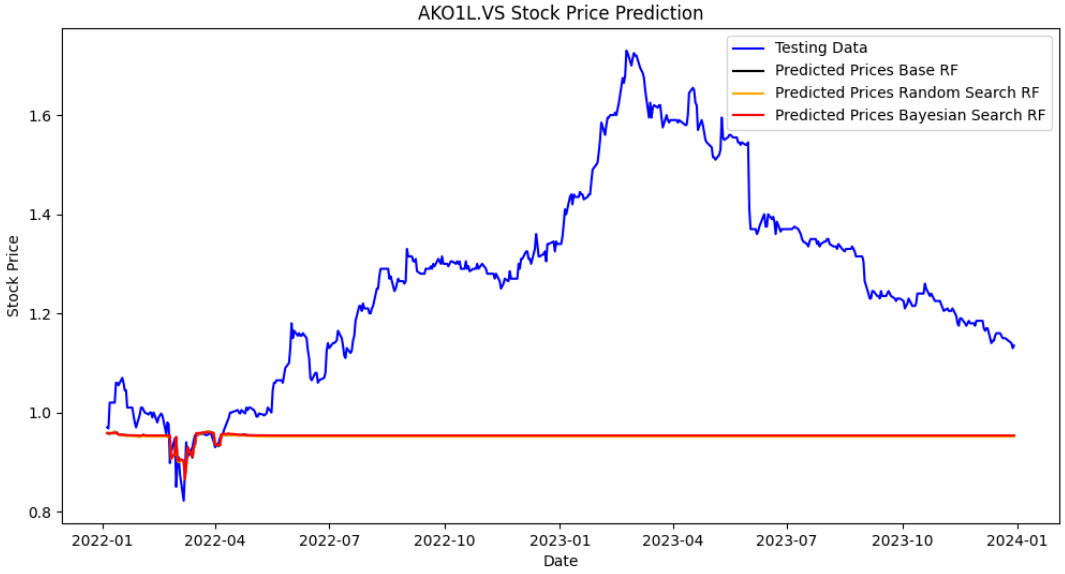

CPA1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
3.0
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
2023-10
2023-11
2023-12
2024-01

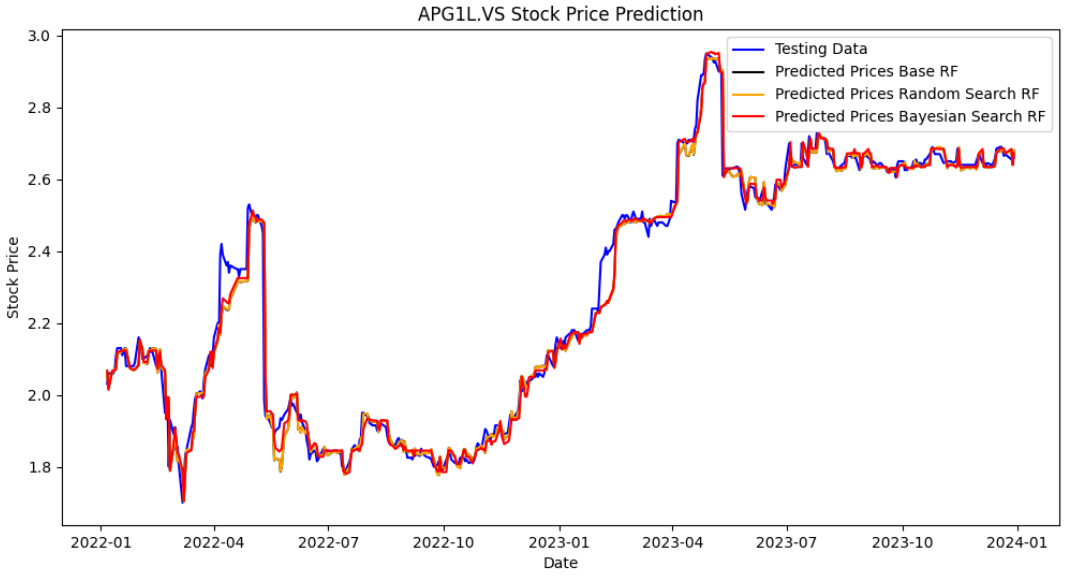

MRK1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
17
16
15
14
13
12
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

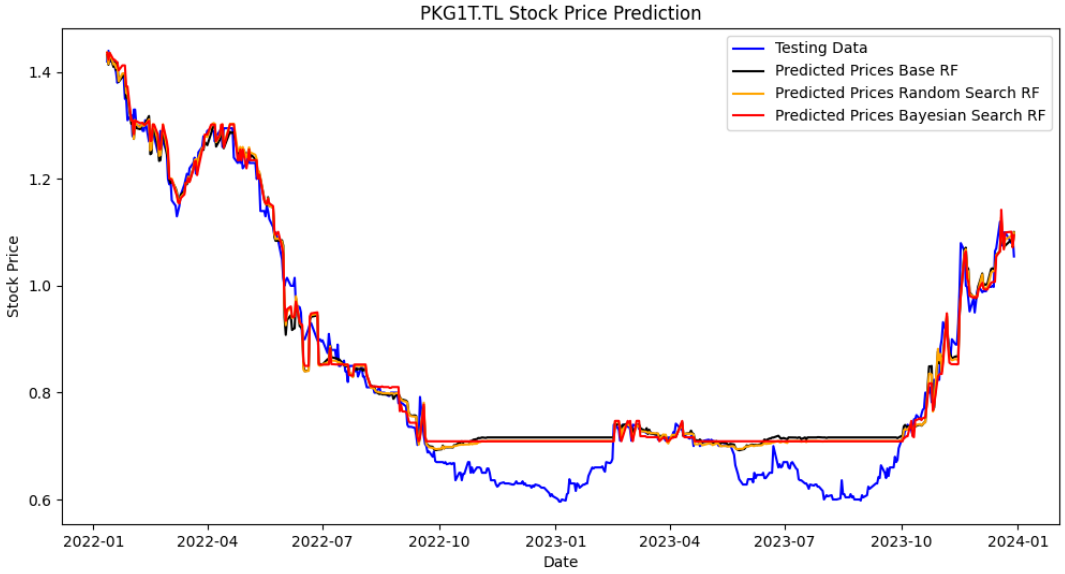

TVE1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
14.5
14.0
13.5
13.0
12.5
12.0
11.5
11.0
10.5
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

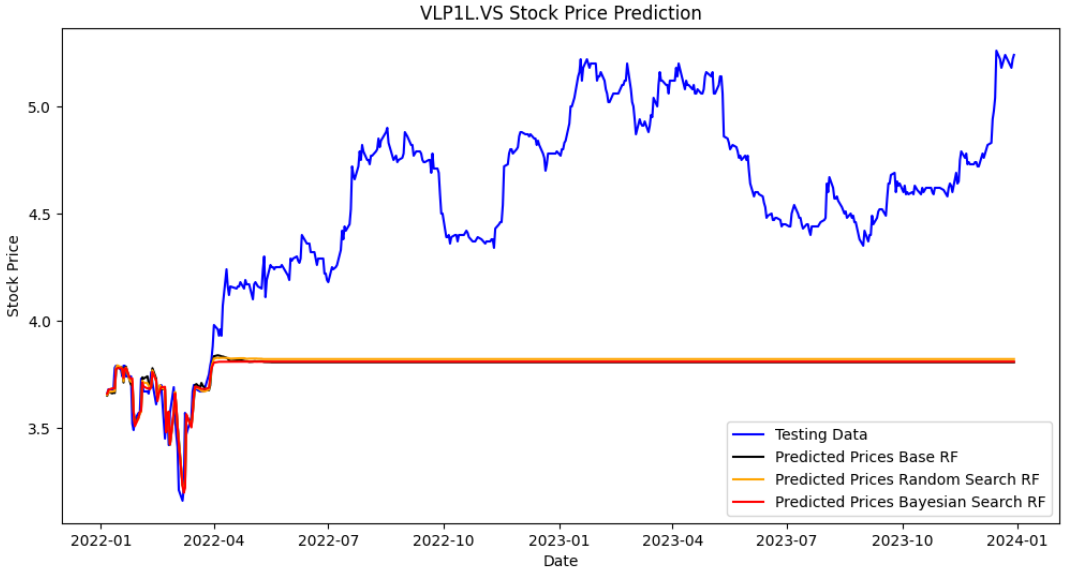

HAE1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
7.5
7.0
6.5
6.0
5.5
5.0
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

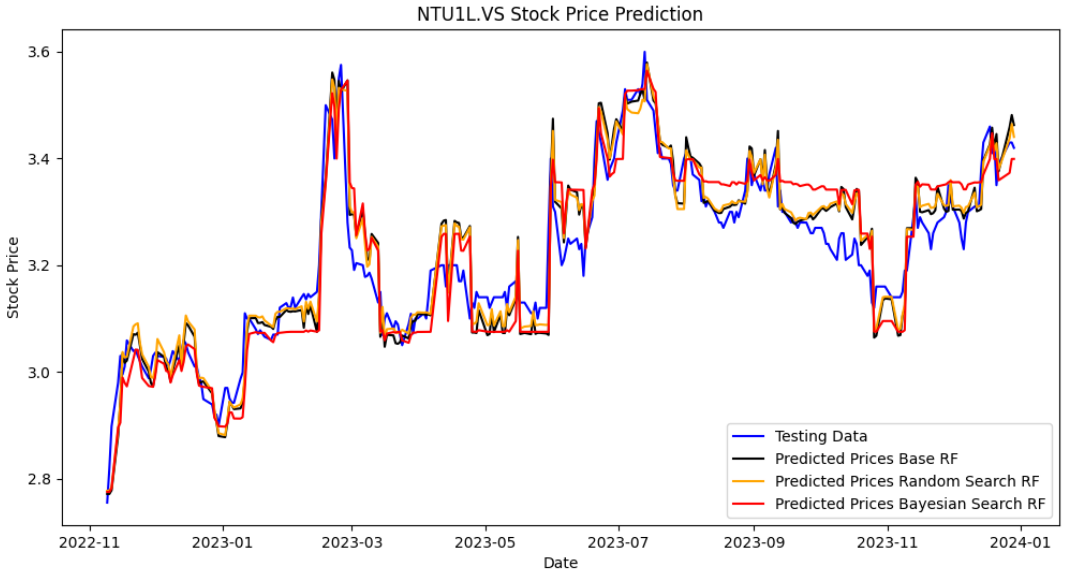

GRG1L.VS Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
1.1
1.0
0.9
0.8
0.7
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

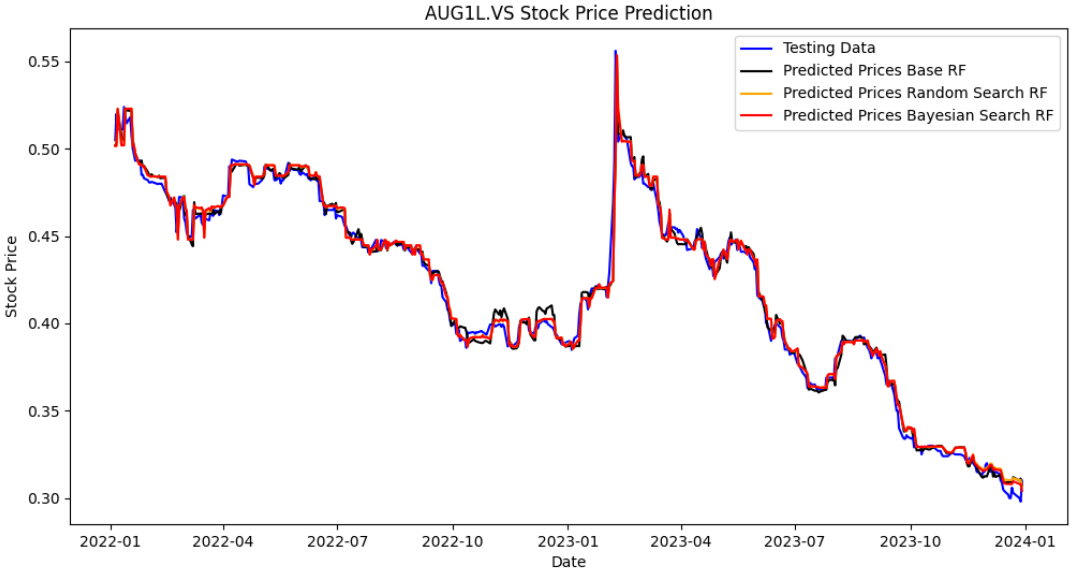

AKO1L.VS Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
1.6
1.4
1.2
1.0
0.8
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

