University of Twente

Faculty of Behavioural, management and Social Sciences

Master Thesis Business Administration: Finance

Trend Price Prediction of the S&P500 Using the VIX index, Moving Average and Trading Volume in a Combined Random Forest and LSTM Model

Abstract: Accurately predicting stock price movements is a critical objective for financial analysts to optimize portfolio performance and manage risk. This study investigates the effectiveness of machine learning models to forecast price trends in the S&P500. Herein, a Random Forest (RF) and Long Short-Term Memory (LSTM) combined model with technical indicators such as the VIX index, moving averages, and trading volume, was investigated. A series of experiments were conducted to evaluate whether these models could outperform traditional investment strategies such as buy-and-hold or random buy heuristics. The results show that the LSTM model, without incorporating the VIX index as a feature, achieved the highest F1-score of 68%. Furthermore, it performed better than a random buy strategy. However, both the LSTM and the Random Forest models were unable to outperform the buy-and-hold strategy. Furthermore, integrating LSTM predictions as a feature into the Random Forest model did not enhance predictive accuracy. A notable finding is that shifting the testing periods negatively affected the models' predictability indicating that the parameters of the model constantly need to be updated to remain accurate. These insights highlight the limitations of current approaches, pointing to areas for improvement such as integrating the VIX index into LSTM models to enhance prediction accuracy.

Keywords: LSTM Model, Moving Average, Price trend prediction, Random Forest, S&P500, Trading Volume, VIX index

Student
Ir. T.A. Dissevelt (S2180200)

Supervisor
Dr.Ir. W.J.A. van Heeswijk
Dr. M.R. Machado

October 17, 2024

Contents

1 Introduction

The ability to predict stock price movement is a persistent desire in financial investment strategy research. Stock traders and investors are naturally seeking to gain (more) insights into the direction of the future price of a stock or index. The ability to accurately predict the directions could significantly impact the trader' risk management practices, investments strategies and their portfolio performance. However, the challenge of predicting the future trend of the market is found in the dynamic and complex nature of the financial markets, in which market movement depend on numerous factors such as the psychology of traders, politics, firm performance, economical performance and exchange rates (Jiao & Jakubowicz, 2017).
Despite the challenges, several studies emphasize the importance of being able to accurately predict stock price in financial decision-making. The understanding of the stock market, in forecasting stock returns, are important for asset pricing models and investment strategies. Investors and traders base their strategies on price movements to maximize returns and minimize risk and thus require reliable predictive tools (Fama & French, 1992). However, many studies suggest that the market is driven under the efficient market hypothesis, which assumes that future stock price movements follow a random walk pattern, meaning that making predictions about stock price movements practically impossible (Fama & French, 1992). The efficient market hypothesis can be classified in three forms: the weak form, the semi-strong form and the strong form. The weak form is known as the circumstance in which the current price reflects all past prices of the financial asset. Herein, other financial information is not included. Therefore, some traders might profit using fundamental information to predict prices trends (Ţiţan, 2015). The semi-strong form assumes that the price of financial assets at any moment reflect all the publicly available information present on the market, thus also including historical prices and other publicly available historical information. Moreover, it is assumed that the prices of the assets change rapidly and are unbiased towards the incorporation of any other new public information released on the market. This theory therefore suggests that neither technical nor fundamental analysis can determine how an investor should divide his funds in order to obtain a higher profitability than that achieved in case of investment in a random portfolio of financial assets. Lastly, the strong form of the efficient market hypothesis assumes that prices include all the available information on a market. So in addition to the semi-strong assumption this assumption also assumes that private information regarding a financial asset is included in the price. Since all information is incorporated, it is impossible to predict price movements, and therefore the price movements are considered to be follow a random walk(Ţiţan, 2015).
Other studies, for example the ones conducted by Gallagher and Taylor (2002) and Fama (1991) do not support the random walk and efficient market hypothesis by suggesting that the stock price movement could be predicted using fundamental analysis or a technical analysis. Fundamental analysis examines the underlying value (the true worth of the stock based on its underlying fundamentals) of the stock by evaluating the financial condition of the company by for example by analysing the earnings, management quality and industry trends to determine whether a stock is priced attractively relative to its intrinsic value. Technical analysis is a evalutation method for future price movement (development of an asset's price over time in terms of up or down trend) that is based on historical price and volume data. Herein, it relies mainly on the stock's price chart and statistical indicators to identify trends in the movement of the price. On the price chart the trader tries to find trend lines to identify the future direction of the trend. Herein, they use technical indicators analyze the price data and to create their future price/trend prediction. Thus, by analyzing charts, patterns, indicators, and using other technical tools, traders attempt to forecast market trends and make trading decisions accordingly. However, it must be noted that technical analysis is subjective and relies on interpretation, so different analysts may reach different conclusions when analyzing the same data (Thompson, 2023). The study of Gallagher and Taylor (2002), in which a technical analysis was conducted, found that by using the inter-day returns, the returns of the intra-day could more effectively be predicted since during the inter-day the news stream was eliminated. Therefore, the market was less exposed to price manipulation due to news streams during this period. The result of the majority of tests showed that the inter-day could help intra-day traders to generate more precise conclusions and therefore they concluded that the market is not fully efficient. Subsequently, the study conducted by Fama (1991) concluded that the stock returns for both short and long hori-

zons are predictable by observing dividend yields, E/P ratios and default spread of low- over high grade bond yields (fundamental analysis), which also suggests that the random-walk hypothesis is not supported. In a technical analysis the trader assumes that the market is able to provide the necessary information (such as price and trading volume time series, trends, chart patterns and the use of technical indicators) to predict the future price action. Herein, it is assumed that information is disseminated to the public in stages, which therefore offers the opportunity for traders which might have early access to data to spot opportunities early (Kumar, 2015). Together with the idea that the market is being driven by human psychologies and how traders interpret the charts to establish their strategy and that these psychologies/interpretation hardly change, the analysts consider that the price trends are recurring and thus predictable (Jiao & Jakubowicz, 2017). This was also found in the study conducted by Fama (1991) in which he disagreed with the random-walk hypothesis. In this study he also found that when an information event which was planned precisely and which had an extensive effect on the price was indeed quickly consumed by the average stock price and thus in contrast to his previous finding he also thus supports the efficient market hypothesis (Fama, 1991). Based on these findings it can be concluded that researchers are divided on how to interpret the market. This also indicates that there is no "one-way solution" and it indicates that the researchers do not have an accurate representation of the actual market. Nevertheless, based on the divided opinions of academics and ability of the technical analysts to outperform the market, the introduction of machine learning could offer a promising tool for improving the predictability of price actions in the stock financial market since machine learning could capture price patterns and also potentially eliminated psychological and strategy mistakes of the trader.

In machine learning, algorithms are used which are characterized by their ability to detect patterns in the provided data. These algorithms can detect long term dependencies which, according to the investors that favour technical analysis, occur in stock prices (Tsai et al., 2011). Different machine learning algorithms such as support vector machines and deep-learning methods allows to extract meaningful information from vast and large datasets by narrowing the input dataset down to a more meaningful data set to better understand the behavior of the market (Sonkavde, 2023). This thesis investigates the predictability of the future price action of the S&P500 using machine learning techniques and financial indicators. Herein, a model is created that captures various aspects of the market behavior using technical indicators and historical price data in order to reduce the bias of the model but subsequently possibly increasing the adaptability to the changing market condition. Herein, we are not seeking to predict an exact price but we are satisfied with a buy or sell signal. In Section 2 a literature study will be conducted in which a set of available machine learning techniques are showcased and explained. Based on the literature review in Section 3 the research question, goals, relevance, scope and research methodology of the research will be presented. In Section 4 the results of the study will be presented and discussed and lastly in Section 5 the conclusion of the study together with the recommendations will be presented.

2 Literature Review

2.1 Introduction

Investigating the literature is essential in order to decide which machine learning algorithm to use and how the used financial indicators are established. Therefore, this study will first start with a comprehensive overview of the available machine learning techniques that will be introduced in sections 2.2-2.7. In these sections also the in the literature found stock prediction applications, will be discussed as well as the machine learning technique its (dis)advantages. In Section 2.8, the VIX index, volatility, and trading volume as financial indicators are explained, as well as how these are/can be used in practice to predict the future price action. Lastly, in Section 2.9 some technical indicators such the VIX index and moving average will briefly discussed. Hereafter, a concluding discussion will be made towards this study will devote its academic efforts to.

2.2 Machine learning

Before diving into how machine learning could be of support in stock prediction we will first introduce what machine learning is and what kinds of machine learning types are existent. Firstly, machine learning is a subfield of artificial intelligence (AI). Herein, it focuses on enabling computers to learn from data and use its knowledge to improve

over time. Where in traditional programming, a human programmer writes code, which instructs the computer how to perform specific tasks, in machine learning, the algorithms analyze the data in which they try to find patterns after which it tries to make predictions or decisions based on that data without the interference of a human (Oracle, 2024). In order to do this, there are several learning techniques available in machine learning to make learn from data and make predictions. Below three types of learning categories are explained:

1. **Supervised learning**: In supervised learning, the algorithm is trained on a labeled dataset. Herein each data input has its corresponding target label (Oracle, 2024). The algorithm learns to map the input data to the correct output by generalizing from the given labeled data. Supervised learning is commonly deployed to make classification and regression predictions (Fumo, 2017). Therefore, it is useful to further delve into the theory since stock price prediction based on machine learning is often located in the category of regression problems due to the goal being to predict a continuous numerical value (future price of a stock), based on historical data.

2. **Unsupervised learning**: In unsupervised learning, the algorithm is given input data without explicit labels. By doing so the algorithm must find patterns in the data on its own. The interference of the data scientist is thus minimal. Typical tasks of such algorithms include clustering (grouping similar data points together) and dimension reduction (reducing the number of features in the data while preserving its structure) (Oracle, 2024).

3. **Reinforcement learning**: In this method, an agent learns to make decisions by interacting with an environment through trial and error. The goal of this algorithm is to learn to take actions that maximize cumulative rewards over time (Bhatt, 2019).

As we have learned from section 1, academics are divided about if stock prices follow a random walk or not. Prices tend to shift constantly, which makes it rather difficult to deploy a trial and error due to the output not being static. Therefore, the more interesting types of learning techniques for this study which also will be further examined in the following sections will be the supervised learning techniques. In the following sections we will delve into the integration of machine learning algorithms to forecast stock market trends, highlighting the opportunities and challenges in this dynamic and complex domain. Machine learning has emerged as a powerful tool in the domain of stock trading, revolutionizing traditional approaches to forecasting the trend of the market (Reddy, 2018). By using advanced algorithms, machine learning can be used to analyze dense datasets containing historical data of the stock market, identifying patterns within the data and to make predictions regarding future trends. In Figure 1 an overview is given of the various available machine learning algorithms. In the next sections for each category of machine learning algorithms one algorithm is chosen which will be explained based on their classification, there mechanism and their (dis)advantages.

2.3 Regression algorithms

Regression algorithms can be used as a type of supervised machine learning technique used to predict the relationship between dependent and independent variables. Herein, they utilizes historical data to learn patterns and make predictions (Enke et al., 2011). Regression algorithms are used to predict the future price of the stock by analyzing at least one attribute. Such attributes can be the closed price, open price, volume, etc. (Sonkavde, 2023). An example of a regression algorithm is a Support Vector Machine (SVM) which operates by identifying a hyperplane that separates data/observation into different classes based in patterns of information also known also features. The hyperplane can in its turn be used to determine the most suitable hyperplane for unseen data (Pisner & Schnyer, 2020). In Figure 2 an example is given of how SVM separates the data into the classes whereas the classes are separated using the hyperplane (Grigoryan, 2017/05). In the context of stock price prediction using machine learning, the aim is to build a model that can analyze historical stock price data to make predictions about future price action. It thus depends on what the desired output it of the model. The SVM is e.g. not optimal to reconginize paterns and relationships in the data making it a difficult to make actual price predictions.

Nevertheless, it was found that SVM can be useful in the context of stock price predictions since it was found that it could be effectively used as a regression model and to determine stock trends (Sonkavde, 2023). SVMs can be used to

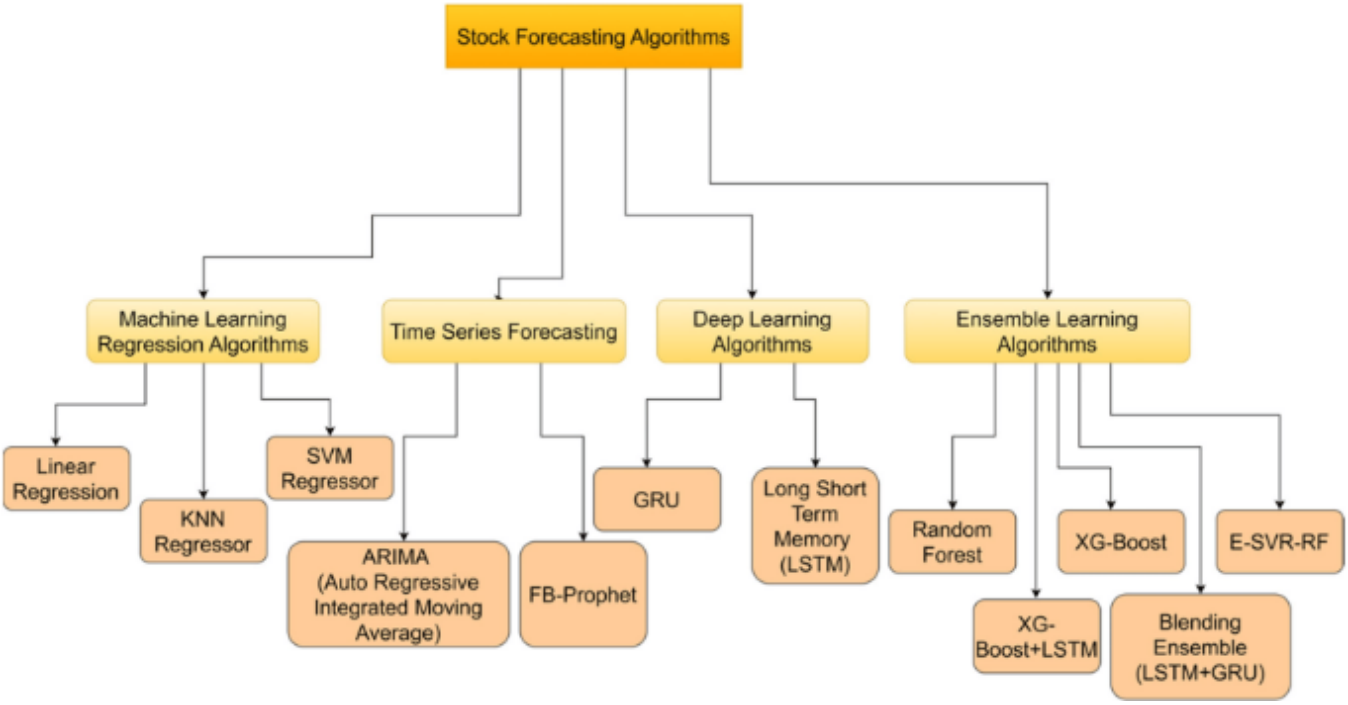

Figure 1: Overview of the different available machine learning algorithms for stock price forecasting (Sonkavde, 2023).