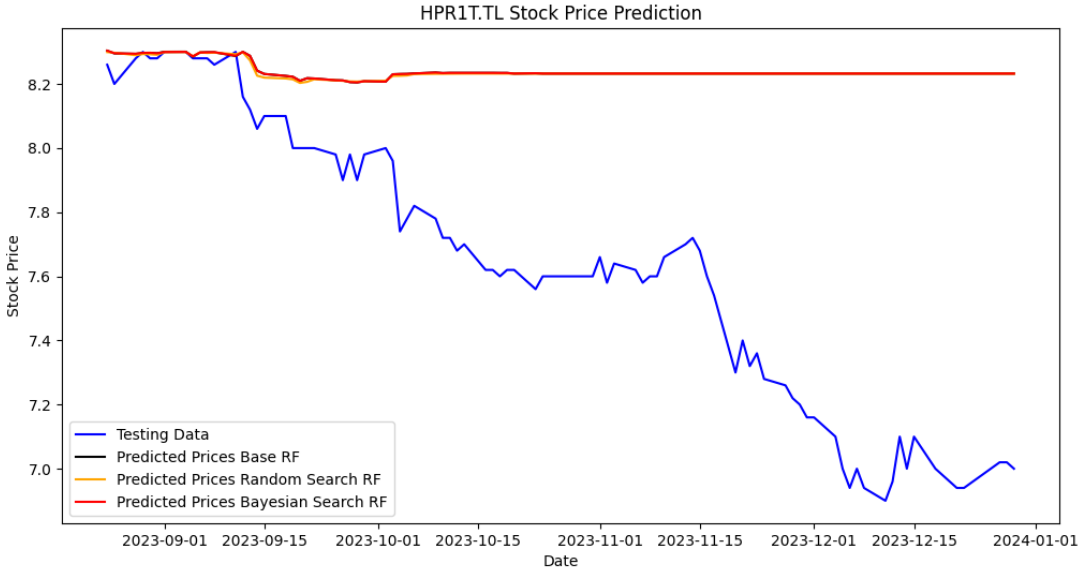

APG1L.VS Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
3.0
2.8
2.6
2.4
2.2
2.0
1.8
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01


PKG1T.TL Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
1.4
1.2
1.0
0.8
0.6
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

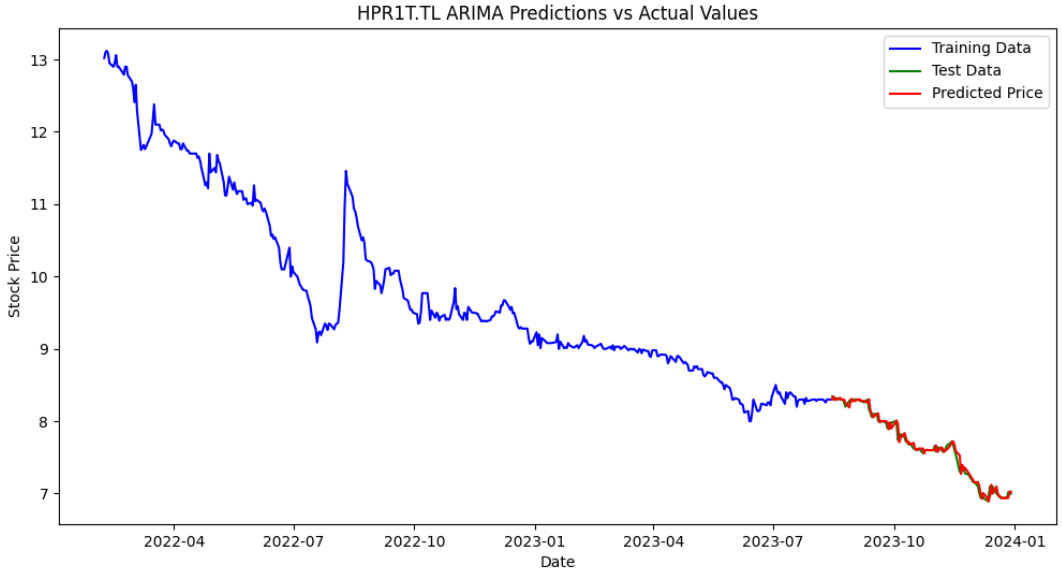

VLP1L.VS Stock Price Prediction
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF
Stock Price
Date
5.0
4.5
4.0
3.5
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

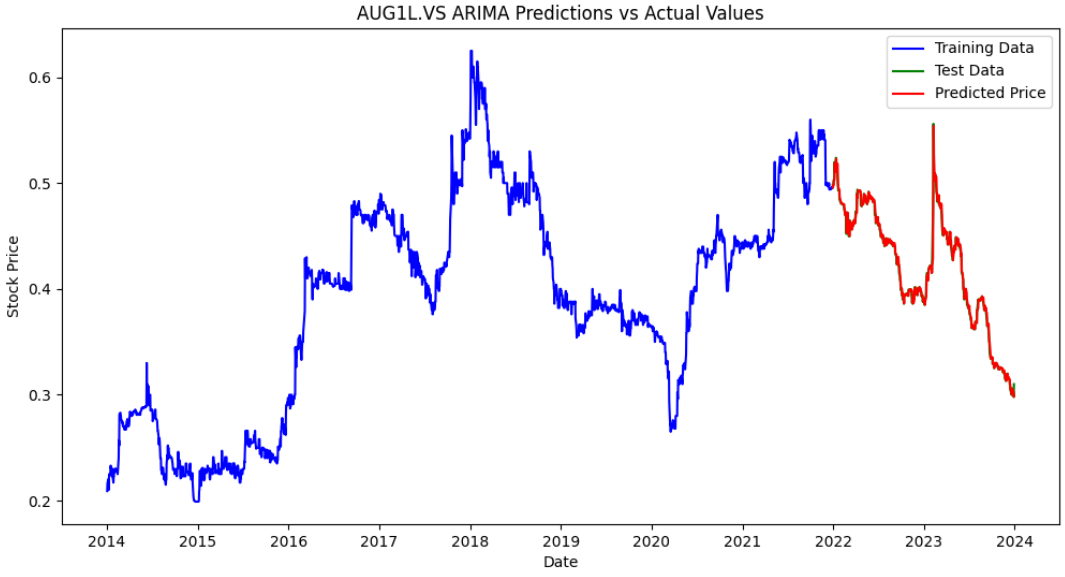

NTU1L.VS Stock Price Prediction
Stock Price
Date
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF

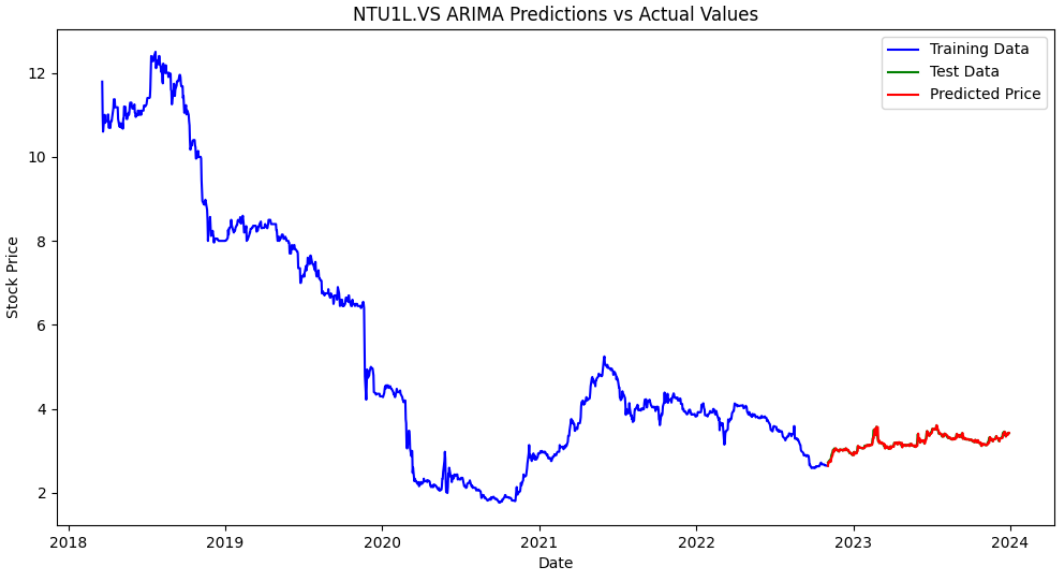

AUG1L.VS Stock Price Prediction
Stock Price
Date
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF

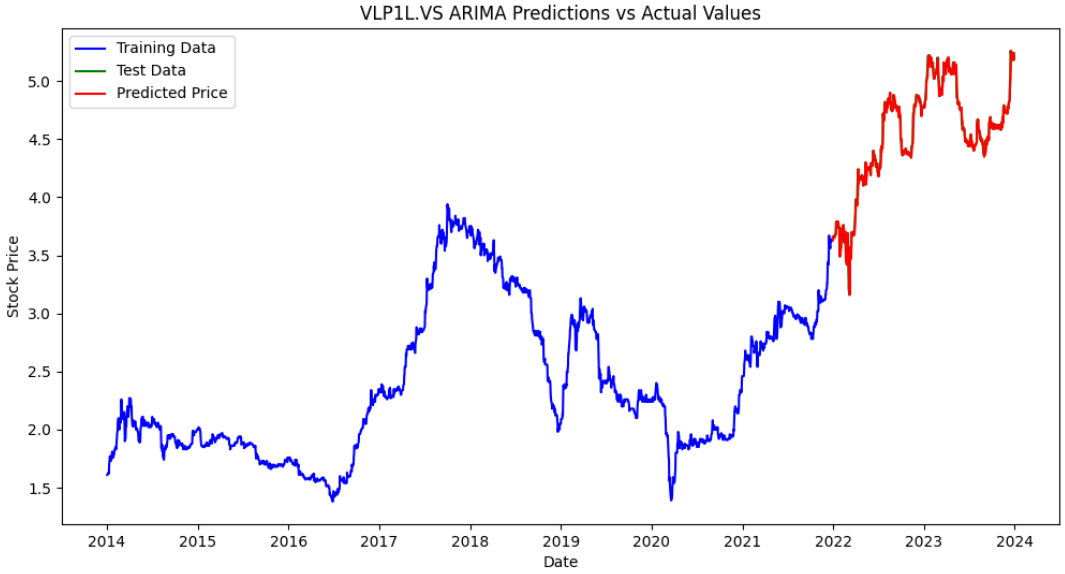

HPR1T.TL Stock Price Prediction
Stock Price
Date
Testing Data
Predicted Prices Base RF
Predicted Prices Random Search RF
Predicted Prices Bayesian Search RF

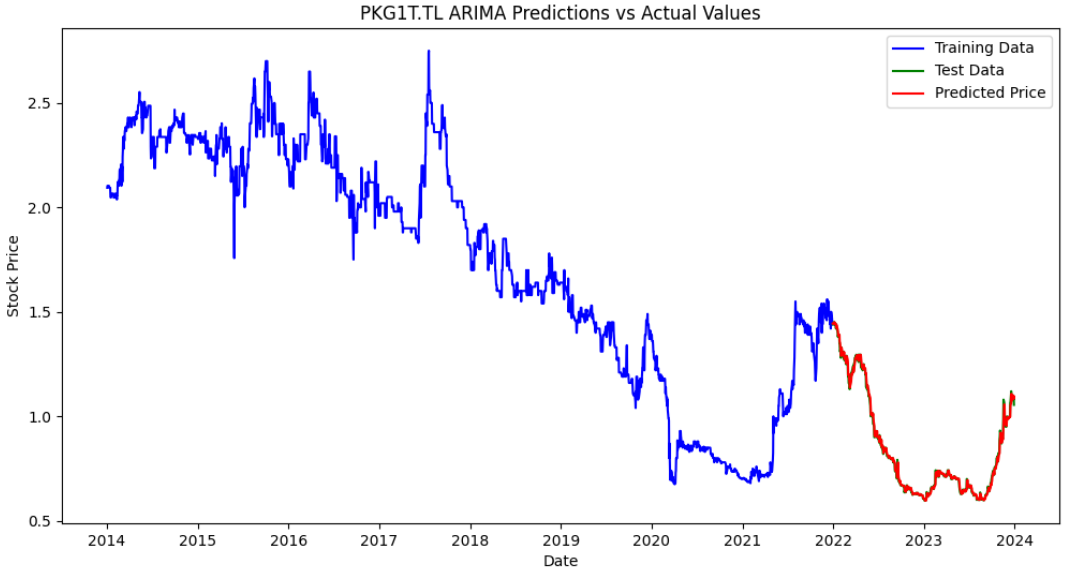
Appendix 4. Result Graphs of ARIMA Models.

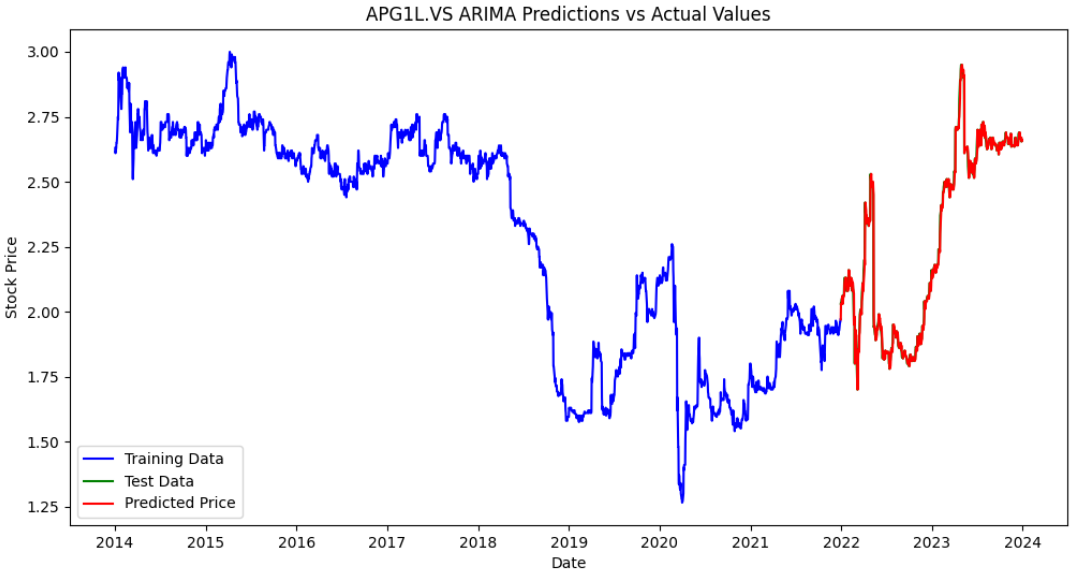


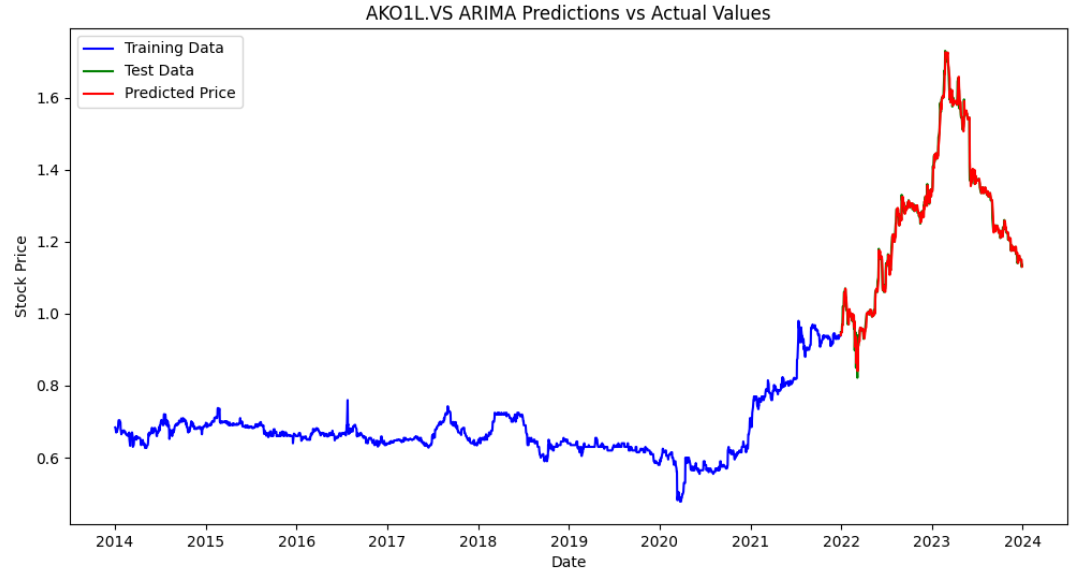


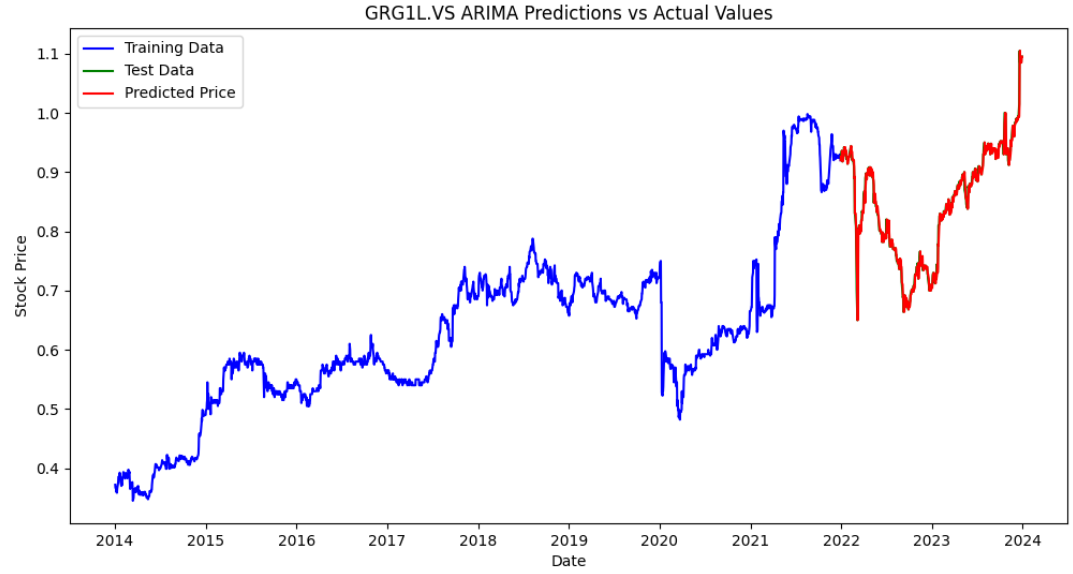

NTU1L.VS ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
12
10
8
6
4
2
2018
2019
2020
2021
2022
2023
2024

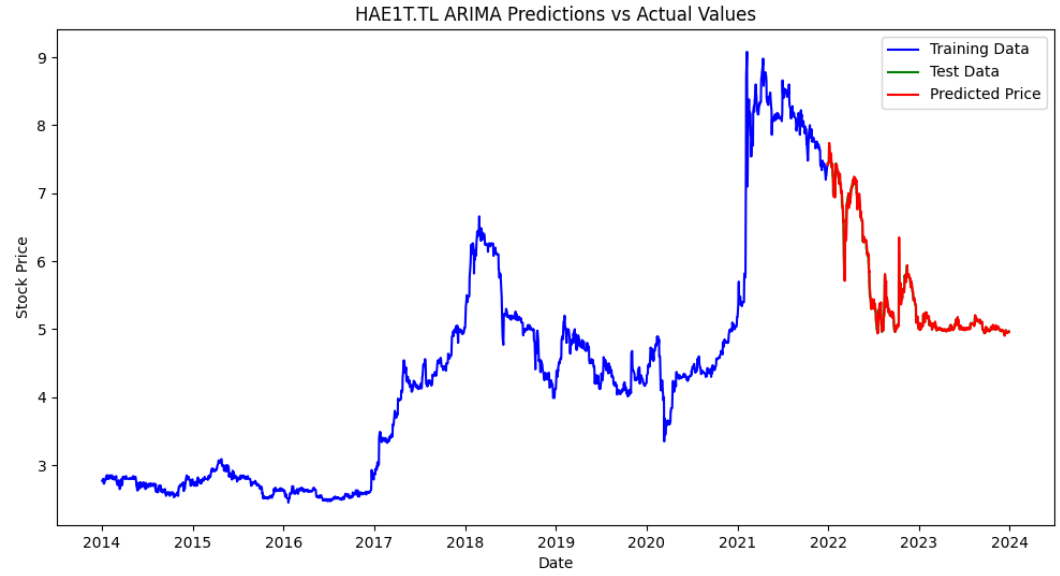

VLP1L.VS ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

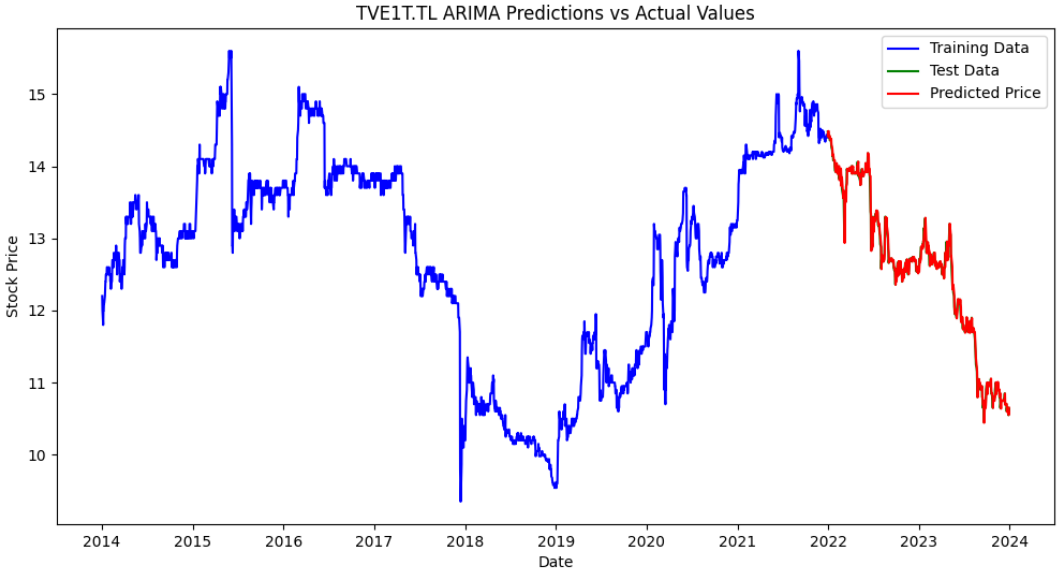

PKG1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
2.5
2.0
1.5
1.0
0.5
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