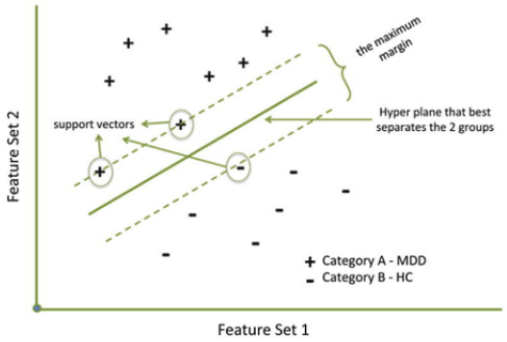

Figure 2: Figure of the hyperplane separating the support vectors corresponding to the two to be predicted classes (Pisner & Schnyer, 2020)

classify whether the stock price will increase or decrease or to predict the future stock price on a continuous scale. Herein, an advantage of SVM in stock prediction is that SVM is able to find global optimum's whereas a neural network, which will later (in section 2.6) be discussed may only find local optimums for the prediction which due to non-linearity in the price development could cause problems in the prediction. Herein, SVM solves this problem by making determination errors during the learning phase in order to minimize the error during testing whereas neural networks during training already try to minimize determination error (Madge & Bhatt, 2015). Moreover, SVMs have the ability to handle non-linear relationships in the data. However, SVMs are sensitive to noise, and the selection of appropriate parameters such as gamma and kernel functions are crucial for optimal performance (Bustos et al., 2017). Herein, the gamma parameter determines the reach of the individual support vectors where a small gamma value gives a wide reach and a soft decision boundary, which makes the model more tolerant and smoother to wrong classifications and vice versa for a high value for gamma. By choosing the "appropriate" gamma value over- or under-fitting can be prevented and is thus a crucial step in the process of setting up the algorithm. The kernel function allows the algorithm to operate in a higher dimensional space without the necessity to calculate the exact coordinates of this higher dimensional space and therefore avoids computational cost (Scikit, 2024).

2.4 Time series forecasting

Time series analysis in machine learning examine and forecast data points which are chronologically ordered. Within the data the algorithm tries to find temporal trends, patterns and/or seasonality. Stock price data are considered to be continuous. Therefore, it is worthwhile to examine this method (Sonkavde, 2023).

The Auto regressive integrated moving average (ARIMA) is a model that uses time series forecasting to predict future stock prices. ARIMA is a form of regression analysis which measures the strength of a single independent variable relative to changing independent variables. Herein the AR term refers to the model that show the independent variable regressing on its own variables. The I term refers to the differentiation of the raw data points which allows the time series to be stationary and therefore removes trends from the data, removing trends is an important step since trends that are not removed could be picked up by the AR component of the model. This component in its turn can then dominate the model making it difficult to capture other patterns and make accurate predictions. Lastly the moving average (MA) term refers to the usage of the dependency between the observation and the residual error from the moving average to on lagged observations in other words the MA component accounts for the random shocks or short-term fluctuations in the time series that cannot be explained using the AR part. Therefore, 3 different parameters are used namely, p, d and q. The p parameter is the order of the AR component. The p parameter can be determined by observing the partial auto-correlation function (PACF) that measures the correlation between the current and past observations while excluding the intermediate lag (Hayes, 2023). Herein, the outstanding spikes in plot at specific lags could potentially indicate values that can be used as the p parameter. The d parameter is the order of differencing required to make the data stationary and the q parameter is the order of the MA component and signifies the number of lagged observations of the residual errors to be included in the model. To determine the d parameter we can use the Augmented Dickey-fuller (ADF) test to test the stationarity of the data (edX, 2023), where adifferencing is applied and the p-value of the test is checked. If it is less than the set significance level then it can be concluded that stationarity has been achieved. The q parameter can be determined by analyzing the auto-correlation function (ACF) plot of the differenced time series. Herein, spikes at certain lags could indicate values for that can be used as the q parameter (Hayes, 2023). According to previous studies it is demonstrated that the ARIMA model fits the stock market index due to it nature of first identifying, estimating and lastly diagnosing the data. As such the algorithm can be used to forecast any financial market (Devi et al., 2013; Khanderwal & Mohanty, 2021).

In order to increase the capabilities of the ARIMA algorithm, the algorithm can be combined with another algorithm called symmetric generalized autoregressive conditional heteroskedasticity (SGARCH) (Sonkavde, 2023). SGARCH is based on two models; the Autoregressive Conditional Heteroskedasticity (ARCH) and the Generalized Auto-regressive Conditional Heteroskedasticity (GARCH) of which the GARCH model is an addition to the GARCH model. Herein the ARCH model is used to capture the time-varying volatility in financial returns. It assumes that the conditional variance of the series is a function of past squared observations, whereas the GARCH model builds on the ARCH model by including the lagged conditional variances combined with the lagged squared observations (Engle, 1982). It introduces an auto-regressive component to capture the persistence of the volatility (Bollerslev, 1986). Finally, the SGARCH model elaborates on the GARCH model by introducing a stochastic (random) component into the conditional variance equation. The random component in the variance equation aids in capturing the random fluctuations in the volatility and therefore could improve the prediction performance. The SGARCH algorithm was modeled and back tested on the S&P500 index. Herein, it was found that the algorithm was able to outperform the benchmark performance of BUY&HOLD the S&P500. Therefore, SGARCH is an effective measure to predict the price trend since the model was more efficient than the market (Vo & Ślepaczuk, 2022). Nevertheless, due to this algorithm being vulnerable against lag, it could result in a large number of parameters to be estimated if too much lag is present. In general for a increasing number of parameters to be estimated the difficulty tends to increase which cause a higher time consumption to set up the model and could eventually deteriorate the accuracy of the prediction made by the model. Moreover, this algorithm is rather unable to react to relatively large price shocks (sudden large change in trend) since it assumes that these shocks should show the same effect on the volatility meaning sudden drops and rises (Vo & Ślepaczuk, 2022). Thus, although SGARCH has shown a promising performance by being able to outperform benchmark strategies, it is not immune to challenges such as parameter estimation complexity and limited responsiveness to sudden price shocks. These methodologies are critical in stock prediction. Therefore, we will continue the explo-

ration of other promising machine learning techniques.

2.5 Deep learning

Deep learning models are recognized across different domains in science and engineering. Their recognition is also observed in the domain of stock price forecasting/trend prediction. This is due to their ability to capture complex patterns, managing dense datasets, engaging in feature learning and representation, and their ability to adapt to dynamic market conditions (Sonkavde, 2023).
Long short term memory (LSTM) is an example of a deep learning method and which is based on an advanced recurring neural networks (RNN) architecture. The advantage of this algorithm is that it is able to handle long historical data sequences while avoiding vanishing/exploding gradients which can be occurring in traditional RNNs. (R)NNs are based on weights, biases and activation functions in which information passes through by using data from previous data points. Herein, during back propagation in the activation function the gradients are calculated which are used to update the parameters of the model. However, due to the constant weights and biases being multiplied with the gradient it could cause that overtime the gradient decreases or increases exponentially for the increasing number of layers. This could subsequently lead to the parameter's of the model overtime will be minimally updated. This limits the ability of the model to learn long-term dependencies. This results in the model being rather complex and potentially causing problems such as over-fitting when the available training data is insufficient. Moreover, the required computational power subsequently increase within increasing the complexity of the model which potentially could cause memory problems (Srivatsavaya, 2023).
The power of the LSTM is found in its capability of managing lengthy sequences of data, and thus can for example be used for stock price time series. Unlike other types of neural networks, LSTMs have memory cells and gates, which allow the model to retain relevant data for a extended period of time while also updating the model with "current" data. In Figure 3 an example of an LSTM cell is given whereas it can be seen that it consists of 3 gates, namely, the input gate, the forgotten gate and the output gate (Sonkavde, 2023). The outputs of every gate are determined by sigmoid activation functions which controls the adaptability of the model to changing market. Herein, the output of the forgotten gate determines how much of the long term memory should be forgotten, the input gate updates the long term memory and the output gate updates the short term memory (Siami-Namini & Namin, 2018). Since it is assumed that for stock trading both historical and recent data is important, this model could be well-suited for predicting stock trends (Ahmed et al., 2023). Nevertheless, the capabilities of LSTM networks, including their adaptability in capturing complex patterns, managing extensive datasets, engaging in feature learning, and adaptability to dynamic market conditions, makes LSTM a promising method for making accurate stock predictions. Moreover, LSTM networks are able to address critical challenges encountered in traditional recurrent neural networks (RNNs), such as vanishing/exploding gradients, which enables a more effective learning method for learning long-term dependencies and coping with issues such as over-fitting. These attributes show the suitability and potential of LSTM networks being deployed in order to make accurate and reliable stock prediction models.

2.6 Ensemble Learning

Ensemble learning techniques use a series of classifiers. Classifiers are algorithms that orders data into classes (Bi et al., 2019). The classifier tries to recognize and classify patterns in the data and tries to draw a conclusion on how these patterns should be labeled (IBM, 2024b). In the context of stock price prediction the ensemble learning serves the purpose of improving prediction accuracy and robustness by combining the predictions of multiple individual models. By leveraging the predictions of multiple models, ensemble learning reduces the risk of over-fitting the data and increases the ability to generalize the data, thus providing investors and analysts with more accurate insights for informed decision-making in financial markets (Mehta et al., 2019). An example of such a classifier is a decision tree, with its predictions contributing to determining the most favourable outcome. The most popular ensemble methods include bagging and boosting. In bagging a random sample of data is selected from a training set with replacement, allowing individual data points to be chosen more than once. Subsequently, multiple data samples are generated, resulting in individually trained data models. Depending on the classification the average or majority of these predictions provides a more precise prediction. This type of machine learning is widely used to cope with variance in datasets which are

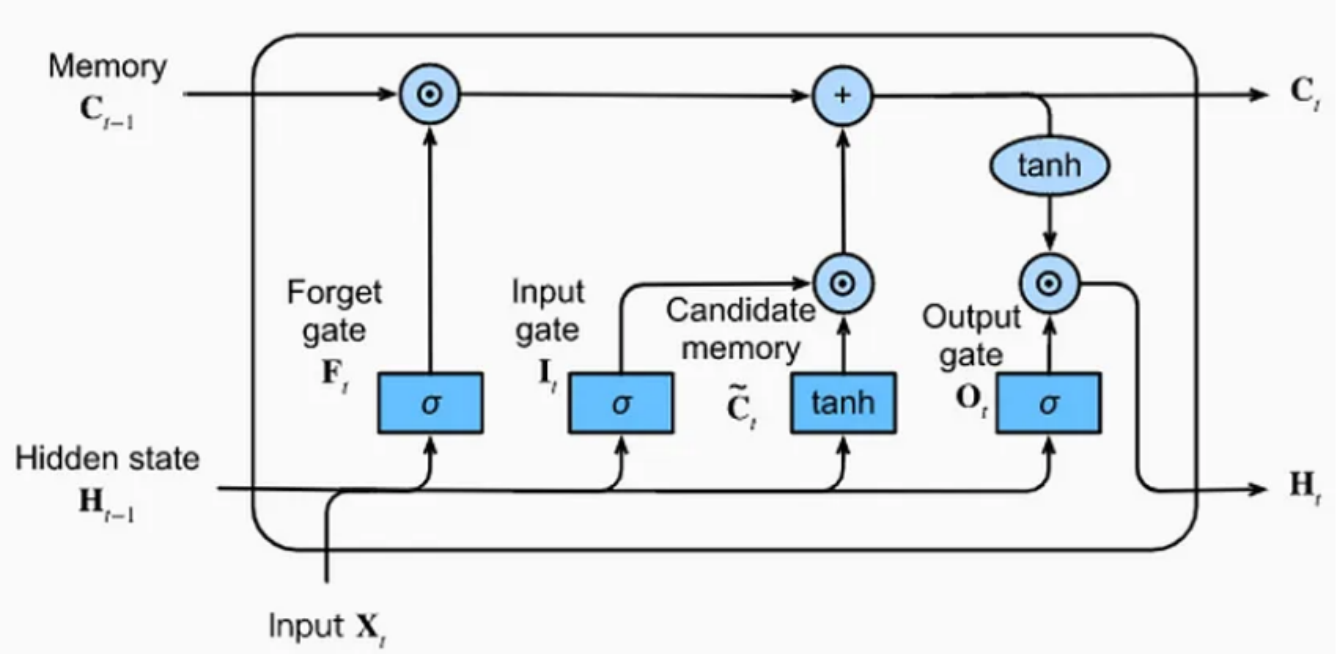

Figure 3: Representation of the composition of an ordinary architecture of an LSTM Unit (Calzone, 2018)

known to contain noise (IBM, 2024a).
A well known Ensemble learning method is a Random forest. Random forest is a learning method located in which finds its efficiency in solving classification and regression problems (Sonkavde, 2023). The random forest algorithm is a concept which bases its predictions on decision trees (Sadorsky, 2021). In figure 4 a random forest is shown, herein it can be seen that the random forest follows the following generic steps to make a prediction:

1. N random data points are picked in chronological order to avoid leakage (test data being used as training data).

2. A decision tree is built based on the N inputs.

3. The number trees that will be considered is determined. This step is rather important since this determines the processing time, memory usage and the average area under the receiving operating curve (ROC) curve (AUC) and its optimum value is found to be between 64-128 trees (Oshiro et al., 2012). The ROC and AUC performance measures are explained below.