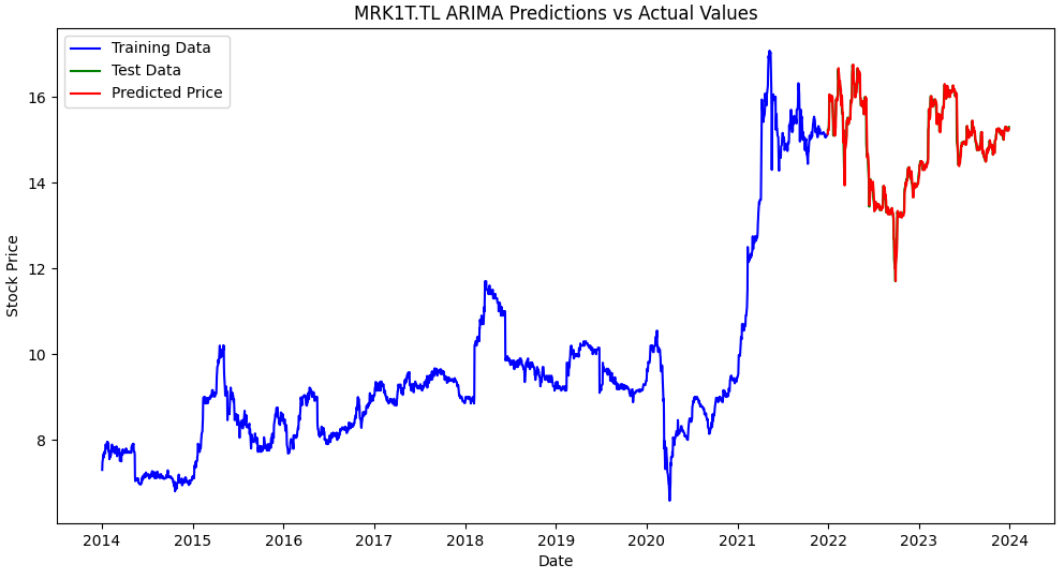

APG1L.VS ARIMA Predictions vs Actual Values
Stock Price
Date
Training Data
Test Data
Predicted Price
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

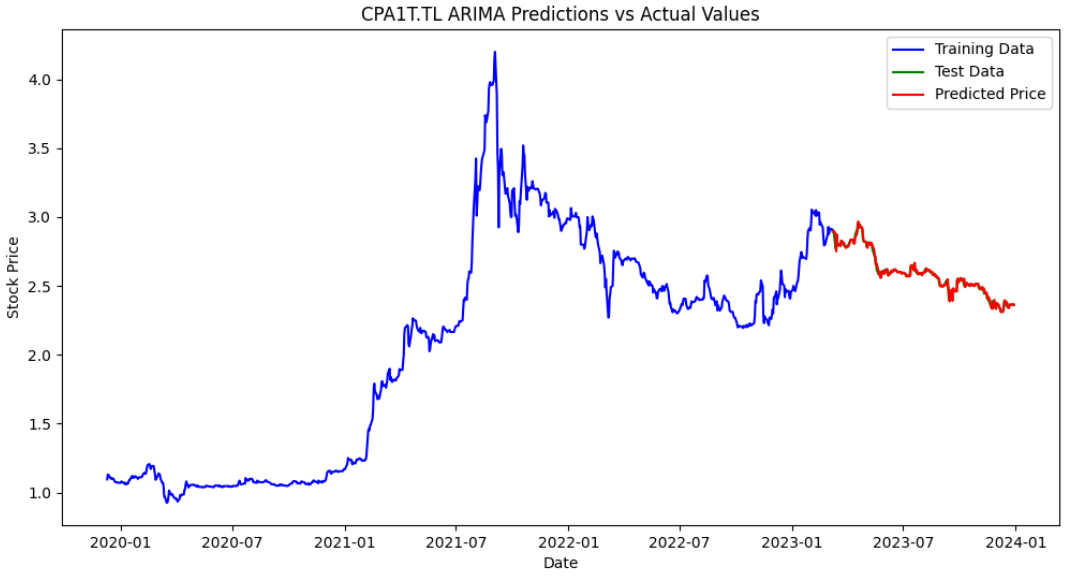

AKO1L.VS ARIMA Predictions vs Actual Values
Stock Price
Date
Training Data
Test Data
Predicted Price
1.6
1.4
1.2
1.0
0.8
0.6
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

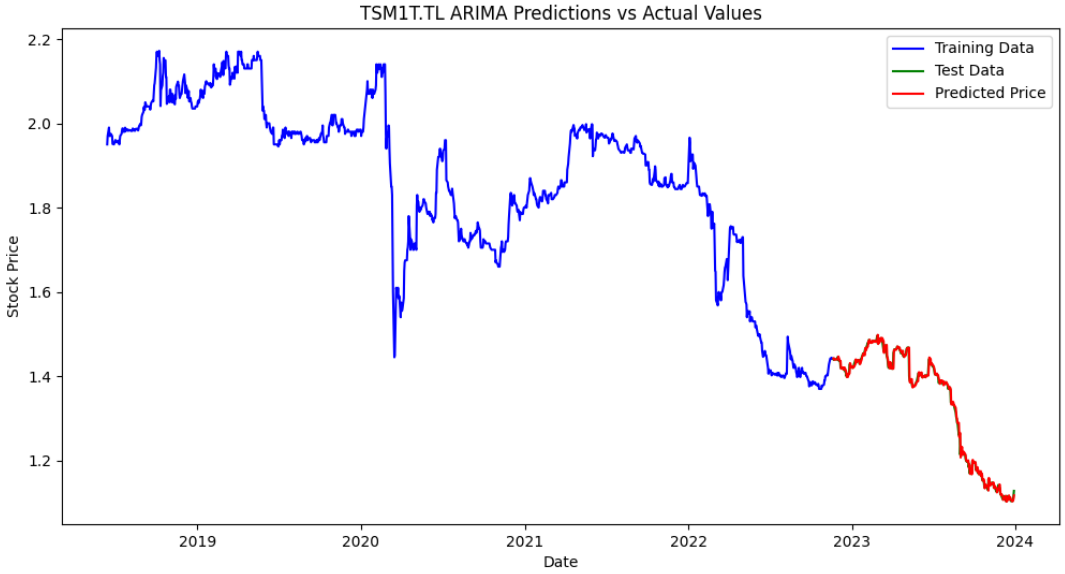

GRG1L.VS ARIMA Predictions vs Actual Values
Stock Price
Date
Training Data
Test Data
Predicted Price
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

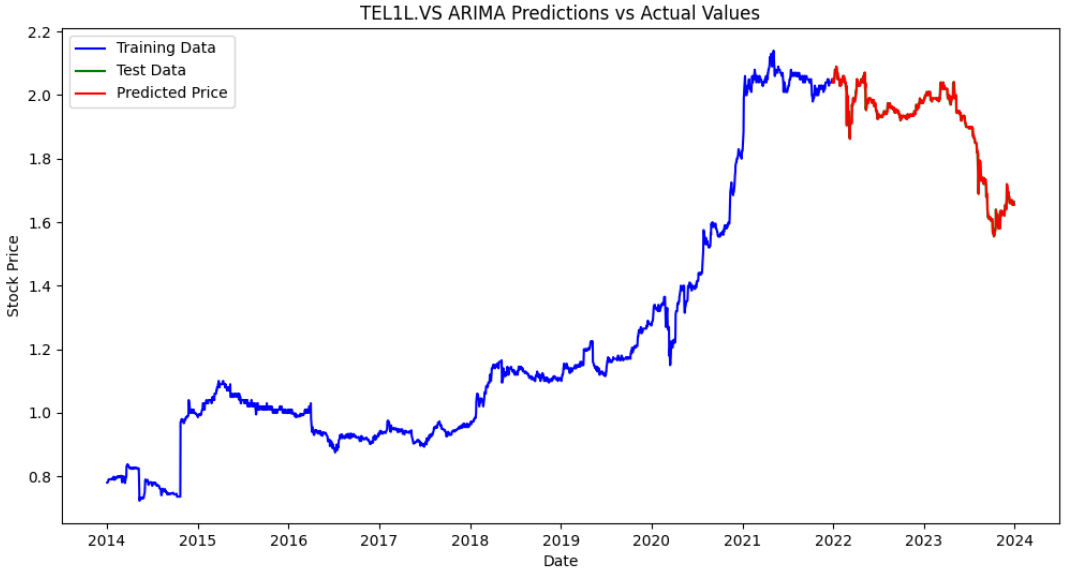

HAE1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

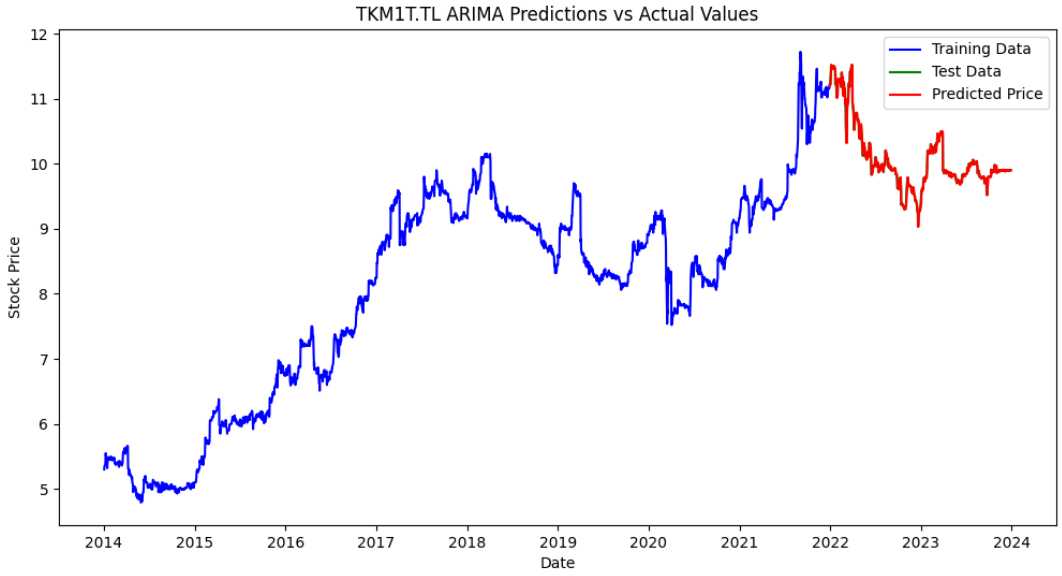

TVE1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

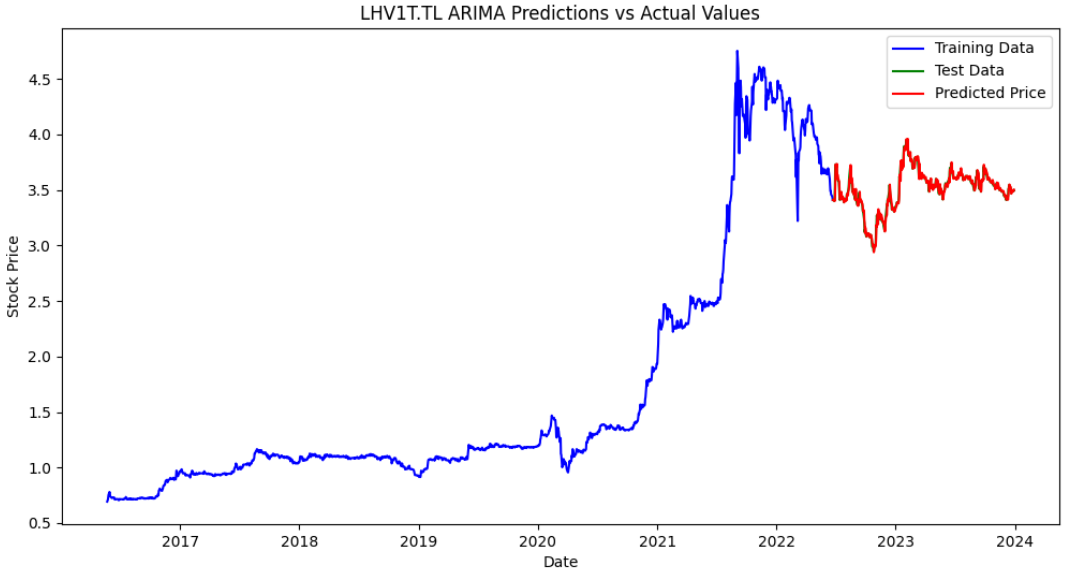

MRK1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

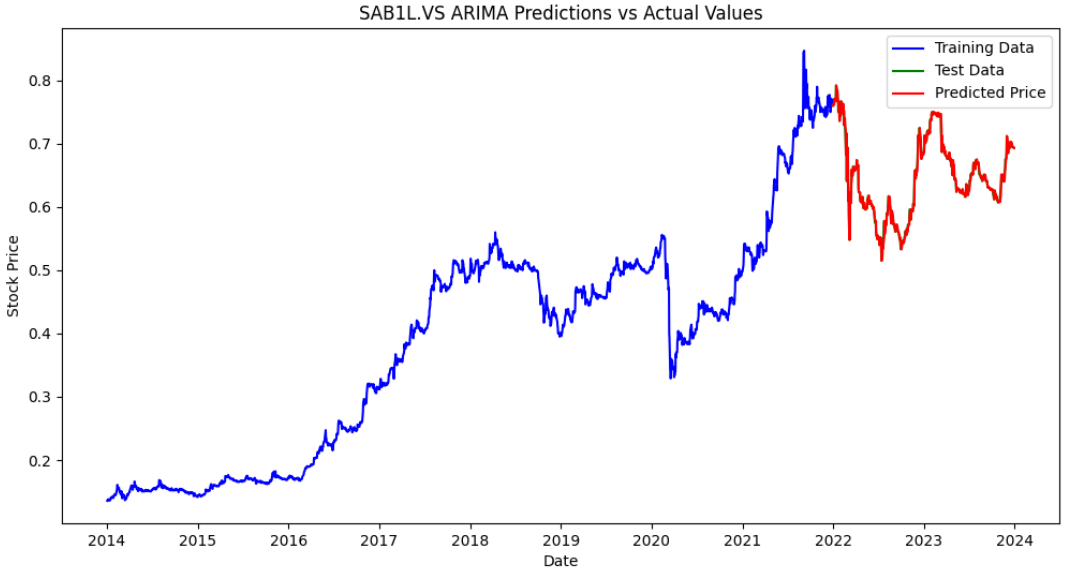

CPA1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
4.0
3.5
3.0
2.5
2.0
1.5
1.0
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01