4. Prediction (Sonkavde, 2023). This is done for a regression problem, by averaging the individual decision trees, whereas for a classification problem, the most frequent categorical variable will be used as the predicted class. after which, the out-of-bag sample is then used for cross-validation, which finalizing the prediction (IBM, 2024a).

An ROC plot is a graph which shows the performance of a classification model across different classification thresholds. It plots two parameters, the true positive rate (TPR) and the false positive rate (FPR). The TPR is also known as the recall which is defined as the fraction of the true relevant instances over all relevant retrieved instances, whereas the FPR is defined as the fraction of the not true relevant instance over all relevant retrieved instances. The ROC graph plots the TPR against the FPR at different classification thresholds. The points on the ROC curve are computed by evaluating a logistic regression model several times with different classification thresholds. However, this method is highly inefficient therefore, AUC can be used. AUC measures the entire area under the ROC curve and therefore provides an aggregate measure of the performance across all possible classification thresholds (Google, 2024). To determine the number of trees to be picked the ROC curve should be examined by identify the number of classifications required until the AUC does not show a significant change anymore. By lowering the classification threshold, the algorithm classifies more items as positive, which can increase both the ability of the random forest to handle large, noisy datasets contributes to the robustness of the performance in the prediction of complex domains such as the stock market (Pavan Kumar Illa, 2022). However this could also mean

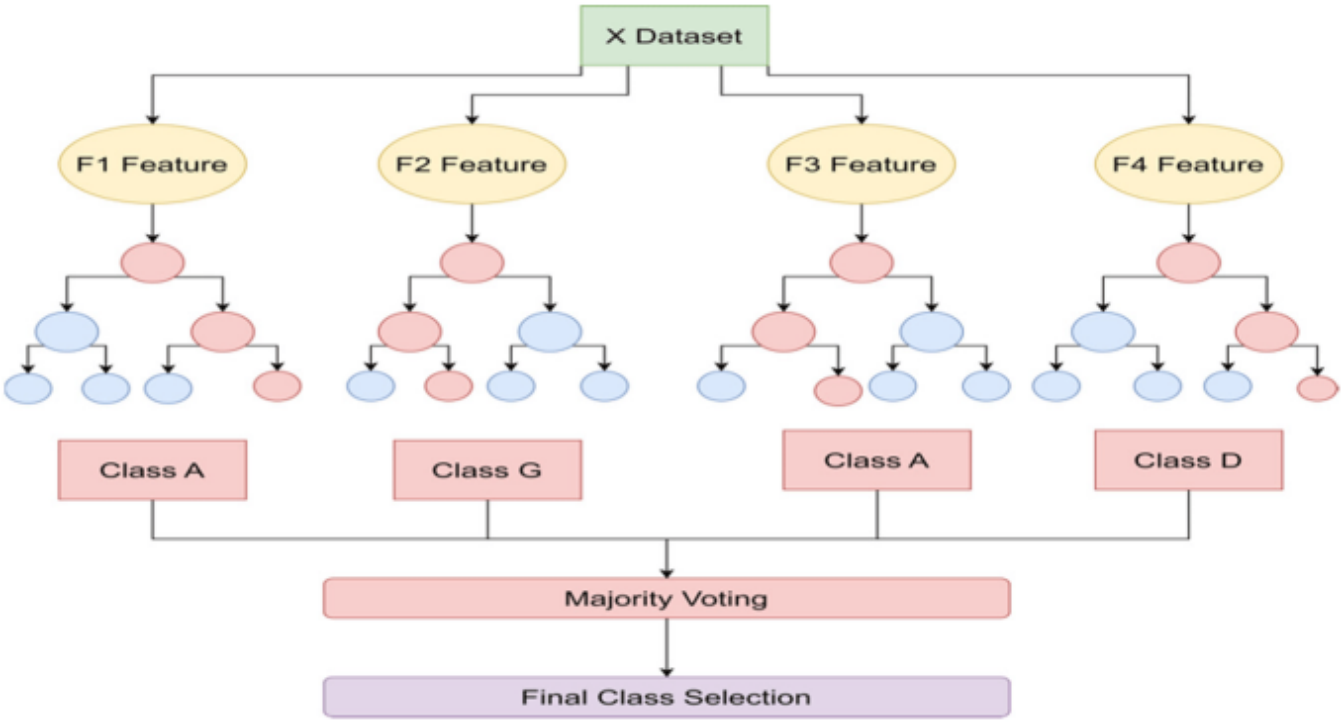

Figure 4: Example of a random forest (Sonkavde, 2023)

that since also the number of false positives are increasing the accuracy of the model is lost (Google, 2024).
In addition to this disadvantage, an advantage of using a random forest is that it is possible to observe the importance of the features included in the prediction. This gives valuable insights of which features to include and which could potentially be discarded to decrease the complexity of the model, reduce the required computational power, and reduce noise. However, a disadvantage could be that due to the increasing complexity over time in which the decision tree becomes larger the computing time to make a prediction drastically increases (Donges, 2023). Moreover, since a random forest uses several decision trees the model is more complex and more difficult to interpret than a model based on a single tree (IBM, 2024a).

2.7 Financial indicators

In order to train the Machine learning model, besides the return series, financial indicators of the index can be used to train the model with data to make more accurate predictions than if the model is only fed with raw price data. In the following sections, the indicators VIX index, moving average, and trading volume will be discussed how it can aid the model.

2.7.1 VIX index

The volatility index (VIX), also known as the fear index, was first introduced in 1993 by the Chicago Board Options Exchange (CBOE). The VIX index indicates the market's expectations for the relative strength of near-term price changes of the S&P 500 Index in which it generates a 30-day future projection of volatility, since it is derived from the options on the S&P500 which are set to expire in a period of 23 to 37 days (Kuepper, 2023). Herein, only the "out of the money" options, which are options which when triggered have no effect for the option holder, are included in the calculation and should therefore, give more insight about future volatility (Giro, 2024). This index is widely recognized by traders and investors since it is considered that this index can both show the market's volatility expectations and reflect the investor sentiment and risk aversion (Ahoniemi, 2008). Therefore, the VIX index thus can be a be a promising technical indicator to be observed to be of aid in the traders' risk management.

2.7.2 Moving average

The Moving average (MA) is a widely used technical indicator used in technical analysis, which provides the average value of the price change of a stock or index. In this paper, a rising trend is found when prices move beyond the MA and a downtrend

is found when the price falls below the MA (Hari & Dewi, 2018). The advantage of the moving average is that it offers smoothed line of the price history which is less prone to temporary price swings. Thus, when trying to make long-term (30-day) predictions, the moving average could give rather insightful information. However, due to its tempering behavior, the MA is slow to detect rather rapid price changes that often appear at market reversal points, which could lead to lag in prediction (Maverick, 2022).

2.7.3 Trading volume

The trading volume is the total number of shares / contracts exchanged between buyers and sellers for a specific security during trading hours on a given day. Here, when the security is actively traded, the volume is high, and vice versa. Herein, the price is more like to change more frequently when the volume is high. The volume thus gives information regarding the activity of traders on the security and the amount of liquidity present. High volumes could indicate specific trading catalysts (Twin, 2022). In this paper, previous studies argue that volume induces price changes because more investors are influenced to trade in the same direction. However, this argument would make the statement that the price implies a random walk invalid (Brailsford, 1996). Since traders were able to make predictions about the trends of the stock market by using technical analysis and previous studies indicating that the random-walk behavior of the market could be invalid. It could be useful to use the trading volume as a feature in the random forest model to both spot long and short term trends in the volume of the stock. By using the trading volume as an input for the random forest, the model enrichment could help the model make the prediction of the future stock price which could mean that an opportunity for enhancing predictive accuracy and gaining insights into market dynamics and potential price movements has been found.

2.8 Discussion

Despite considerable academic efforts to implement machine learning models such as Random Forest and LSTM for financial forecasting, and to explore the impact of individual financial indicators like the VIX index, moving averages, and trading volume on stock price prediction, researchers have yet to discover a highly accurate model. Furthermore, the potential benefits of combining these indicators across models have received little attention, particularly in how they might complement each other and improve predictive accuracy.

A noteworthy aspect of technical indicators is the role of the VIX index, which has been argued to help traders gauge market sentiment, thus enabling models to form a more informed bias toward predicting positive or negative market trends (Giot, 2005). Additionally, moving averages are frequently used to smooth out short-term price fluctuations, while trading volume highlights the strength of price movements. The combination of these three indicators offers the opportunity for models to capture both market sentiment and technical price patterns, providing a more comprehensive dataset that could enhance prediction accuracy.

Random Forest models are well-suited for handling noisy or high-dimensional financial data, making them effective for identifying non-linear relationships among the VIX index, moving averages, and trading volume. On the other hand, LSTM models are particularly adept at predicting time-series data, making them ideal for recognizing sequential dependencies in S&P500 price movements. By integrating these models, their complementary strengths can be leveraged to improve predictive performance.

A significant gap in the literature lies in the fact that prior studies have largely overlooked the inclusion of the VIX index as a technical indicator in machine learning models, particularly those combining Random Forest and LSTM. Moreover, few studies have explored the potential of using the output of a Random Forest model as an input for an LSTM model. In response to these findings, Section 3 of this paper will introduce the study's contribution to the literature by addressing these gaps.

The combination of Random Forest and LSTM is proposed due to the unique strengths of each model: Random Forest excels in managing non-linear relationships among multiple financial indicators, while LSTM is highly effective at capturing temporal patterns. By integrating these models, a more holistic approach is achieved, where the weaknesses of one model are compensated by the strengths of the other.

This decision to combine Random Forest and LSTM stems from the limitations identified in previous research, where single models have struggled to capture both complex non-linear relationships

between financial indicators and the temporal patterns of stock price movements. By employing a hybrid model, this study aims to overcome these challenges, drawing on recent advancements in hybrid model design for financial forecasting.

3 Problem Statement & research methodology

This study aims to contribute to research of stock price prediction by investigating the predictability of the future price action of the S&P500 using machine learning techniques, where the VIX index, volatility, and trading volume are used as the financial indicators for the model to predict the future price action. Herein, the VIX index is chosen since the VIX index reflects the volatility of the market. This data could be of great use to the model since the volatility could show reversal in the market trend because high volatility often mirrors downtrend in the market which could aid the model mitigate risk (Kuepper, 2023). On the other hand the moving average smooths the price movement by filtering out the noise due to random large short-term price fluctuations (volatility) and thus revealing the underlying trend of the market (Mitchell, 2023a). Lastly, the volume of the market will be used since the volume indicates the number of shares or contracts being traded. This thus serves as an indicator of the strength-/activity of the market (Mitchell, 2023b). Unusual volume spikes could give the model more information about long and/or short term sentiments of the market (Plachý, 2014). The idea is to use the three indicators together with the ordinary price data of the S&P500 to feed the model and thus not only an average indication of the market trend, but also potential trend reversal indicators. Potentially being able to early spot reversal in the trend of the market by using the VIX index, confirming of supporting the idea using the volume data, while benefiting from the fluctuation noise reduction using the moving average. The model will thus perform an automated technical analysis by using the VIX index, moving average, trading volume and the historical prices to form price trend predictions. In summary, capturing various aspects of the market behavior in order to reduce the bias of the model but subsequently possibly increasing the adaptability to the changing market condition. As such the following research question is constructed: "Does the combination of the VIX index, moving average, and trading volume, as the financial indicators, affect the predictive accuracy and trading performance when used in a combined random forest and LSTM model, compared to a BUY&HOLD strategy, individual Random Forest and LSTM models and a random buy heuristic?"

Herein, we divide the main question into subquestion:

- How does the combination of the VIX index, moving average, trading volume, and LSTM predictions as inputs for the Random Forest affect the predictability of future stock price movements? How does this compare to using these inputs individually in the Random Forest model?
- What is the predictive performance of the combined Random Forest and LSTM model when compared to individual Random Forest and LSTM models? How does the hybrid model leverage the strengths of each to improve prediction accuracy?
- How does the combined model's trading performance compare to a traditional BUY&HOLD strategy? Does the model lead to better financial outcomes in terms of profitability and risk management?
- How does the combined model compare to a random buy heuristic in terms of predictive accuracy and trading performance?

3.1 Objectives

The objectives of this research are

1) *Determine if the proposed combined model consisting of the random forest with the LSTM predictions, VIX index, moving average, and trading volume as features of the Random forest leads to an enhanced performance in prediction compared to standard LSTM and random forest prediction models with moving average, and trading volume as features. Furthermore, to determine if the combined model leads to a higher return on investment than traditional BUY HOLD strategy. This will be assessed over the testing period from January 1, 2010, to January 1, 2024.*

2) *The proposed model is able to have a higher performance in terms of the percentage of number of correct predictions made compared to the percentage of correct predictions of the price trend of the S&P500 using a random*

buying/selling heuristic over the testing period which starts on the January 1st 2010 and range till January 1st of 2024.

3.2 Relevance

This study explores the impact of using the VIX index, moving average, and trading volume as financial indicators in a hybrid machine learning model composed of Random Forest and LSTM. By investigating the combination of these specific financial indicators in the combined random forest and LSTM model, this research adds to the academic literature on financial forecasting. Although many studies have focused on individual models and/or indicators, this study provides insights into how combining both machine learning models (Random Forest and LSTM) and different financial indicators (VIX index, moving averages, trading volume) affects the predictabillity. This study fills a gap in the existing literature by offering a new approach to integrating market sentiment, technical indicators, and machine learning, potentially advancing the understanding of hybrid model performance in financial contexts, where if the model in fact does shows an improved predictabillity, it could have significant real-world applications for portfolio management and investment strategies. Accurate predictions of stock price movements using this model could enhance the performance of investment portfolios by informing better buy/sell decisions, thus optimizing returns while managing risks. In practice, financial institutions, investors, and algorithmic traders could leverage the findings to develop more robust trading strategies that outperform traditional methods such as the BUY&HOLD strategy or random heuristics.