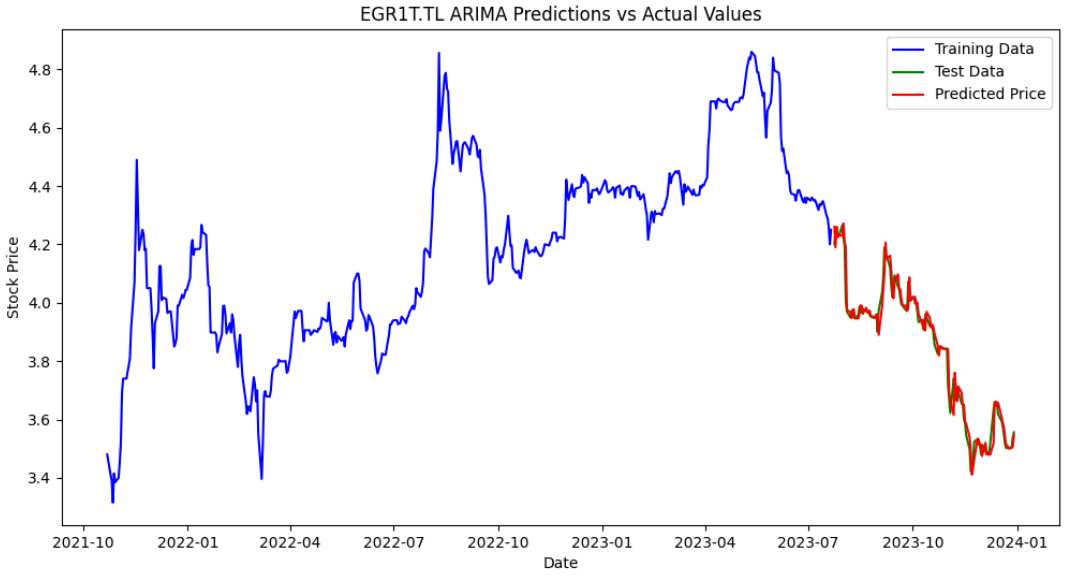

TSM1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
2.2
2.0
1.8
1.6
1.4
1.2
2019
2020
2021
2022
2023
2024

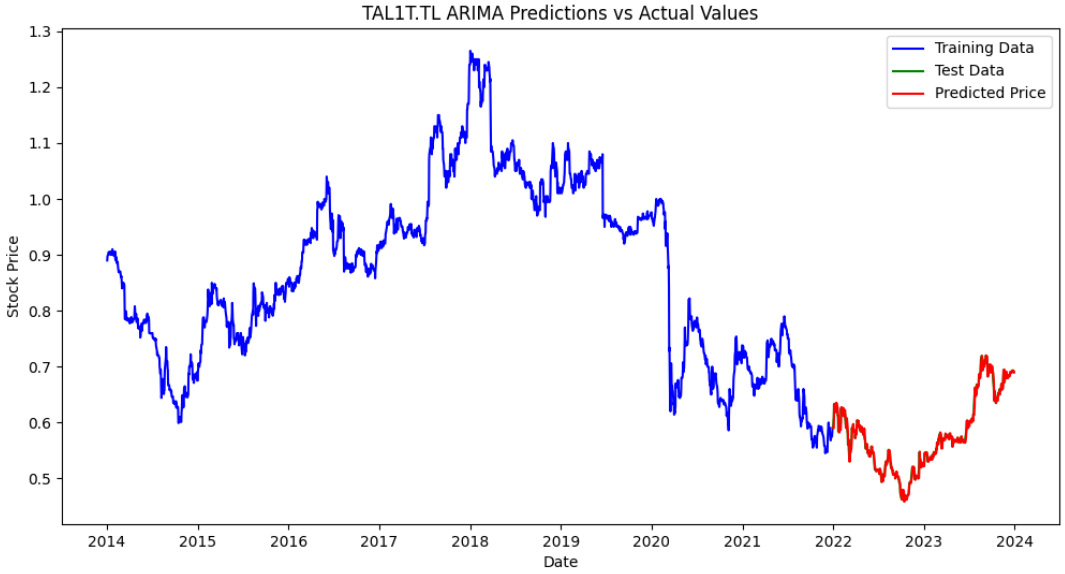

TEL1L.VS ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

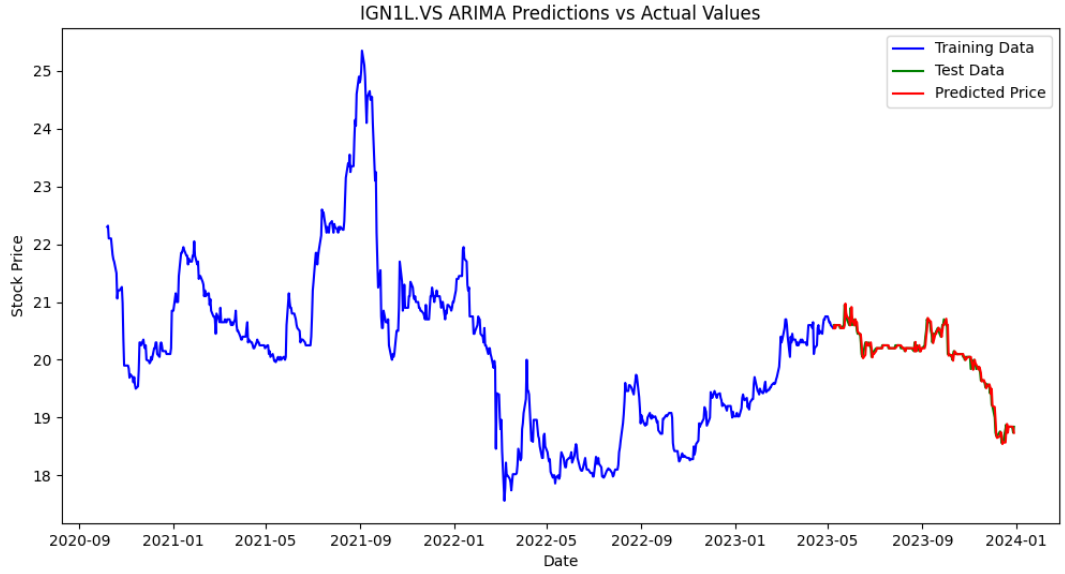

TKM1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
12
11
10
9
8
7
6
5
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024


LHV1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
2017
2018
2019
2020
2021
2022
2023
2024

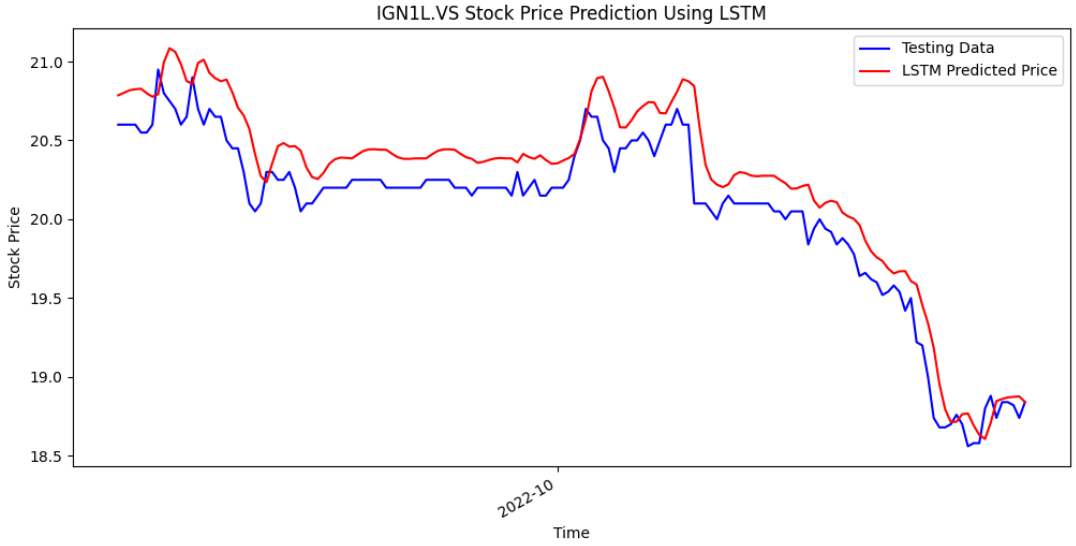

SAB1L.VS ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
0.8
0.7
0.6
0.5
0.4
0.3
0.2
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

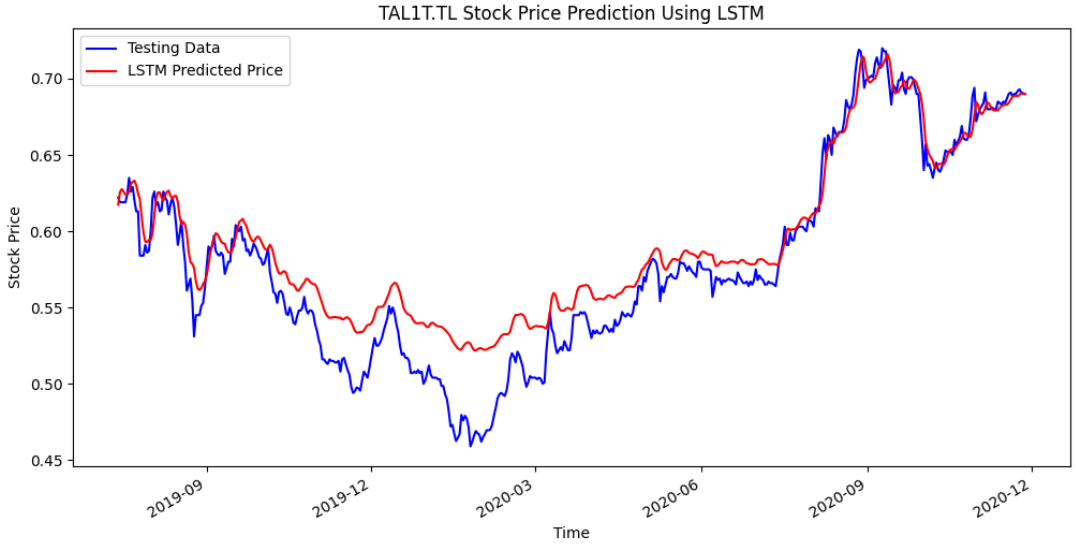

EGR1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
4.8
4.6
4.4
4.2
4.0
3.8
3.6
3.4
2021-10
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