3.3 Scope & limitations

This study focuses on the effectiveness of introducing the VIX index along with the moving average and trading volume in machine learning to accurately predict the stock trend of the S&P500. Other technical indicators will not be considered. With respect to the machine learning algorithms used, this study will limit it self to the use of only random forest and LSTM learning methods to predict the future trend of the S&P500 since a combination of the 2 learning techniques has not been tested yet in the literature. Furthermore, in this study we are not seeking to predict an exact price but we are satisfied with a buy or sell signal.

3.4 Methodology

In order to investigate the effectiveness of the technical indicators such as the VIX index on the stock price prediction of the S&P500 using machine learning. The first step of this study is to collect the required data. This data will originate from yahoo finance (Yahoo, 2024). From this website we will import the historical data of not only the price but also the technical indicators and the VIX index. The period that will be used starts from January 1st 1990 until January 1st of 2024. Of this data the tradeable days between January 1st 1990 and January 1st 2010 will be used as the training set for the models remaining part of the data will be used for back-testing the model. From the data stock splits and dividends will be discarded as well by using the adjusted closing price. The data will be imported to the python file in which the random forest and LSTM are modelled. Hereafter, we will establish a LSTM which is trained on historical prices of the S&P500. This model will make based on the historical price data a prediction of the trend for the price difference between today's price and tomorrow's price. Herein, the predicted price difference is translated to a downward market trend in which the output of the model is 0 (sell) or an upward market trend in which the output of the model is 1 (buy). These actions will be used next to the historical price data, moving average, trading volumes and corresponding VIX index of the S&P500 as the input for the random forest. The random forest will predict the next day price price difference which will be translated to a 0 value if the difference is negative or 1 value if the difference is positive in which a 0 is a downtrend prediction and a 1 is a uptrend prediction. The performance of the model will be determined by comparing the returns made from the prediction by the random forest with the returns of a simple BUY&HOLD of the S&P500. The BUY&HOLD is used as a baseline for the return. This method is the easiest trading method for an investor since it requires no skill. Therefore, the model becomes interesting if it can "beat" the market in other words outperform the BUY&HOLD of the S&P500. Herein, we compare the returns made for both methods. Moreover, we will also compare by only observing the performance of the random forest and LSTM. The LSTM is not affected by the other random forest and thus the performance can directly be observed. For the random forest a model will be established which is the similar

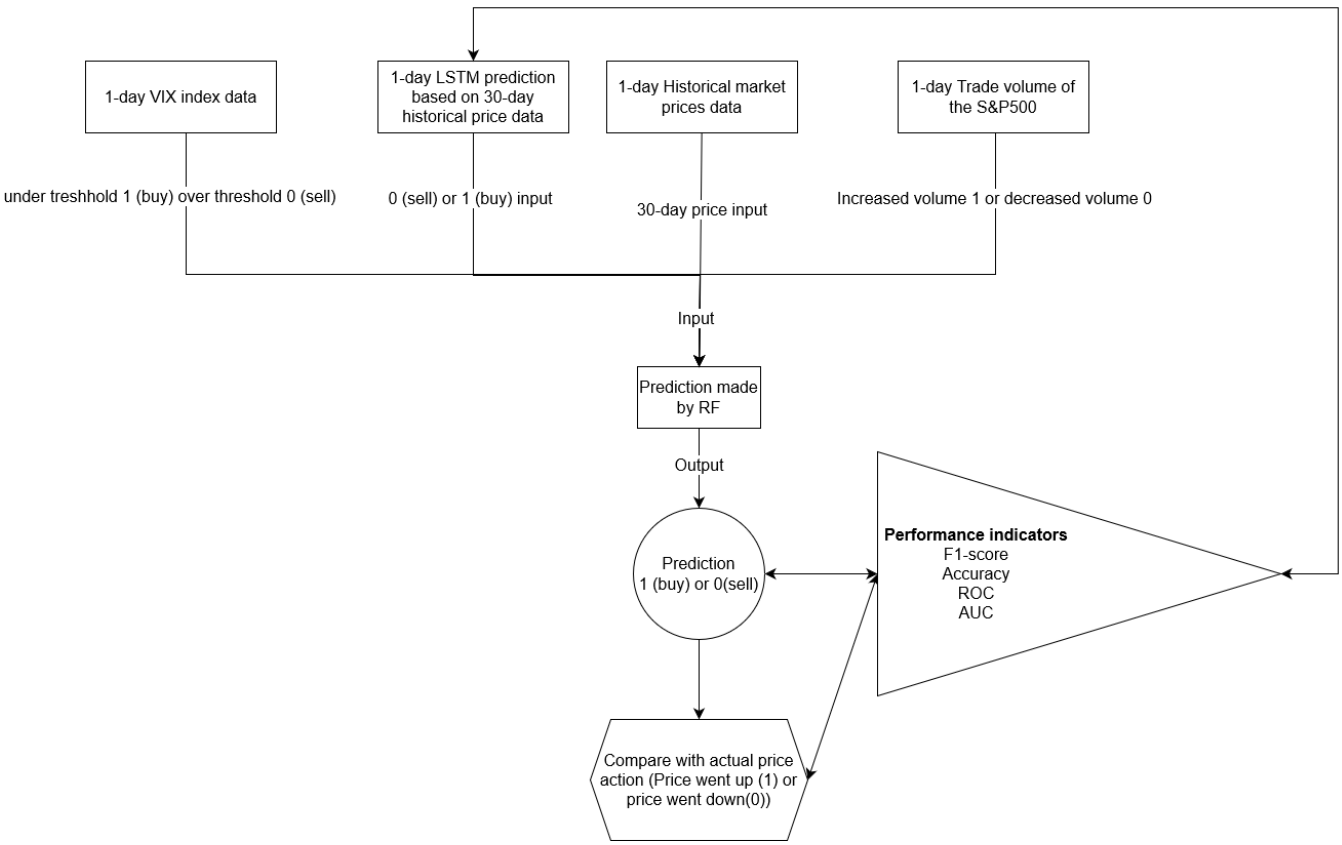

Figure 5: Experimental overview

to the combined model however in this model the input of the LSTM is discarded.
The predictability performance of the models will be examined by observing the F1-scores of the models and by providing the standard deviation of predictions we can show also determine the fluctuation of the model. Since we determine the F-scores we also determine the precision and recall of the model since the F1-score is based on these two. The precision determines the proportion of correctly predicted predictions out of all predictions. The accuracy determines the overall correctness of the model's predictions. Recall measures the ability of the model to capture all positive instances by taking the fraction of correctly predicted observations over the total of actual positives. The F1-score is on its turn the harmonic mean of the precision and the recall and will primarily be used since the recall and precision can than simultaneously be used (Fehst et al., 2018). For the random forest, also the ROC and AUC will determined. Based on these measures the performance of the model will be judged. Beneath the methodolgy is shown in a stepwise manner for the different models.

LSTM model

1. Collect the adjusted closing price historical price data, volume and VIX index data of the S&P500 from Yahoo finance ranging January 1st of 1990 till January 1st of 2024. The data ranging from January 1st of 1990 till January 1st of 2010 is used as training data and the data ranging from January 2st of 2010 till January 1st of 2024 is used for testing.

2. Model an LSTM model which predicts the next day's price difference of the S&P500 using the adjusted closing price, volume, moving average and (VIX index).

3. Determine the predictive performance of the model by determining the F1-score, accuracy, precision, recall and mean squared error (MSE).

4. Translate the predicted price difference to a 0 (sell) if the difference is negative or a 1 (buy) if the difference is positive.

5. If a 1 signal is given the model should buy 1 stock at the current market price and sell at the end of the day at the closing price. The return is difference in stock price on that day.

6. Sum all buy signals returns to obtain the fi-

nal return made by the model over the testing period.

7. Compare the return of the model against a buy&hold and a random buy heuristic and the other models.

Random forest model

1. Collect the adjusted closing price historical price data, volume and VIX index data of the S&P500 from Yahoo finance ranging January 1st of 1990 till January 1st of 2024. The data ranging from January 1st of 1990 till January 1st of 2010 is used as training data and the data ranging from January 2st of 2010 till January 1st of 2024 is used for testing.

2. Model random forest model which predicts the next day's price difference of the S&P500 using the adjusted closing price, volume, moving average and (VIX index).

3. Determine the predictive performance of the model by determining the F1-score, accuracy, precision, recall and area under the curve (AUC).

4. Translate the predicted price difference to a 0 (sell) if the difference is negative or a 1 (buy) if the difference is positive.

5. If a 1 signal is given, the model should buy 1 stock at the current market price and sell at the end of the day at the closing price. The return is difference in stock price on that day.

6. Sum all buy signals returns to obtain the final return made by the model over the testing period.

7. Compare the return of the model against a buy&hold and a random buy heuristic and the other models.

Combined Random forest and LSTM model

1. Collect the adjusted closing price historical price data, volume and VIX index data of the S&P500 from Yahoo finance ranging January 1st of 1990 till January 1st of 2024. The data ranging from January 1st of 1990 till January 1st of 2010 is used as training data and the data ranging from January 2st of 2010 till January 1st of 2024 is used for testing.

2. Model an LSTM model with a rolling horizon which predicts the next day's price difference of the S&P500 using the adjusted closing price, volume, moving average and (VIX index) over a horizon of January 1st of 2004 till January 1st of 2024. The rest of the predictions from January 1st of 1990 till December 31st of 2003 will be 0 values.

3. Model random forest model which predicts the next day's price difference of the S&P500 using the adjusted closing price, volume, moving average, (VIX index) and the LSTM predictions.

4. Determine the predictive performance of the model by determining the F1-score, accuracy, precision, recall and area under the curve (AUC).

5. Translate the predicted price difference to a 0 (sell) if the difference is negative or a 1 (buy) if the difference is positive.

6. If a 1 signal is given the model should buy 1 stock at the current market price and sell at the end of the day at the closing price. The return is difference in stock price on that day.

7. Sum all buy signals returns to obtain the final return made by the model over the testing period.

8. Compare the return of the model against a buy&hold and a random buy heuristic and the other models.

In conclusion, in this study a combined model is established in which an LSTM model's predictions are used as a feature input for a random forest along side other features. The random forest makes predictions of the next days price difference of the S&P500 based on historical price data.

4 Results

In this chapter the results of the proposed models are presented. This section is structured as follows: First, in section 4.1 the performance of the standalone models LSTM and Random Forest in predicting the future price changes of the S&P500 is evaluated. Then, in section 4.2 the performance of the combined model, where LSTM predictions are integrated into the Random Forest model as input features is presented. And finally, in section

4.3 we compare all models (LSTM, Random Forest, and the combined model) against the benchmark strategies namely the Buy&Hold (B&H) and the Random Buy (RB) heuristic. Throughout this chapter, a distinction must be made between two key concepts of predictions, namely classification and regression. This also affects the type of used key performance indicators (KPIs) to evaluate model performance. Here, for a classification in which the focus is on predicting if the price of the SP500 will go up or down, which is essential for making buy/sell decisions, the following KPIs are used:

- Accuracy: Measures the percentage of correct directional predictions (up or down movement).
- Precision: The proportion of predicted uptrends that were correct (true positives / (true positives + false positives)).
- Recall: The ability of the model to correctly identify actual uptrends (true positives / (true positives + false negatives)).
- F1-Score: The harmonic mean of precision and recall, representing the balance between these two metrics.

While for regression the focus is on predicting the actual price or the price difference. While this is more detailed, it is often more challenging and sensitive to volatility. Therefore, the following KPI is used to evaluated the perfomance:

- Mean Squared Error (MSE): The average squared difference between predicted and actual prices, which is used for regression tasks.

Lastly, all models are examined by the return. The return measures the financial outcome if the model's predictions were used for trading (buy on predicted uptrend, sell on predicted downtrend). The return is modeled as when the model predicts a buy signal fictively 1 share at the at that time current market price is bought and sold at that days closing price. This continues over the full testing period. Hereafter, all returns (negative and positive) are summed which results in the final return of the model. In this chapter we focus primarily on classification metrics (accuracy, precision, recall, and F1-score) for evaluating the models' ability to predict market direction. However, we also assess regression performance using MSE as a supplementary metric to understand the accuracy of price-level predictions. Furthermore, the returns of the models are examined to observe the financial efficiency of the models.