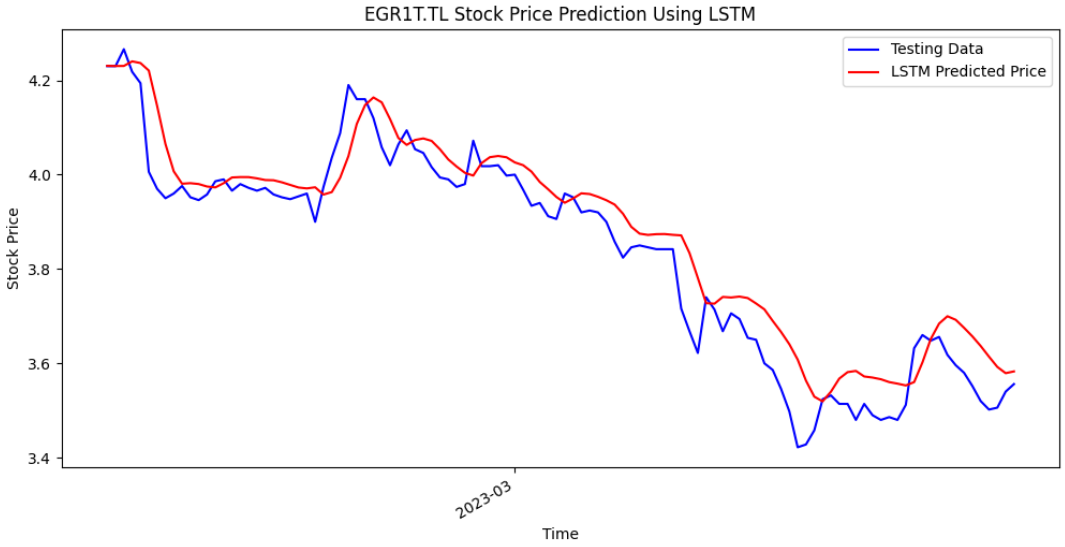

TAL1T.TL ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

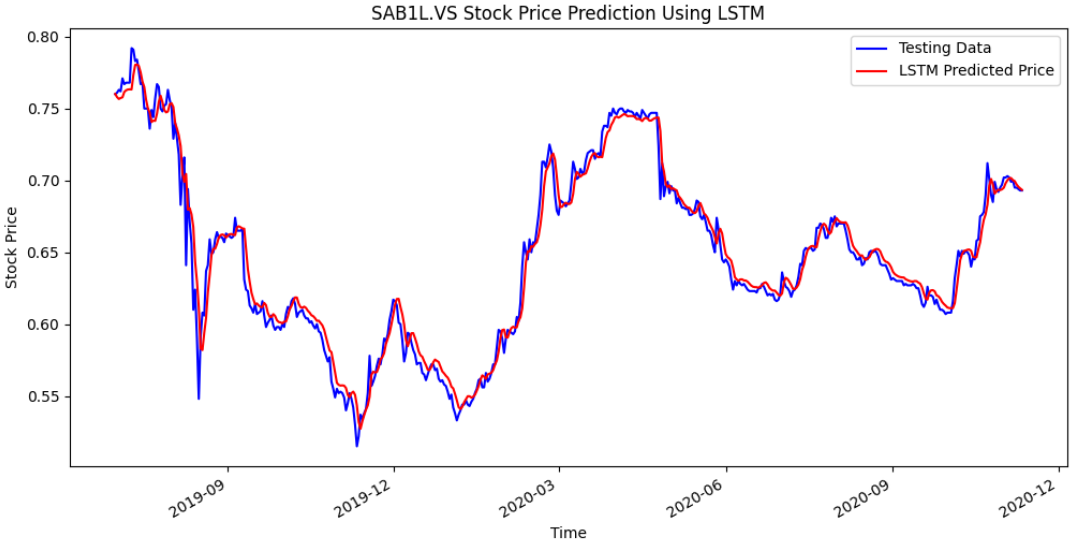

IGN1L.VS ARIMA Predictions vs Actual Values
Training Data
Test Data
Predicted Price
Stock Price
Date
25
24
23
22
21
20
19
18
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01

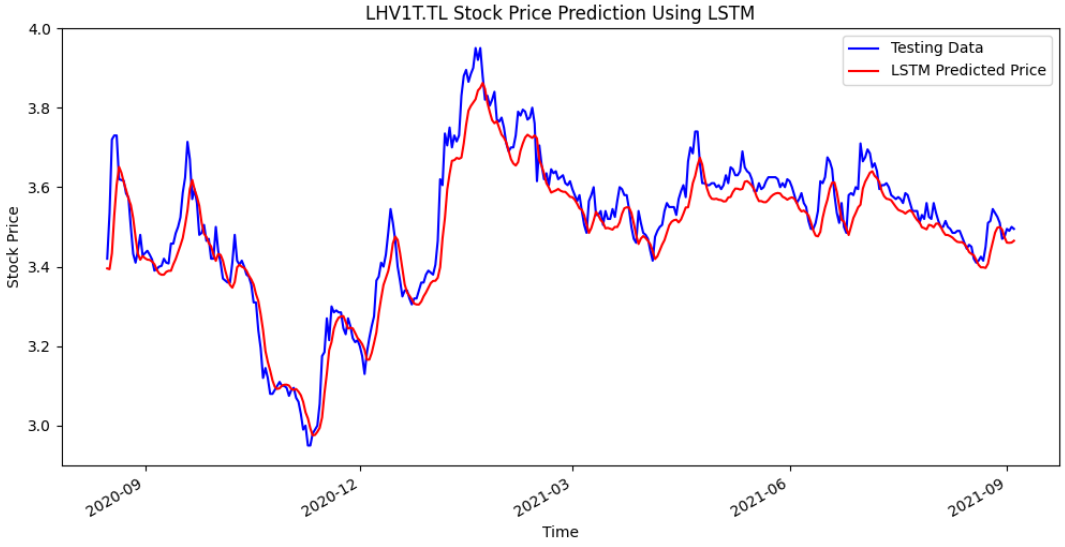
Appendix 5. Result Graphs of LSTM Models.

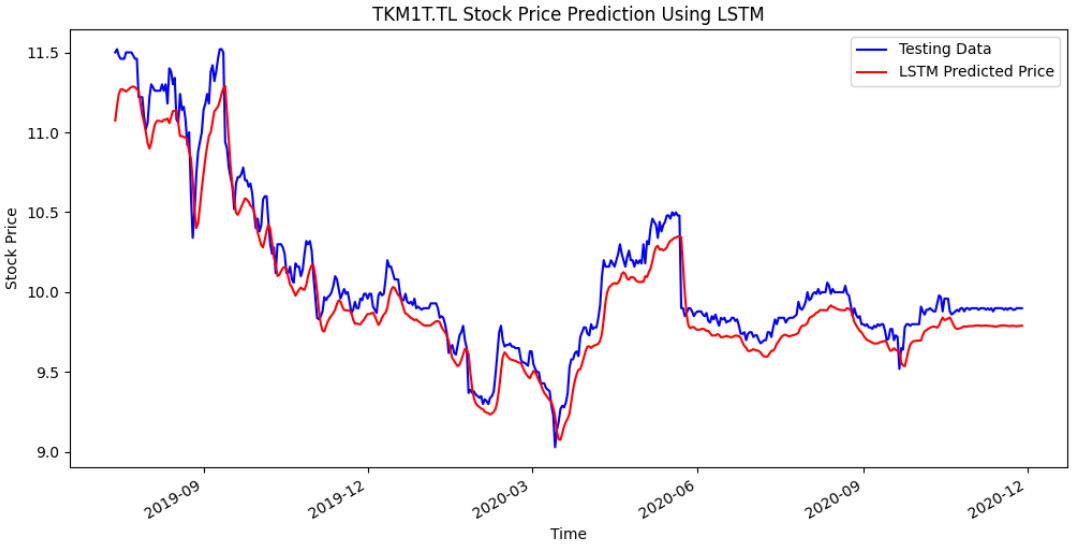


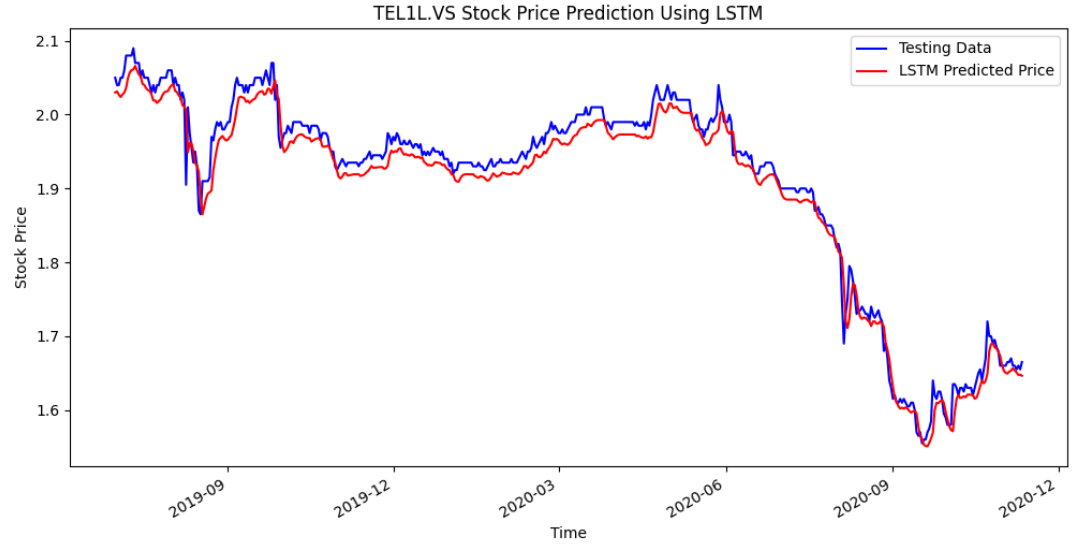


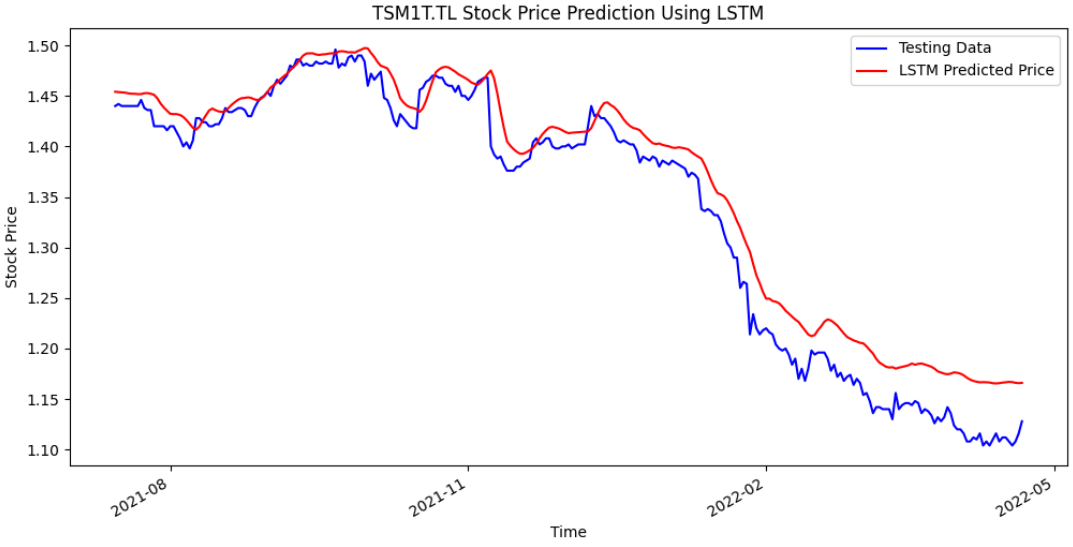

EGR1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
4.2
4.0
3.8
3.6
3.4
2023-03
Time