4.1 LSTM solo

With the establishment of the solo LSTM we try to make predictions in price change for the next day for the S&P500. Since we would like to observe the effect on the predictability of a random forest of integrating the predictions of an LSTM in the input of the random forest we first have to observe the predictability of a LSTM model. We established a model in which we used historical price data of the S&P500 from 01-01-1994 until 01-01-2024 of which 80% was given to do model as "training" data, while the remaining 20% was used for testing the predictability of the model. In order to further enhance the predictability we used the following features:

- price difference defined as *today's price – yesterday's price*
- volume difference defined as *today's volume – yesterday's volume*
- 20 day exponential moving average
- 60 day exponential moving average
- 90 day exponential moving average
- 120 day exponential moving average
- 150 day exponential moving average

By targeting the next days closing price, we obtained the following results which are compromised in figure 6. In this figure, we can clearly observe a trend recognition by the LSTM with an accuracy of 78.96% with a mean squared error of 0.504. The LSTM was able to rather copy the chart of the S&P500. However, by further observing the data points and chart we observed something interesting. The graph of the LSTM predictions seemed to be a copy of the graph of the S&P500 however with a lag of few days. Since the increase and decrease of the absolute prices per day are absolutely seen rather small (difference in price compared to base price) the LSTM could make a prediction that is close to the price of the previous day which will in a first observation look promising. Since the price difference per day is relatively small compared to the base price if the LSTM makes an off prediction. However, the prediction is close to the base price

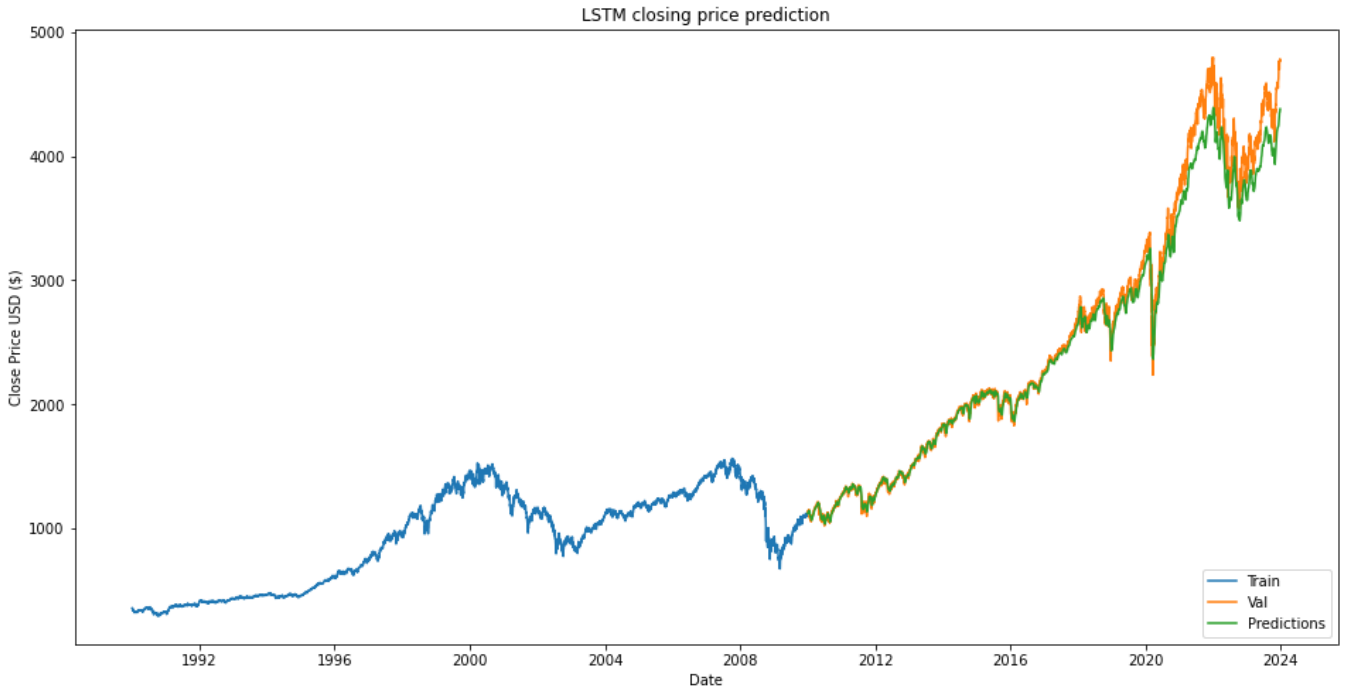

Figure 6: LSTM closing price prediction

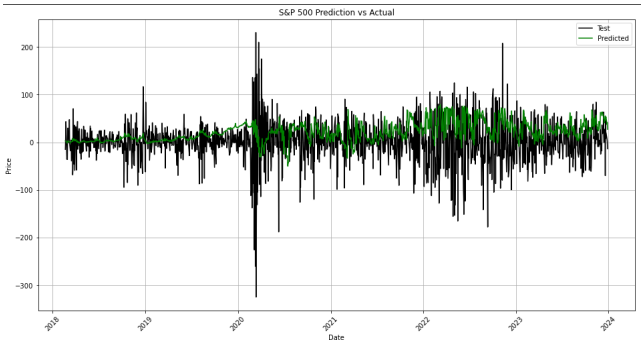

Figure 7: Daily LSTM predictions of the price difference and S&P500 actual price difference per day over the testing period.

the mean squared error will return as small. Possibly meaning that the LSTM is just extrapolated historical prices, rather than making actual accurate predictions of future price changes. In order to observe this the prediction target was changed from close price to close price difference in which close price difference is defined as *tomorrow's closing price - today's closing price.* Herein, the same LSTM model and inputs as was used for the closing price prediction however, we re-examined the hyperparameters. By tuning those we used the following parameters:

- **LSTM Layer Hyperparameters:**
  - **Units (LSTM Neurons):** 180
- **Dense Layer Hyperparameters:**
  - **Units:** 1
- **Input Sequence Length:**
  - **Backcandles:** 30
- **Training Hyperparameters:**
  - **Batch Size:** 15
  - **Epochs:** 30
- **Optimizer:**
  - **Optimizer:** Adam
- **Loss Function:**
  - **Loss Function:** Mean Squared Error (MSE)
- **Validation Split:**
  - **Validation Split:** 0.07
- **Feature Scaling:**
  - **Scaler:** MinMaxScaler with feature range (-1, 1)
- **Random Seed:**
  - **NumPy:** 42
  - **TensorFlow:** 42
  - **Random:** 42
- **Dataset Split:**

– **Train/Test Split:** 80% training, 20% testing

By using those parameters the model obtained its best results. In figure 7 the prediction made by the LSTM on the price difference against the actual price difference of the S&P500 (in the graph depicted as test) is displayed. Herein, it can be seen that the LSTM makes quite restrained prediction, since the predicted absolute price differences are small. Furthermore, it can be observed that when the market had an increased volatility the LSTM tried to adjust but still the predicted price change were still always smaller than the actual market. This may be caused by the loss function of the model which could be argued to have a too small sensitivity to catch larger differences. These points however, should not be a problem for this study since we are only interested in the model being able to correctly spot a positive or negative future price change. However, it is clear from our observations that the LSTM struggles to accurately predict the absolute future price difference. This is further emphasized by evaluating the performance metrics of both models, as shown in Table 1. Here, it becomes evident that the performance of the LSTM predicting the price difference was significantly lower than that of the LSTM predicting the closing price.

Ultimately, the LSTM model predicting price differences achieved an accuracy of 52.68% with a mean squared error (MSE) of 0.0356. These results suggest that while the model demonstrates some predictive capability in identifying market trends, it may primarily be extrapolating historical data rather than making accurate predictions about future price changes.

4.2 Random forest solo

The "solo" random forest was established to serve as a benchmark performance to observe predictability on next day's price change of the S&P500 without introducing the VIX-index and LSTM predictions. Herein, the same features and training and testing periods are used as in the solo LSTM model. For the hyper parameters of the model we have used 100 estimators, no maximum layer depth and 100 minimum sample splits. The combination of the features and these hyper parameter settings resulted in the performance displayed in table 1 column RF. Herein, we find a baseline performance in which we have an accuracy of 53.7%, a F1-score of 63% and a projected return if the 1 calls were bought and sold at the end of the day, of 572.30$. These results are relatively poor compared to the LSTM performance. However, it must be considered that the next day's price prediction of a stock is better considered as an regression model rather than a solely classification problem. Therefore, in order to try to improve the performance of the random forest we introduce the VIX index. Herein, we have introduced the VIX index to the model as a ratio of today's VIX value divided by yesterdays VIX value. Herein, we thus either find a relative increase or decrease in the VIX ratio which is more informative to the model rather than the VIX values which from day-to-day were varying rather much. Moreover, by comparing the VIX charts with the price charts of the S&P500 we observed that change in the VIX value did not necessarily result in a drastic decrease or increase of the price of the S&P500. Therefore, to also decrease the effect of the VIX value gradually going up we decided to use the ratio instead of the plane VIX-value as an additional feature for the random forest model. This additional feature increased the accuracy of the model from 53.7% to 55%, the F1-score from 63% to 64.8% and the return from 572.30$ to 899.63$. Although the performance of the random forest has increased and the accuracy being higher than the accuracy of the solo LSTM the return of the random forest is still 50% lower than the return of the LSTM solo. Nevertheless, the increases that were found due to the addition of the VIX ratio are a good benchmark performance for the eventual combined model which will be discussed in the following section. In which we will add the LSTM predictions to the current feature list of the random forest.

4.3 LSTM & Random forest combined model

In order to observe whether the predictability of the random forest can be enhanced by including LSTM predictions as a feature it was necessary to prepare the LSTM model. This was due to the training and testing period required for the random forest. The LSTM prediction data set had to be of the same length as the data input of random forest to ensure that the predictions generated were aligned with the length of the data input required for the random forest. Increasing the prediction period at the expense of the training data did not solve this problem since it drastically impacted the predictability of the LSTM, causing the learning period to be too small and therefore, the model was

Metrics	LSTM	RF	RF+VIX	LSTM+RF	LSTM+RF+VIX	B&H	RB
Accuracy	0.527	0.537	0.550	0.505	0.507	-	0.506
Precision	0.530	0.559	0.566	0.541	0.542	-	0.554
F1-score	0.68	0.63	0.648	0.59	0.598	-	0.529
Recall	0.949	0.722	0.758	0.564	0.666	-	0.507
AUC	-	0.533	0.546	0.483	0.490	-	-
MSE	0.036	-	-	-	-	-	-
TP	727	606	607	450	449	-	410
TN	50	186	190	295	299	-	336
FP	632	478	474	382	381	-	330
FN	66	205	204	348	346	-	399
Predictions	1475	1475	1475	1475	1475	-	1475
Return	1116.27$	572.30$	899.63$	251.67$	468.27$	2067.09$	755.36$

Table 1: Performance evaluation of the different established models

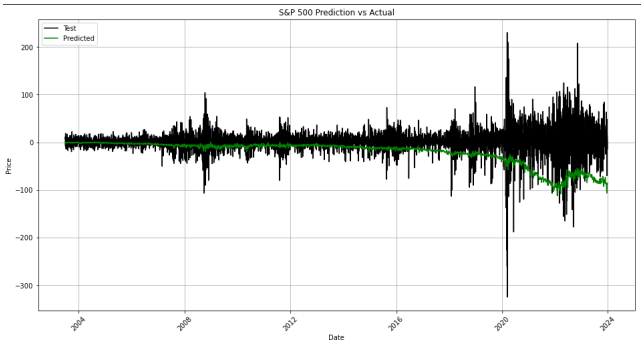

Figure 8: Solo LSTM predictions on the price difference in which 50% of the input data was used as the learning data and 50% as the validation data

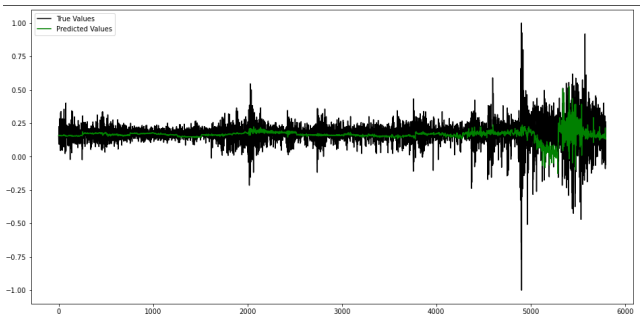

Figure 9: Daily LSTM predictions of the price difference and S&P500 actual price difference per day using a rolling window

unable to get familiar with the data which resulted in losing the trend overtime which can be observed in figure 8. Therefore, hyperparameter tuning did not result in a better performance. This problem is solved with the introduction of a rolling horizon to the model. Herein, constantly 6 years of data of which 80% is training data. Each time the model has learned and tested the window is reset and shifted to the next period. By doing so the LSTM can be updated with shifting data. Since the price of the S&P500 is not stationary and over the years has shown a continuous upwards trend the model also have to be updated accordingly. This problem is likely to be caused by the LSTM taking into account too much "old" data which causes it to lose sight of the trend overtime which can be observed in figure 8. In figure 9 the predictions of the LSTM with a rolling window is displayed. Herein, it can be seen that the model able to maintain the trend better than the model used in figure 8. Using the rolling horizon we have thus also decreased the trend loss issue.

With the LSTM model established, its predictions were incorporated as features into the random forest model. For the random forest, the same model architecture was used as in the solo random forest approach. Despite efforts to improve the model through hyperparameter tuning testing different numbers of estimators, maximum depths, and minimum sample splits no significant improvements in accuracy or precision were observed. Consequently, the parameters from the solo random forest model (100 estimators, no maximum layer depth, and a minimum sample split of 100) were retained.

In table 1 the results of the combined model is displayed in column LSTM+RF. Herein, it can be observed that the model was able to obtain an accuracy of 50.05% meaning that only half of the predictions of the next day's closing price up or downward directions were made correctly. Furthermore, the model was examined on the precision score herein it was found that the model is 54.1% of the cases it made a buy prediction, the prediction

was indeed a upwards move in the market. In trading it is important to have a high precision on your to avoid wrong market interpretations. Comparing the accuracy and precision of the model, it can be observed that the model is more accurate result is found when we only observe the buy predictions, which is in line with our strategy since we do not short the market. On the other side we observe the recall of this model in which we obtained 56.4%. In order to observe the balance between the recall and the precision of this model, the F1-score can be observed. For this model an F1-score of 59% was found indicating a fairly balance between the precision and the recall, meaning that the model was reasonably able to identify uptrends while managing false uptrends and downtrends. Herein, the ability of the model to manage false uptrends is important since the model only enters the market when it predicts an uptrend. False downtrends will only cause a loss in potential profit which is less worse then making a loss due to a falsely buy signal. The performance of the model has resulted in, if all uptrends calls were bought on the call day and sold before the market closes during the period of 01-01-2018 till 01-01-2024, a return of 251.67$. Efforts to enhance the performance of the model by adding the VIX index as a feature to this model showed a slightly increase in performance. In particular the model was more able to identify actual up trends than the model that did not include the VIX index. The results of this model are displayed in table 1 in column LSTM+RF+VIX. Also, the return of the model including the VIX index as a feature increased by approximately 85%. indicating that including the VIX index indeed enhanced the performance of the model.

4.4 Model comparison

In order to observe the overall effectiveness of using the machine learning models used to predict the next days price difference of the S&P500, we will compare the performance of solo LSTM model with the solo random forest (RF), the combined RF models, Buy&Hold (B&H) and random buy (RB). First of all, it must be noted that the only actual predictions that are of interest to us are the "true positive" and "false positive" predictions. This is because in case of a negative prediction, a 0 signal, the model will not buy nor sell and will continue to the next day. Therefore, we will, in the worst case scenario, miss out on positive return but we are protected for negative return. When we get a positive prediction, a 1 signal, we are not protected against negative return since the model will then enter the market and place a position. Therefore, the most interesting metric is actually the precision score since this metric analysis the ration between the "true positive" and "false positive" predictions. However, this metric can easily be misinterpreted. This is because we thus also still have the "false negative" and "true negative" predictions if the number of those predictions are high, we are still able to obtain a high precision score but we still get a poor return since the model will not enter the market in this situation. In order to obtain the highest return we therefore must find the largest number of "True positives" but at the same time obtain the lowest number of "False positives". We can observe this phenomena by looking at table 1 in this table we can observe the "True positive" (TP), "True negative" (TN), "False positive" (FP) and the "False negative" (FN) predictions of the models. Herein, we can see that the Random forest with the VIX index as a feature (RF+VIX) model obtained the highest precision score of 56.6%, however obtains a lower return then the LSTM which has a precision 53%. The higher return can be explained by the fact that the LSTM was able to predict 120 more TP predictions than the RF+VIX model, which means that we obtain returns for 120 extra days. However, the LSTM also 158 more FP predictions than the RF+VIX which could be interpreted as the benefit of the large number of TP being canceled out, since also the precision score is lower than the RF+VIX. However, when analysing the chart of the S&P500 over the period of 1994-01-01 till 2024-01-01 which is displayed in figure 10 it can be seen that over the period the S&P500 experienced a rather continues upwards trend in which the price has experienced a climb of approximately 10x in its price, increasing from 471$ on 1994-01-01 to 4743$ on 2024-01-02. This means that the upwards price trend is stronger than the downwards price trend. Projecting this finding back to the TP and FP predictions of ours models that would mean that the FP predictions should have less impact on the return than the TP predictions in general otherwise we would not obtain the ordinary upwards price trend urge of the S&P500. Therefore, up to a certain point it can be considered that it is more important to increase the number of TP predictions rather than optimizing the precision score. This also due to the fact that in order to make the model an interesting investment method it should be able

Figure 10: Price movement of the S&P500 over the training and testing period (Yahoo, 2024)

to obtain a higher return than at least the RB but more importantly the B&H. Herein, in table 1 by first observing the RF models, it can be observed that the return of the RF is enhanced when the VIX index is included to the features list of the models, whereas when the LSTM is added the return and performance are drastically decreased to less than half of the returns found in the models that did not include the LSTM. This can be the result of the LSTM input already introducing an implied error to the model due to faulty predictions. Furthermore, these faulty predictions have an increased impact on the model due to the input of the LSTM being a classified input meaning the input is either a 1 or a 0 value. Therefore, the RF can easily be too biased towards the faulty prediction of the LSTM. Therefore, it can be concluded that introducing LSTM predictions of price difference as an additional input feature for the RF model to be an ineffective measure to improve the predictability of the RF model to predict the next day's closing price difference of S&P500. Furthermore, comparing the returns of the RF models with the returns of the B&H it can be observed that the models were not able to outperform the return of the B&H in which the B&H realized an approximately 130% higher return than the best performing RF model (RF+VIX) during the same period. Nevertheless, the return over a 6-year period is still low compared to the Buy&Hold (B&H) return which is displayed in table 1 in column B&H. The buy and hold was able to obtain a return of 2067.09$ in the same period. Furthermore, observing the returns of a random buy (RB), which can be observed in table 1 column RB, which turned out to be 755.36$ during the same period. These results not only indicate that our models were not able to outperform the B&H which should be outperformed to consider the models as a meaningful trading strategy, but also that only the solo LSTM and RF+VIX models were able to out perform the RB. Meaning that including the LSTM predictions in the features of the RF feature list did not result in the forseen increased predictability as expected. Furthermore, when we increase the testing period from 01-01-2024 to 01-06-2024 we observe that all models further lose their predictability. In Table 2 we have displayed the LSTM solo and the RF+VIX which were the 2 best performing models for the testing period till 01-01-2024.

Metrics	LSTM	RF+VIX	B&H	RB
Accuracy	0.463	0.513	-	0.473
Precision	0.500	0.525	-	0.497
F1-score	0.36	0.61	-	0.485
Recall	0.280	0.727	-	
AUC	-	0.512	-	-
MSE	0.025	-	-	-
TP	227	606	-	410
TN	464	186	-	336
FP	225	478	-	330
FN	576	205	-	399
Predictions	1492	1492	-	1492
Return	593.43$	-98.89$	2133.72$	-552.57$

Table 2: Performance evaluation of the shifted testing period for the solo LSTM and RF+VIX models

5 Conclusion & recommendations

The primary purpose of this study was to determine whether machine learning models specifically, Random Forest (RF) and Long Short-Term Memory (LSTM) combined with technical indicators such as the VIX index, moving averages, and trading volume, can accurately predict price trends in the S&P500 index. The main research question was whether the combination of the VIX index, moving average, and trading volume, as the financial indicators, used in a combined random forest and LSTM model could have a better performance, compared to a BUY&HOLD strategy, individual Random Forest and LSTM models and a random buy heuristic. This was investigated by buying and or doing nothing the day after in terms of predictive accuracy and returns. The results indicate that the LSTM model alone, without the VIX index, achieved a prediction accuracy of 52.68%, with an F1-score of 68%. It correctly predicted the up or down movement of the S&P500 price slightly more than half of the time, outperforming a random buy strategy. However, despite its predictive accuracy, the LSTM model did not exceed the performance of a simple buy&hold strategy. The buy&hold yielded 130% higher returns over the same period than the LSTM model. The Random Forest model, when incorporating the VIX index improved accuracy to 55%, with an F1-score of 64.8%. Herein, the VIX index was used to capture market sentiment and volatility trends, providing insights into potential market reversals. Despite this, the Random Forest model still fell short of outperforming the buy-and-hold strategy in terms of overall returns. Moreover, the attempt to combine LSTM predictions as a feature for the Random Forest model did not result in an improved performance. Instead, it reduced the models predictive accuracy to 50.5%, indicating that the integration of LSTM predictions into the Random Forest did not enhance the model's ability to predict stock price trends effectively. One significant finding was that using a "newer" testing period with the same model inputs negatively affected the models' performance, reducing their predictability when applied to different market conditions. This highlights the importance of model robustness across various time frames, particularly in high-volatility phases of the market. In answer to the main research question, while the machine learning models showed moderate success in predicting price trends, they did not outperform the traditional buy-and-hold strategy in terms of total returns. Thus although the models being reasonably accurate, they struggled to provide better results than straightforward investment strategies such as a buy&hold. Based on the findings, we recommend further exploration of how the VIX index, along with other technical indicators like moving averages and trading volume, can be better integrated into machine learning models such as LSTM. While the VIX index improved the accuracy of the model, additional research is needed to explore how to leverage it more effectively for trend prediction. We also suggest examining the impact of testing period shifts more closely in terms of what is the optimum testing time frame before the hyperparamters have to be adjusted, as these can significantly influence model robustness and predictive accuracy due to influence of old data. The

model could retain too much of this data causing it to be unable to make accurate predictions. Therefore, it is worthwhile to study the what the optimum input data length is before the models it's input and hyperparameters must be re-examined. Additionally, future studies should focus on optimizing these machine learning models to outperform not only random buy strategies but also simple investment strategies like buy-and-hold. Enhancing model resilience across different market conditions and time periods will be crucial for their practical application in stock market prediction. Furthermore, future research can be conducted towards the ethical and legal aspect of the use of machine learning in stock price prediction to be used in stock trading.

References

Ahmed, S., Alam, M. S., Hassan, M., Rodela, M., Ishtiak, T., Rafa, N., Rahman, M. M., Ali, A. B. M. S., & Gandomi, A. (2023). Deep learning modelling techniques: Current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, *56*. https://doi.org/10.1007/s10462-023-10466-8

Ahoniemi, K. (2008). Modeling and forecasting the vix index. *Available at SSRN 1033812*.

Bhatt, S. (2019, April). Reinforcement learning 101. https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

Bi, Q., Goodman, K. E., Kaminsky, J., & Lessler, J. (2019). What is machine learning? a primer for the epidemiologist. *American Journal of Epidemiology*. https://doi.org/10.1093/aje/kwz189

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327. https://doi.org/10.1016/0304-4076(86)90063-1

Brailsford, T. J. (1996). The empirical relationship between trading volume, returns and volatility. *Accounting & Finance*, *36*(1), 89–111.

Bustos, O., Pomares, A., & Gonzalez, E. (2017). A comparison between svm and multilayer perceptron in predicting an emerging financial market: Colombian stock market, 1–6.

Calzone, O. (2018). *An intuitive explanation of lstm* [Medium]. https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c

Devi, B. U., Sundar, D., & Alli, P. (2013). An effective time series analysis for stock trend prediction using arima model for nifty midcap-50. *International Journal of Data Mining & Knowledge Management Process*, *3*(1), 65.

Donges, N. (2023, September). Random forest: A complete guide for machine learning. https://builtin.com/data-science/random-forest-algorithm

edX. (2023, December). Arima modeling. https://www.mastersindatascience.org/learning/statistics-data-science/what-is-arima-modeling/

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, *50*(4), 987–1007. Retrieved February 4, 2024, from http://www.jstor.org/stable/1912773

Enke, D., Grauer, M., & Mehdiyev, N. (2011). Stock market prediction with multiple regression, fuzzy type-2 clustering and neural networks. *Procedia Computer Science*, *6*, 201–206. https://doi.org/10.1016/j.procs.2011.08.038

Fama, E. F. (1991). Efficient capital markets: Ii. *The Journal of Finance*, *46*(5), 1575–1617. https://doi.org/10.1111/j.1540-6261.1991.tb04636.x

Fama, E. F., & French, K. R. (1992). The cross-section of expected stock returns. *The Journal of Finance*, *47*(2), 427–465.

Fehst, V., La, H., Nghiem, T.-D., Mayer, B., Englert, P., & Fiebig, K.-H. (2018). Automatic vs. manual feature engineering for anomaly detection of drinking-water quality, 5–6. https://doi.org/10.1145/3205651.3208204

Fumo, J. (2017, August). Types of machine learning algorithms you should know. https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

Gallagher, L. A., & Taylor, M. P. (2002). Permanent and temporary components of stock prices: Evidence from assessing macroeconomic shocks. *Southern Economic Journal*, *69*(2), 345. https://doi.org/10.2307/1061676

Giot, P. (2005). Relationships between implied volatility indices and stock index returns. *Journal of Portfolio Management*, *31*(3), 92–100.

Giro, D. (2024). Wat is de vix index? https://www.degiro.nl/leren-beleggen/beleggersacademie/beginner-cursus/vix-index

Google. (2024, February). Classification: Roc curve and auc — machine learning — google for developers. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Grigoryan, H. (2017/05). Stock market trend prediction using support vector machines and variable selection methods, 210–213. https://doi.org/10.2991/ammsa-17.2017.45

Hari, Y., & Dewi, L. P. (2018). *Forecasting system approach for stock trading with relative strength index and moving average indicator* [Doctoral dissertation, Petra Christian University].

Hayes, A. (2023, September). Autoregressive integrated moving average (arima) prediction model. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp
IBM. (2024a). *Random forest - ibm* [IBM Website]. https://www.ibm.com/topics/random-forest
IBM. (2024b). What is Supervised Learning? — IBM — ibm.com [[Accessed 29-01-2024]].
Jiao, Y., & Jakubowicz, J. (2017). Predicting stock movement direction with machine learning: An extensive study on sp 500 stocks, 4705–4713. https://doi.org/10.1109/BigData.2017.8258518
Khanderwal, S., & Mohanty, D. (2021). Stock price prediction using arima model. *International Journal of Marketing & Human Resource Research*, *2*(2), 98–107.
Kuepper, J. (2023, December). Cboe volatility index (vix): What does it measure in investing? https://www.investopedia.com/terms/v/vix.asp
Kumar, R. (2015, November). Efficient capital markets and its implications. https://www.sciencedirect.com/science/article/abs/pii/B9780128023037000036?via%3Dihub
Madge, S., & Bhatt, S. (2015). Predicting stock price direction using support vector machines. *Independent work report spring*, *45*.
Maverick, J. (2022). What are the main advantages and disadvantages of using a simple moving average (sma)? https://www.investopedia.com/ask/answers/013015/what-are-main-advantages-and-disadvantages-using-simple-moving-average-sma.asp
Mehta, S., Rana, P., Singh, S., Sharma, A., & Agarwal, P. (2019). Ensemble learning approach for enhanced stock prediction. *2019 Twelfth International Conference on Contemporary Computing (IC3)*, 1–5. https://doi.org/10.1109/IC3.2019.8844891
Mitchell, C. (2023a, September). How to use a moving average to buy stocks. https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp#:~:text=The%20moving%20average%20(MA)%20is,time%20period%20the%20trader%20chooses.
Mitchell, C. (2023b, September). How to use a moving average to buy stocks. https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp#:~:text=The%20moving%20average%20(MA)%20is,time%20period%20the%20trader%20chooses.
Oracle. (2024). Wat is machine learning? https://www.oracle.com/nl/artificial-intelligence/machine-learning/what-is-machine-learning/
Oshiro, T., Perez, P., & Baranauskas, J. (2012). How many trees in a random forest? *Lecture notes in computer science*, *7376*. https://doi.org/10.1007/978-3-642-31537-4_13
Pavan Kumar Illa, A. K. S., Balakesavareddy Parvathala. (2022). Stock price prediction methodology using random forest algorithm and support vector machine. *Materials Today: Proceedings*, *56*(4), 1776–1782. https://doi.org/https://doi.org/10.1016/j.matpr.2021.10.460
Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. In *Machine learning* (pp. 101–121). Elsevier.
Plachý, R. (2014). Impact of trading volume on prediction of stock market development. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, *62*, 1373–1380. https://doi.org/10.11118/actaun201462061373
Reddy, V. K. S. (2018). Stock market prediction using machine learning. *International Research Journal of Engineering and Technology (IRJET)*, *5*(10), 1033–1035.
Sadorsky, P. (2021). A random forests approach to predicting clean energy stock prices. *Journal of Risk and Financial Management*, *14*(2), 48.
Scikit. (2024, January). Rbf svm parameters. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html
Siami-Namini, S., & Namin, A. S. (2018). A comparison of arima and lstm in forecasting time series. *arXiv preprint arXiv:1803.06386*. https://arxiv.org/abs/1803.06386
Sonkavde, G. (2023). Forecasting stock market prices using machine learning. *Cryptography*, *11*(3), 94. https://www.mdpi.com/2227-7072/11/3/94
Srivatsavaya, P. (2023). Lstm — implementation, advantages and diadvantages. https://medium.com/@prudhviraju.srivatsavaya/lstm-implementation-advantages-and-diadvantages-914a96fa0acb
Thompson, C. (2023, December). Fundamental vs. technical analysis: What's the difference? https://www.investopedia.com/ask/answers/difference-between-fundamental-and-technical-analysis/

Ţiţan, A. G. (2015). The efficient market hypothesis: Review of specialized literature and empirical research. *Procedia Economics and Finance*, *32*, 442–449. https://doi.org/10.1016/s2212-5671(15)01416-1

Tsai, C.-F., Hsu, Y.-C., & Wu, S.-C. (2011). Using machine learning to assess information in stock prices. *Expert Systems with Applications*, *38*(10), 12759–12766.