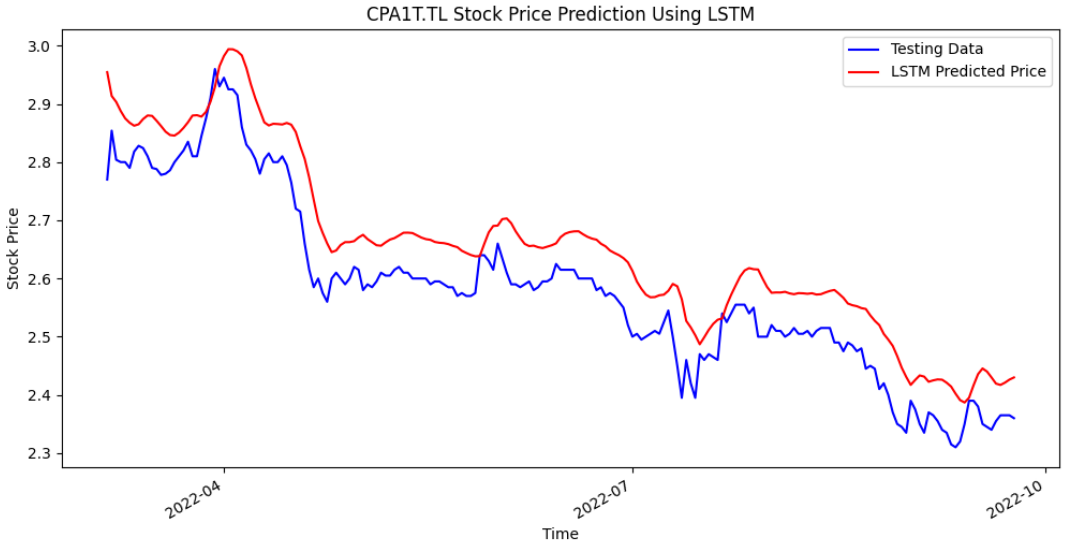

SAB1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
0.80
0.75
0.70
0.65
0.60
0.55
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12
Time

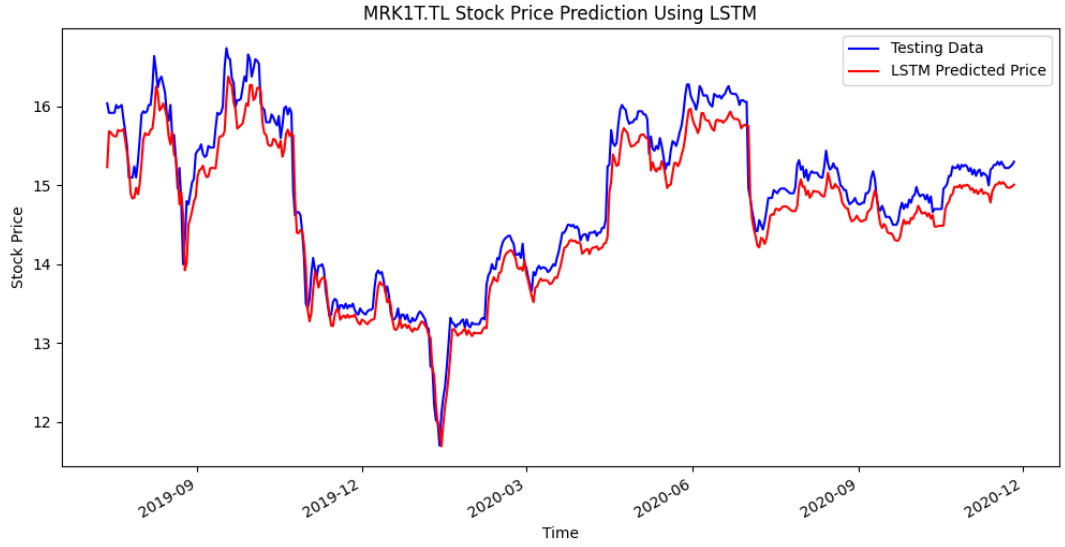

LHV1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
4.0
3.8
3.6
3.4
3.2
3.0
2020-09
2020-12
2021-03
2021-06
2021-09
Time

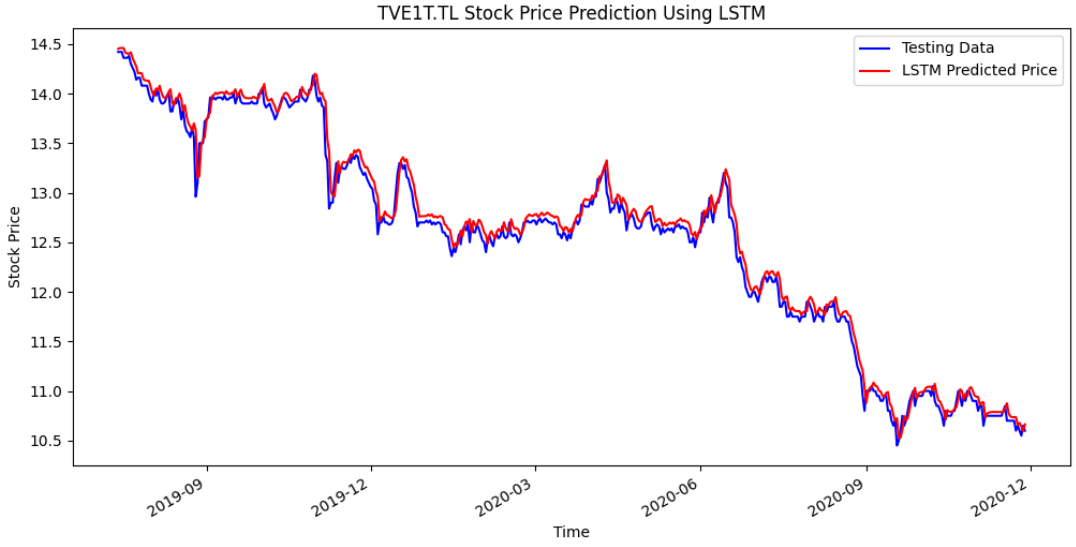

TKM1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
11.5
11.0
10.5
10.0
9.5
9.0
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12

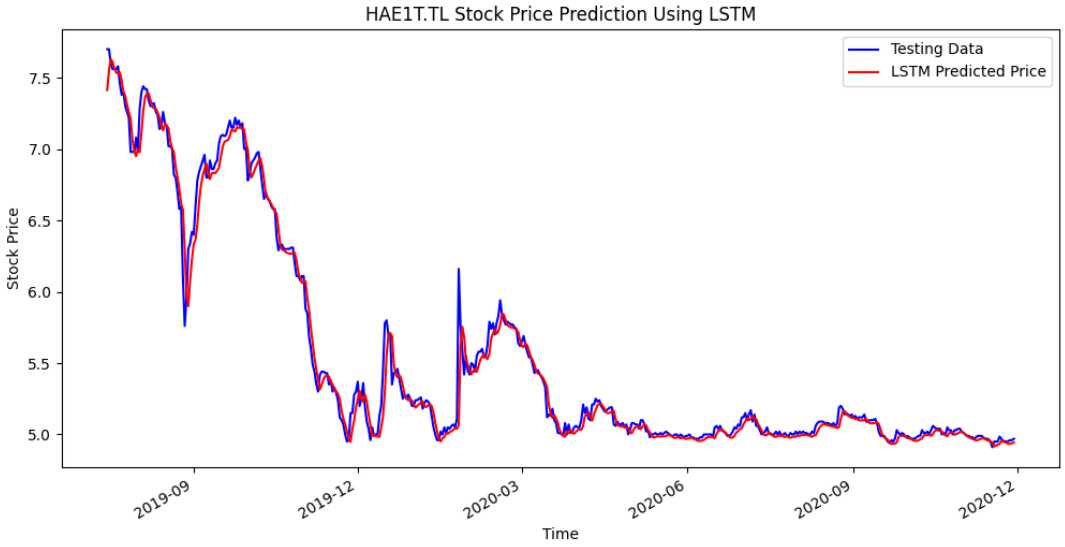

TEL1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
2.1
2.0
1.9
1.8
1.7
1.6
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12

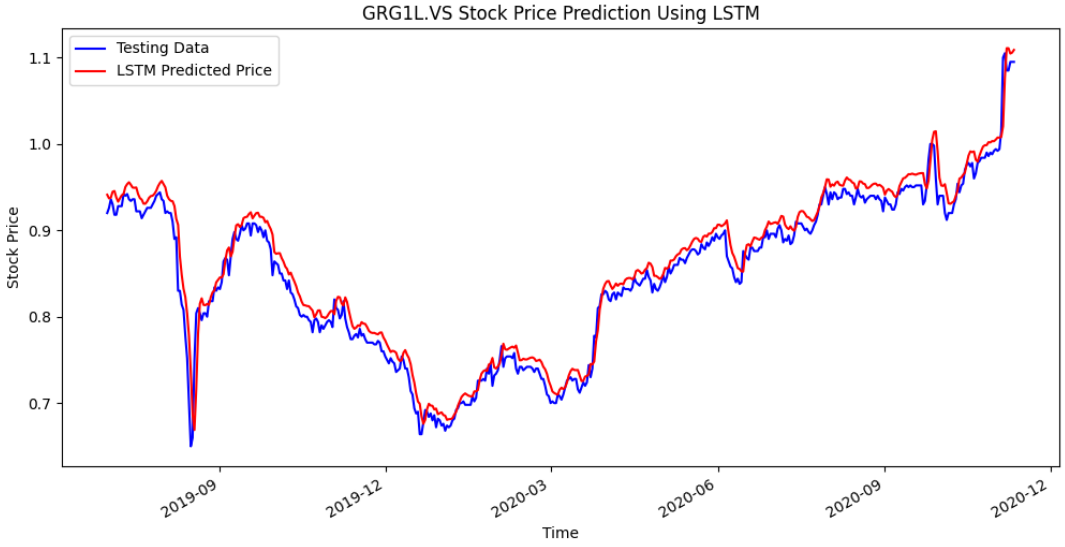

TSM1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
1.50
1.45
1.40
1.35
1.30
1.25
1.20
1.15
1.10
2021-08
2021-11
2022-02
2022-05

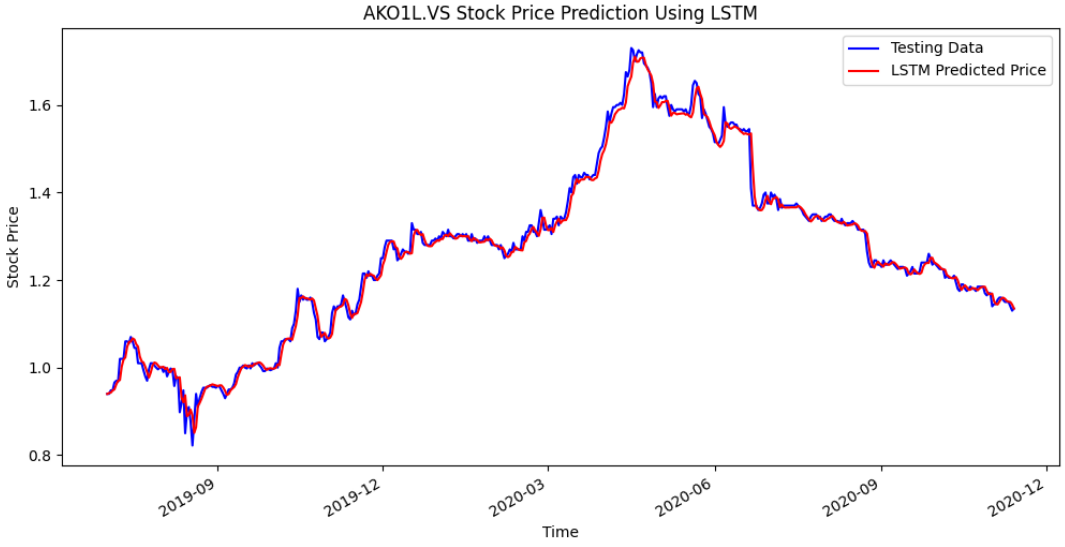

CPA1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
3.0
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2022-04
2022-07
2022-10