Twin, A. (2022, January). Volume of trade: How it works, what it means, and examples. https://www.investopedia.com/terms/v/volumeoftrade.asp

Vo, N., & Ślepaczuk, R. (2022). Applying hybrid arima-sgarch in algorithmic investment strategies on s&p500 index. *Entropy*, *24*(2), 158.

Yahoo. (2024). Yahoo finance - stock market live, quotes, business finance news. https://finance.yahoo.com/

Appendices

A Random forest code

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 15 21:56:09 2024

@author: Gebruiker
"""

import yfinance as yf
import pandas as pd
import numpy as np
import pandas_ta as ta
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, accuracy_score, f1_score, mean_squared_error,
↪  roc_curve, auc, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import random

# Set seeds for reproducibility
np.random.seed(42)
random.seed(42)

# Download S&P 500 data
sp500 = yf.download('^GSPC', start='1994-01-01', end='2024-06-01')
vix = yf.download('^VIX', start='1994-01-01', end='2024-06-01')

# Create target columns
sp500["Tomorrow"] = sp500["Close"].shift(-1)
sp500["Target_diff"] = sp500["Tomorrow"] - sp500["Close"]
sp500["Target_class"] = (sp500["Target_diff"] > 0).astype(int)

# Data preprocessing
sp500["yesterday_diff"] = sp500["Close"] - sp500["Close"].shift(1)
sp500["yesterday_vol_diff"] = sp500["Volume"] - sp500["Volume"].shift(1)
sp500["MA20"] = ta.ema(sp500.Close, length=20)
sp500["MA60"] = ta.ema(sp500.Close, length=60)
sp500["MA90"] = ta.ema(sp500.Close, length=90)
sp500["MA120"] = ta.ema(sp500.Close, length=120)
sp500["MA150"] = ta.ema(sp500.Close, length=150)

# Merge with VIX data
vix.rename(columns={'Close': 'VIX'}, inplace=True)
sp500 = sp500.merge(vix[['VIX']], left_index=True, right_index=True, how='left')
sp500['Ratio_vix'] = sp500["VIX"] / sp500["Close"]

# Add rolling means for ratios
sp500['MMA20'] = sp500["Close"] / sp500["Close"].rolling(20).mean()
sp500['MMA60'] = sp500["Close"] / sp500["Close"].rolling(60).mean()
sp500['MMA90'] = sp500["Close"] / sp500["Close"].rolling(90).mean()
sp500['MMA120'] = sp500["Close"] / sp500["Close"].rolling(120).mean()
sp500['MMA150'] = sp500["Close"] / sp500["Close"].rolling(150).mean()

# Drop rows with NaN values
sp500.dropna(inplace=True)
```

```
55
56 # Add trends
57 sp500['10day_trend'] = sp500['Target_diff'].shift(1).rolling(10).sum() / 10
58 sp500['30day_trend'] = sp500['Target_diff'].shift(1).rolling(30).sum() / 30
59 sp500['50day_trend'] = sp500['Target_diff'].shift(1).rolling(50).sum() / 50
60
61 # Filter data from 1990 onwards
62 sp500 = sp500.loc["1990-01-01":].copy()
63
64 # Define predictors and target
65 predictors = ["Close", "Volume", 'MA20', 'MA60', 'MA90', 'MA120', 'MA150', '10day_trend',
   ↪  '30day_trend', '50day_trend','Ratio_vix']
66 X = sp500[predictors]
67 y = sp500["Target_class"]  # Use the binary classification target
68
69 # Split data into train and test sets (80-20 split)
70 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
71
72 # Handle NaN values using SimpleImputer
73 imputer = SimpleImputer(strategy='mean')
74 X_train_imputed = imputer.fit_transform(X_train)
75 X_test_imputed = imputer.transform(X_test)
76
77 # Define and train the model
78 model = RandomForestClassifier(n_estimators=100, min_samples_split=100, random_state=1)
79 model.fit(X_train_imputed, y_train)
80
81 # Make predictions
82 preds = model.predict(X_test_imputed)
83
84 # Predict probabilities for ROC calculation
85 y_probs = model.predict_proba(X_test_imputed)[:, 1]
86
87 # Calculate ROC curve
88 fpr, tpr, thresholds = roc_curve(y_test, y_probs)
89
90 # Calculate AUC
91 roc_auc = auc(fpr, tpr)
92
93 # Plot ROC curve
94 plt.figure()
95 plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
96 plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
97 plt.xlim([0.0, 1.0])
98 plt.ylim([0.0, 1.05])
99 plt.xlabel('False Positive Rate')
100 plt.ylabel('True Positive Rate')
101 plt.title('Receiver Operating Characteristic')
102 plt.legend(loc="lower right")
103 plt.show()
104
105 # Make random predictions (0 or 1)
106 random_preds = np.random.choice([0, 1], size=len(preds))
107
108 # Calculate performance metrics for the test set based on predicted values
109 predicted_target_diff = []
110 total_predicted_diff = 0  # To accumulate total predicted differences
111
112 random_predicted_target_diff = []
```

```
total_random_predicted_diff = 0  # To accumulate total predicted differences for random predictions

test_indices = X_test.index  # Get the indices from X_test which align with y_test

for i in range(len(preds)):
    index = test_indices[i]
    if preds[i] == 1:
        predicted_diff = sp500.loc[index, "Target_diff"]
        predicted_target_diff.append(predicted_diff)
        total_predicted_diff += predicted_diff
    else:
        predicted_target_diff.append(0)

    # Calculate for random predictions
    if random_preds[i] == 1:
        random_predicted_diff = sp500.loc[index, "Target_diff"]
        random_predicted_target_diff.append(random_predicted_diff)
        total_random_predicted_diff += random_predicted_diff
    else:
        random_predicted_target_diff.append(0)

# Calculate performance metrics for the test set
precision = precision_score(y_test, preds)
accuracy = accuracy_score(y_test, preds)
f1 = f1_score(y_test, preds)
# For MSE, we need to calculate it manually since it's a regression metric, not classification
mse = mean_squared_error(y_test, predicted_target_diff)

# Calculate performance metrics for random predictions
random_precision = precision_score(y_test, random_preds)
random_accuracy = accuracy_score(y_test, random_preds)
random_f1 = f1_score(y_test, random_preds)
random_mse = mean_squared_error(y_test, random_predicted_target_diff)

# Calculate confusion matrix to get TP, TN, FP, and FN
cm = confusion_matrix(y_test, preds)
tn, fp, fn, tp = cm.ravel()

# Print performance metrics for the test set
print("\nTest Set Performance Metrics:")
print(f"Precision: {precision:.4f}")
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"MSE: {mse:.4f}")
print(f"Total Predicted Difference: {total_predicted_diff:.4f}")

# Print TP, TN, FP, and FN
print("\nConfusion Matrix:")
print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")

# Print ROC and AUC
print(f"ROC AUC: {roc_auc:.4f}")

print("\nRandom Predictions Performance Metrics:")
print(f"Precision: {random_precision:.4f}")
print(f"Accuracy: {random_accuracy:.4f}")
print(f"F1 Score: {random_f1:.4f}")
print(f"MSE: {random_mse:.4f}")
```

```
print(f"Total Random Predicted Difference: {total_random_predicted_diff:.4f}")

# Print the periods of predictions
start_date = X_test.index.min()
end_date = X_test.index.max()
print(f"\nPredictions made between {start_date} and {end_date}")
```

B LSTM code

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun  9 13:23:02 2024

@author: Gebruiker
"""

import numpy as np
import yfinance as yf
import pandas as pd
import pandas_ta as ta
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, precision_score, recall_score, f1_score
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input, Activation
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
import random

# Download S&P 500 data
sp500 = yf.download('^GSPC', start='1994-02-01', end='2024-06-01')
vix = yf.download('^VIX', start='1994-02-01', end='2024-06-01')

# Set seeds for reproducibility
np.random.seed(42)
tf.random.set_seed(42)
random.seed(42)

# Data preprocessing
sp500.drop(columns=["High", "Low", "Adj Close"], inplace=True)
vix.rename(columns={'Close': 'VIX'}, inplace=True)
sp500 = sp500.merge(vix[['VIX']], left_index=True, right_index=True, how='left')
sp500['Ratio_vix'] = sp500["VIX"] / sp500["Close"]
sp500["yesterday_diff"] = sp500["Close"] - sp500["Close"].shift(1)
sp500["yesterday_vol_diff"] = sp500["Volume"] - sp500["Volume"].shift(1)
sp500["MA20"] = ta.ema(sp500.Close, length=20)
sp500["MA60"] = ta.ema(sp500.Close, length=60)
sp500["MA90"] = ta.ema(sp500.Close, length=90)
sp500["MA120"] = ta.ema(sp500.Close, length=120)
sp500["MA150"] = ta.ema(sp500.Close, length=150)
sp500["Tomorrow"] = sp500["Close"].shift(-1)
sp500["Target_diff"] = sp500["Tomorrow"] - sp500["Close"]
sp500["Target"] = (sp500["Tomorrow"] > sp500["Close"]).astype(int)

sp500.dropna(inplace=True)
sp500.reset_index(inplace=True)

# Define backcandles before using it
```

```
backcandles = 30

# Feature scaling
data_set = sp500.iloc[:, 1:15]
sc = MinMaxScaler(feature_range=(-1, 1))
data_scaled = sc.fit_transform(data_set)

# Prepare the dataset for LSTM
X = []
for j in range(11):  # -1 to exclude the target column
    X.append([])
    for i in range(backcandles, data_scaled.shape[0]):
        X[j].append(data_scaled[i - backcandles:i, j])

X = np.moveaxis(X, [0], [2])
X, yi = np.array(X), np.array(data_scaled[backcandles:, -1])
y = np.reshape(yi, (len(yi), 1))

# Split data into train and test sets
splitlimit = int(len(X) * 0.8)
X_train, X_test = X[:splitlimit], X[splitlimit:]
y_train, y_test = y[:splitlimit], y[splitlimit:]

# Correct the input shape for the LSTM layer
n_features = X_train.shape[2]

# LSTM model
lstm_input = Input(shape=(backcandles, n_features), name='lstm_input')
inputs = LSTM(180, name='lstm_layer')(lstm_input)
inputs = Dense(1, name='dense_layer')(inputs)
output = Activation('linear', name='output')(inputs)
model = Model(inputs=lstm_input, outputs=output)

# Compile and train the model
adam = Adam()
model.compile(optimizer=adam, loss='mse')
model.fit(x=X_train, y=y_train, batch_size=15, epochs=30, shuffle=True,
↪  validation_split=0.07)

# Predictions
y_pred = model.predict(X_test)

# Print the start date of the test period
# Extract dates for the test set (match length with y_test_unscaled)
dates = sp500['Date'][backcandles:].reset_index(drop=True)
dates_test = dates[splitlimit:].reset_index(drop=True)  # Reset the index to align with the
↪  test data

start_date_test_period = dates_test.iloc[0]
print(f'Start Date of Test Period: {start_date_test_period}')

# Print predictions and true values
for i in range(11):
    print(y_pred[i], y_test[i])

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

```

```
data_test_scaled = data_scaled[splitlimit + backcandles:, :-1]  # Match the length of y_test and y_pred
y_test_unscaled = sc.inverse_transform(np.concatenate((data_test_scaled, y_test), axis=1))[:, -1]
y_pred_unscaled = sc.inverse_transform(np.concatenate((data_test_scaled, y_pred), axis=1))[:, -1]

# Print unscaled predictions and true values
for i in range(10):
    print(f'Predicted: {y_pred_unscaled[i]}, Actual: {y_test_unscaled[i]}')

y_test_target = np.where(y_test_unscaled > 0, 1, 0)
y_pred_target = np.where(y_pred_unscaled > 0, 1, 0)

# Calculate total predicted difference
total_predicted_diff = 0  # To accumulate total predicted differences

for i in range(len(y_pred_target)):
    if y_pred_target[i] == 1:
        index = splitlimit + backcandles + i
        predicted_diff = sp500.loc[index, "Target_diff"]
        total_predicted_diff += predicted_diff

print(f'Total Predicted Difference: {total_predicted_diff}')

# Summation of price differences from the prediction period
total_price_diff = sp500.loc[splitlimit + backcandles:, "Target_diff"].sum()
print(f'Summation of Price Difference from Prediction Period: {total_price_diff}')

# Calculate TP, TN, FP, and FN
TP = np.sum((y_pred_target == 1) & (y_test_target == 1))
TN = np.sum((y_pred_target == 0) & (y_test_target == 0))
FP = np.sum((y_pred_target == 1) & (y_test_target == 0))
FN = np.sum((y_pred_target == 0) & (y_test_target == 1))

print(f'Total Predictions: {len(y_pred_target)}')
print(f'True Positives: {TP}')
print(f'True Negatives: {TN}')
print(f'False Positives: {FP}')
print(f'False Negatives: {FN}')

# Plot results with dates on the x-axis
plt.figure(figsize=(16, 8))
plt.plot(dates_test, y_test_unscaled, color='black', label='Test')
plt.plot(dates_test, y_pred_unscaled, color='green', label='Predicted')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('S&P 500 Prediction vs Actual')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Calculate accuracy
accuracy = (y_pred_target == y_test_target).mean() * 100
print(f'Accuracy: {accuracy:.2f}%')

# Calculate precision, recall, and F1-score
precision = precision_score(y_test_target, y_pred_target)
recall = recall_score(y_test_target, y_pred_target)
```

```
f1 = f1_score(y_test_target, y_pred_target)

print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1-score: {f1:.2f}')

results_df = pd.DataFrame({
    'True Target': y_test_target.flatten(),
    'Predicted Target': y_pred_target.flatten()
})
```

C LSTM code for RF feature input

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 16 18:49:18 2024

@author: Gebruiker
"""

# -*- coding: utf-8 -*-
"""
Created on Sun Jun  9 13:23:02 2024

@author: Gebruiker
"""

import numpy as np
import yfinance as yf
import pandas as pd
import pandas_ta as ta
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dense, Input, Activation
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
import random

# Download S&P 500 data
sp500 = yf.download('^GSPC', start='1994-01-01', end='2024-01-01')

# Set seeds for reproducibility
np.random.seed(42)
tf.random.set_seed(42)
random.seed(42)

# Data preprocessing
sp500.drop(columns=["High", "Low", "Adj Close"], inplace=True)
sp500["yesterday_diff"] = sp500["Close"] - sp500["Close"].shift(1)
sp500["yesterday_vol_diff"] = sp500["Volume"] - sp500["Volume"].shift(1)
sp500["MA20"] = ta.ema(sp500.Close, length=20)
sp500["MA60"] = ta.ema(sp500.Close, length=60)
sp500["MA90"] = ta.ema(sp500.Close, length=90)
sp500["MA120"] = ta.ema(sp500.Close, length=120)
sp500["MA150"] = ta.ema(sp500.Close, length=150)
sp500["Tomorrow"] = sp500["Close"].shift(-1)
```

```
sp500["Target"] = (sp500["Tomorrow"] > sp500["Close"]).astype(int)
sp500["Target_diff"] = sp500["Tomorrow"] - sp500["Close"]

sp500.dropna(inplace=True)
sp500.reset_index(inplace=True)

# Feature scaling
data_set = sp500.iloc[:, 1:14]
sc = MinMaxScaler(feature_range=(-1, 1))
data_scaled = sc.fit_transform(data_set)

# Prepare the dataset for LSTM
X = []
backcandles = 30
for j in range(10):  # -1 to exclude the target column
    X.append([])
    for i in range(backcandles, data_scaled.shape[0]):
        X[j].append(data_scaled[i - backcandles:i, j])

X = np.moveaxis(X, [0], [2])
X, yi = np.array(X), np.array(data_scaled[backcandles:, -1])
y = np.reshape(yi, (len(yi), 1))

# Rolling horizon prediction setup
rolling_window_size = 6 * 252  # 6 years of data, assuming 252 trading days per year
prediction_horizon = 252  # Predict for the next year
total_years = 20
total_predictions = total_years * prediction_horizon

# Initialize lists to store results
all_predictions = []
all_true_values = []

# Loop over the rolling windows
for start in range(0, len(X) - rolling_window_size - prediction_horizon + 1,
↪  prediction_horizon):
    end = start + rolling_window_size
    next_end = end + prediction_horizon

    # Split data into train and test sets
    X_train, y_train = X[start:end], y[start:end]
    X_test, y_test = X[end:next_end], y[end:next_end]

    # LSTM model
    lstm_input = Input(shape=(backcandles, X_train.shape[2]), name='lstm_input')
    inputs = LSTM(300, name='lstm_layer')(lstm_input)
    inputs = Dense(1, name='dense_layer')(inputs)
    output = Activation('linear', name='output')(inputs)
    model = Model(inputs=lstm_input, outputs=output)

    # Compile and train the model
    adam = Adam()
    model.compile(optimizer=adam, loss='mse')
    model.fit(x=X_train, y=y_train, batch_size=40, epochs=40, shuffle=True,
    ↪  validation_split=0.07)

    # Predictions
    y_pred = model.predict(X_test)

    # Store predictions and true values
```

```
    all_predictions.extend(y_pred.flatten())
    all_true_values.extend(y_test.flatten())

# Convert lists to arrays for final evaluation
all_predictions = np.array(all_predictions)
all_true_values = np.array(all_true_values)

# Plot results
plt.figure(figsize=(16, 8))
plt.plot(all_true_values, color='black', label='True Values')
plt.plot(all_predictions, color='green', label='Predicted Values')
plt.legend()
plt.show()

# Evaluate the model
mse = mean_squared_error(all_true_values, all_predictions)
print(f'Mean Squared Error: {mse}')

# Unscale the predictions and true values for comparison
data_test_scaled = data_scaled[len(data_scaled) - len(all_predictions):, :-1]  # Match the length
all_true_values_unscaled = sc.inverse_transform(np.concatenate((data_test_scaled, all_true_values.reshape(-1, 1)), axis=1))[:, -1]
all_predictions_unscaled = sc.inverse_transform(np.concatenate((data_test_scaled, all_predictions.reshape(-1, 1)), axis=1))[:, -1]

# Print unscaled predictions and true values
for i in range(10):
    print(f'Predicted: {all_predictions_unscaled[i]}, Actual: {all_true_values_unscaled[i]}')

y_test_target = np.where(all_true_values_unscaled > 0, 1, 0)
y_pred_target = np.where(all_predictions_unscaled > 0, 1, 0)

# Calculate accuracy
accuracy = (y_pred_target == y_test_target).mean() * 100
print(f'Accuracy: {accuracy:.2f}%')

results_df = pd.DataFrame({
    'True Target': y_test_target.flatten(),
    'Predicted Target': y_pred_target.flatten()
})

total_price_diff = sp500.loc[splitlimit + backcandles:, "Target_diff"].sum()
print(f'Summation of Price Difference from Prediction Period: {total_price_diff}')

# Calculate TP, TN, FP, and FN
TP = np.sum((y_pred_target == 1) & (y_test_target == 1))
TN = np.sum((y_pred_target == 0) & (y_test_target == 0))
FP = np.sum((y_pred_target == 1) & (y_test_target == 0))
FN = np.sum((y_pred_target == 0) & (y_test_target == 1))
```

D Random forest model in which the LSTM is used as a feature code

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 30 14:52:07 2024

```

```
@author: Gebruiker
"""

import yfinance as yf
import pandas as pd
import numpy as np
import pandas_ta as ta
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, accuracy_score, f1_score, mean_squared_error,
↪  confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import random
import os

# Set seeds for reproducibility
np.random.seed(42)
random.seed(42)

# Download S&P 500 data
sp500 = yf.download('^GSPC', start='1994-01-01', end='2024-01-01')
vix = yf.download('^VIX', start='1994-01-01', end='2024-01-01')

# Create target columns
sp500["Tomorrow"] = sp500["Close"].shift(-1)
sp500["Target_diff"] = sp500["Tomorrow"] - sp500["Close"]
sp500["Target_class"] = (sp500["Target_diff"] > 0).astype(int)

# Data preprocessing
sp500["yesterday_diff"] = sp500["Close"] - sp500["Close"].shift(1)
sp500["yesterday_vol_diff"] = sp500["Volume"] - sp500["Volume"].shift(1)
sp500["MA20"] = ta.ema(sp500.Close, length=20)
sp500["MA60"] = ta.ema(sp500.Close, length=60)
sp500["MA90"] = ta.ema(sp500.Close, length=90)
sp500["MA120"] = ta.ema(sp500.Close, length=120)
sp500["MA150"] = ta.ema(sp500.Close, length=150)

# Merge with VIX data
vix.rename(columns={'Close': 'VIX'}, inplace=True)
sp500 = sp500.merge(vix[['VIX']], left_index=True, right_index=True, how='left')
sp500['Ratio_vix'] = sp500["VIX"] / sp500["Close"]

# Add rolling means for ratios
sp500['MMA20'] = sp500["Close"] / sp500["Close"].rolling(20).mean()
sp500['MMA60'] = sp500["Close"] / sp500["Close"].rolling(60).mean()
sp500['MMA90'] = sp500["Close"] / sp500["Close"].rolling(90).mean()
sp500['MMA120'] = sp500["Close"] / sp500["Close"].rolling(120).mean()
sp500['MMA150'] = sp500["Close"] / sp500["Close"].rolling(150).mean()

# Drop rows with NaN values
sp500.dropna(inplace=True)

# Add trends
sp500['10day_trend'] = sp500['Target_diff'].shift(1).rolling(10).sum() / 10
sp500['30day_trend'] = sp500['Target_diff'].shift(1).rolling(30).sum() / 30
sp500['50day_trend'] = sp500['Target_diff'].shift(1).rolling(50).sum() / 50

# Filter data from 1990 onwards
sp500 = sp500.loc["1990-01-01":].copy()

```

```
# Load additional data from Excel file
file_path = r'C:\Users\Gebruiker\Documents\thesis\y_pred_target.xlsx'

# Check if the file exists
if os.path.exists(file_path):
    # Load the Excel file
    excel_data = pd.read_excel(file_path)

    # Display the first few rows of the Excel data for debugging
    print("Excel data preview:")
    print(excel_data.head())

    # Ensure the length of excel_data matches the length of sp500 DataFrame
    if len(excel_data) == len(sp500):
        # Assign the index of sp500 to excel_data
        excel_data.index = sp500.index

        # Rename the column to avoid conflicts
        excel_data.columns = ["y_pred_target"]

        # Merge the Excel data with sp500 DataFrame
        sp500 = sp500.merge(excel_data, left_index=True, right_index=True, how='left')
    else:
        print(f"Length mismatch: excel_data has {len(excel_data)} rows, but sp500 has
        ↪  {len(sp500)} rows.")
        # Adjust the length by trimming or padding the excel_data
        if len(excel_data) > len(sp500):
            excel_data = excel_data.iloc[:len(sp500)]
        else:
            additional_rows = pd.DataFrame({"y_pred_target": [np.nan] * (len(sp500) -
            ↪  len(excel_data))}, index=sp500.index[len(excel_data):])
            excel_data = pd.concat([excel_data, additional_rows])

        # Assign the index of sp500 to excel_data
        excel_data.index = sp500.index

        # Merge the Excel data with sp500 DataFrame
        sp500 = sp500.merge(excel_data, left_index=True, right_index=True, how='left')
else:
    print(f"File not found: {file_path}")

# Define predictors and target
predictors = ["Close", "Volume", "VIX", 'MA20', 'MA60', 'MA90', 'MA120', 'MA150',
↪  'Ratio_vix', '10day_trend', '30day_trend', '50day_trend', 'y_pred_target']
X = sp500[predictors]
y = sp500["Target_class"]  # Use the binary classification target

# Split data into train and test sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Handle NaN values using SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Define and train the model
model = RandomForestClassifier(n_estimators=100, min_samples_split=100, random_state=1)
model.fit(X_train_imputed, y_train)

# Make predictions
```

```
preds = model.predict(X_test_imputed)

# Calculate performance metrics for the test set based on predicted values
predicted_target_diff = []
total_predicted_diff = 0  # To accumulate total predicted differences

test_indices = X_test.index  # Get the indices from X_test which align with y_test

for i in range(len(preds)):
    index = test_indices[i]
    if preds[i] == 1:
        predicted_diff = sp500.loc[index, "Target_diff"]
        predicted_target_diff.append(predicted_diff)
        total_predicted_diff += predicted_diff
    else:
        predicted_target_diff.append(0)

# Calculate performance metrics for the test set
precision = precision_score(y_test, preds)
accuracy = accuracy_score(y_test, preds)
f1 = f1_score(y_test, preds)
# For MSE, we need to calculate it manually since it's a regression metric, not
↪  classification
mse = mean_squared_error(y_test, predicted_target_diff)

# Calculate confusion matrix for detailed prediction statistics
conf_matrix = confusion_matrix(y_test, preds)
tn, fp, fn, tp = conf_matrix.ravel()

# Print performance metrics for the test set
print("\nTest Set Performance Metrics:")
print(f"Precision: {precision:.4f}")
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"MSE: {mse:.4f}")
print(f"Total Predicted Difference: {total_predicted_diff:.4f}")

# Print confusion matrix results
print("\nConfusion Matrix Results:")
print(f"True Positives: {tp}")
print(f"True Negatives: {tn}")
print(f"False Positives: {fp}")
print(f"False Negatives: {fn}")

# Print the periods of predictions
start_date = X_test.index.min()
end_date = X_test.index.max()
print(f"\nPredictions made between {start_date} and {end_date}")
```

E LSTM closing price prediction code

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug  7 22:32:20 2024

@author: Gebruiker
"""

```

```python
# -*- coding: utf-8 -*-
"""
Created on Mon May 20 13:27:45 2024

@author: Gebruiker
"""

import tensorflow as tf
print("TensorFlow version:", tf.__version__)

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import yfinance as yf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Load data from Yahoo Finance
df = yf.download('^GSPC', start='1990-01-01', end='2024-01-01')

# Preprocess data
data = df['Close'].values.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)

train_data_len = len(df[df.index < '2010-01-01'])
train_data = scaled_data[:train_data_len, :]
test_data = scaled_data[train_data_len-60:, :]

x_train, y_train = [], []
for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

# Build LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=25))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)

# Prepare test data
x_test, y_test = [], []
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])
    y_test.append(test_data[i, 0])
x_test, y_test = np.array(x_test), np.array(y_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

# Make predictions
predictions = model.predict(x_test)
```

```
predictions = scaler.inverse_transform(predictions)
y_test = scaler.inverse_transform(y_test.reshape(-1, 1))

# Plot the results
train = df[:train_data_len]
valid = df[train_data_len:]
valid['Predictions'] = predictions

plt.figure(figsize=(16, 8))
plt.title('LSTM closing price prediction')
plt.xlabel('Date')
plt.ylabel('Close Price USD (\$)')
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```