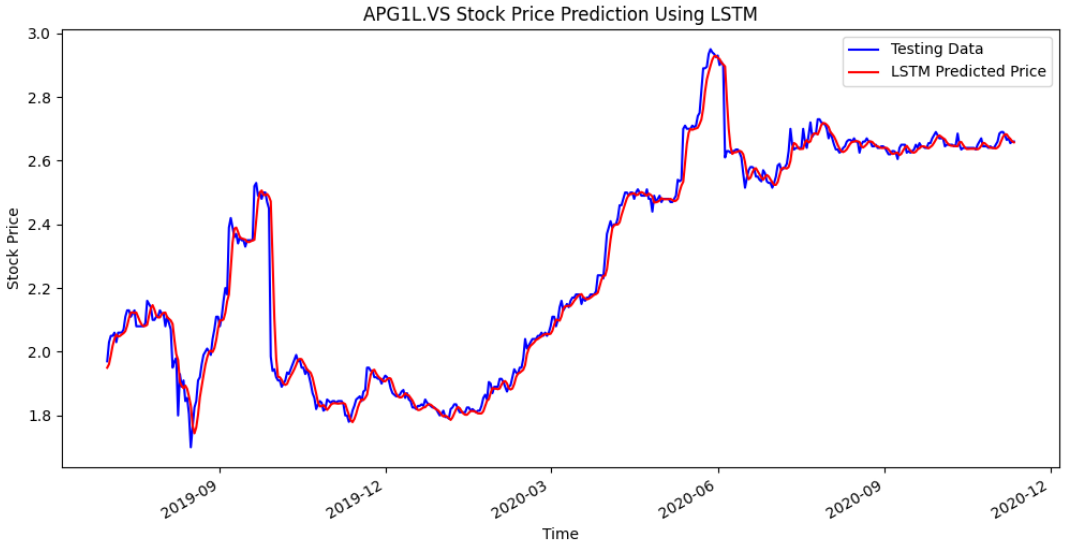

MRK1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
16
15
14
13
12
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12

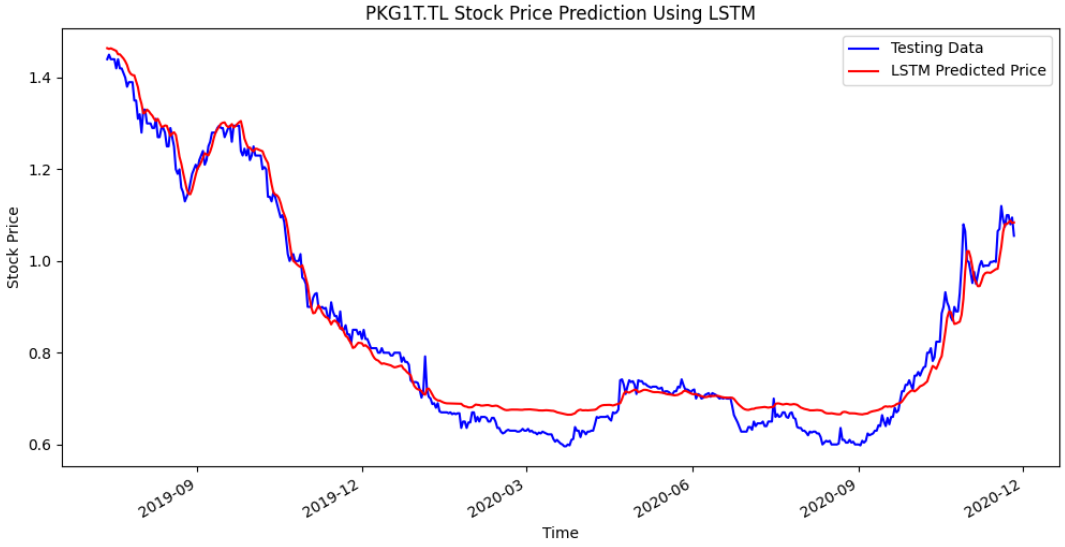

TVE1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
14.5
14.0
13.5
13.0
12.5
12.0
11.5
11.0
10.5
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12

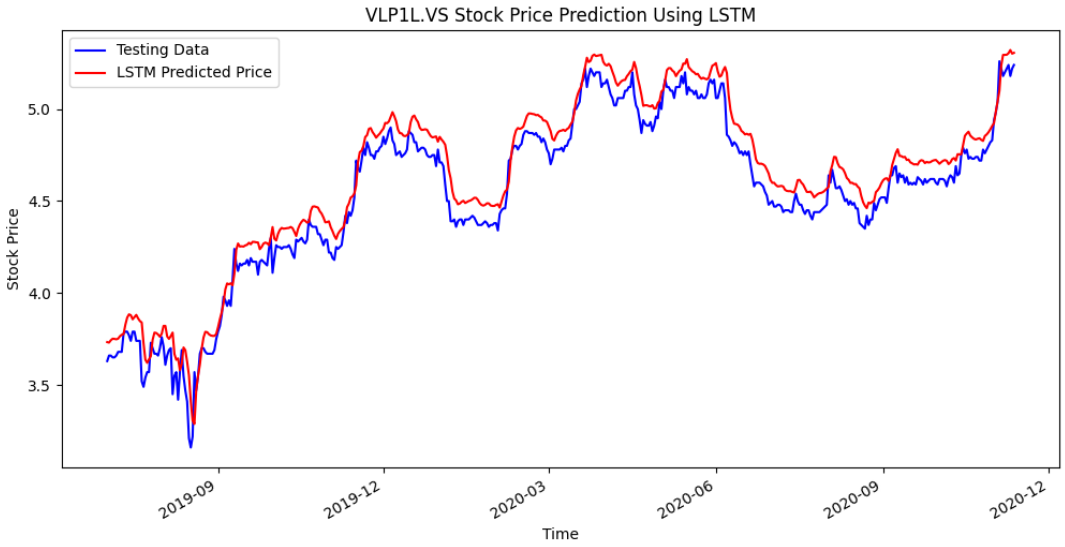

HAE1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
7.5
7.0
6.5
6.0
5.5
5.0
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12
Time

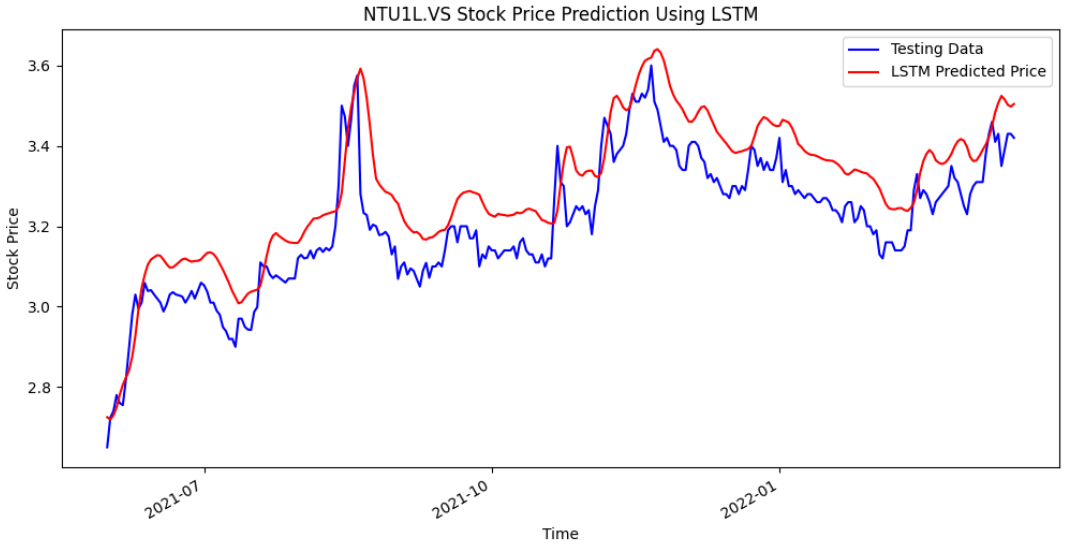

GRG1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
1.1
1.0
0.9
0.8
0.7
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12
Time

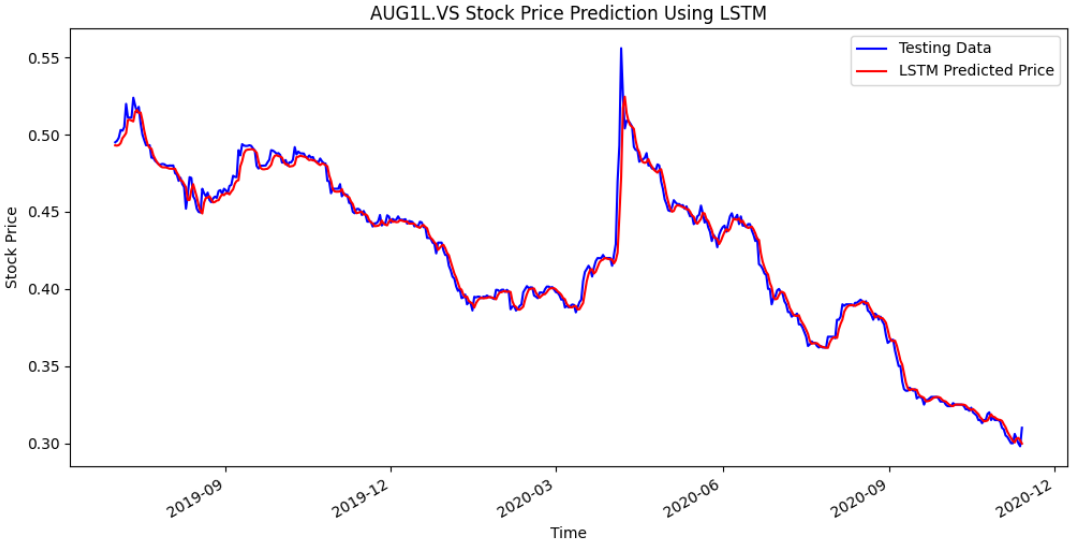

AKO1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
1.6
1.4
1.2
1.0
0.8
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12
Time

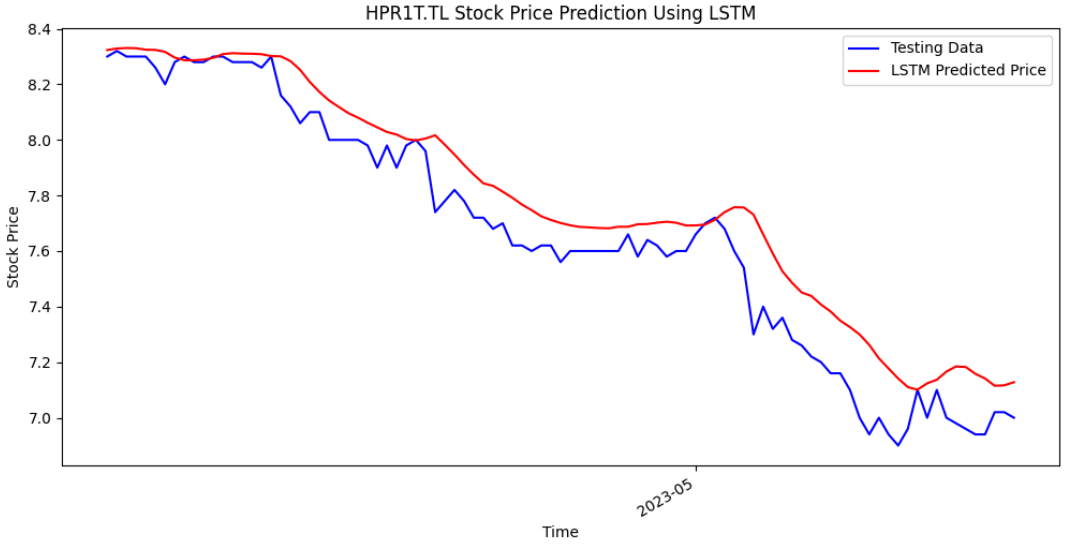

APG1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
3.0
2.8
2.6
2.4
2.2
2.0
1.8
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12


PKG1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
1.4
1.2
1.0
0.8
0.6
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12


VLP1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
Time
5.0
4.5
4.0
3.5
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12


NTU1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
3.6
3.4
3.2
3.0
2.8
2021-07
2021-10
2022-01
Time


AUG1L.VS Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
0.55
0.50
0.45
0.40
0.35
0.30
2019-09
2019-12
2020-03
2020-06
2020-09
2020-12
Time


HPR1T.TL Stock Price Prediction Using LSTM
Testing Data
LSTM Predicted Price
Stock Price
8.4
8.2
8.0
7.8
7.6
7.4
7.2
7.0
2023-05
Time