Comparison of Different Methods for Forecasting Stock Market Movements

by

Vachan Sardar

A Master's Thesis

Submitted in partial fulfilment of the requirements for the award of
Master of Science of Loughborough University

September 2024

Acknowledgements

I want to extend my deepest gratitude to Dr. Parisa Derakhshan for her unwavering direction throughout this research. Her experience has helped guide this paper's direction, with her valuable insights providing a significant uplift to its quality.

I also want to thank the computer science faculty and staff at Loughborough University for their resources in completing this project. Though these were behind-the-scenes contributions, they have been instrumental in moving my work forward.

Finally, I am grateful to all my family and friends for their support. Without them, it would have been impossible to deliver this thesis.

Abstract

In the stock market, predicting future price movements on past data is highly valuable to make informed decisions for investment and risk management. In this study, we carried out analysis for predicting stock prices using historical data from Meta Platforms Inc. using the different forecasting models: Auto-Regressive Integrated Moving Average (ARIMA), Seasonal Auto-Regressive Integrated Moving Average (SARIMA), Auto-Regressive Integrated Moving Average with exogenous input (ARIMAX), Seasonal Auto-regressive Integrated Moving Average with exogenous variables (SARIMAX), Linear Regression, Support Vector Regression (SVR), Random Forest, Neural Networks such as convolutional neural network (CNN), Recurrent neural networks (RNN), Long short-term memory (LSTM). Of the models tested, ARIMAX using external economic indicators was found to be the most accurate and delivered the lowest Mean Squared Error (MSE). The linear and ridge regression models were also rated suitable for linear situations. Neural networks were successfully able to capture non-linear patterns, although they needed tuning in order not to overfit on this problem. Additionally, we found that historically designed models were ineffective in dynamic markets.

This paper focuses on the importance of incorporating external variables and model selection for changing market circumstances. In addition to these direct contributions, our study also provides directions for future research. Hybrid models could be developed in combination with sentiment analysis and fundamental analysis, refining real-time forecasting methods. The results will be useful for both researchers and financial practitioners to improve the efficacy of stock price forecasts.

Contents

List of Figures

List of Tables

List of Abbreviations

The following list describes several symbols, abbreviations, and acronyms that will be later used within the body of this thesis.

ARIMA AutoRegressive Integrated Moving Average

ARIMAX AutoRegressive Integrated Moving Average with Exogenous Regressors

CNN Convolutional Neural Network

KNN k-Nearest Neighbors

LSTM Long Short-Term Memory

MACD Moving Average Convergence Divergence

MAE Mean Absolute Error

MSE Mean Squared Error

RMSE Root mean squared error

RNN Recurrent Neural Network

RSI Relative Strength Index

SARIMA Seasonal AutoRegressive Integrated Moving Average

SARIMAX Seasonal AutoRegressive Integrated Moving Average with Exogenous Regressors

SMA Simple Moving Average

Chapter 1

Introduction

1.1 Background

Stock market forecasting represents one of the most crucial aspects to be researched for decades, receiving an enormous amount of attention from both investors and business analysts who need a way to make their investment decisions. Reliable stock price forecasts are critical to both investors and financial institutions, helping them increase their chances of success while also taking advantage of market movements. On the other hand, predicting stock markets is highly complicated as it spans across several dimensions such as economic indicators, the sentiment of investors, and company-specific news which makes accurate prediction tasks difficult to solve [1], [2]. Although presentational advancements have enabled more accurate predictive models, applying these in a strategy setting is difficult; often complex methodologies are needed to realise the desired outcome [3].

1.2 Objectives

The purpose of this study is to examine the use of a range of forecasting models for predicting stock prices by taking into consideration an empirical study and thereby accepting suitable methods. The following models spanning different types (statistical, machine learning, and deep learning) were evaluated: ARIMA, SARIMA, Linear Regression, Random Forest, Support Vector Regression (SVR), KNN, Gradient boosting, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), LSTM-Long Short

Term Memory (LSTM). The models were trained and validated using historical data from Meta Platforms Inc., with 80% of the data being used for training. At the same time, the existing 20% was dynamically reserved to serve as a test set. The purpose of this research is to identify the models with the best results as seen from a pricing accuracy perspective and applicability in the financial domain.

1.3 Contributions

This work contributes in multiple ways to stock price forecasting. The literature review is done to identify the utility of price forecasting for stock prices and various models in predicting time series data. The literature review identified the relevant characteristics of time series from stock market analysis and other time series-based evaluations that can be used in the design of this project. that A detailed description of identified models, including their mathematical description, is compiled. The existing literature was used to design the study and optimise the models in statistical, machine learning, and deep learning domains. Additionally, the comparative analysis of the performance of various models is provided to inform model selection in different scenarios and their limitations.

1.4 Paper Structure

The structure of the paper is as follows: Chapter 2 provides a detailed literature review of the forecasting models used in this study. Details of data collection, preprocessing, and implementation for each model are presented in Chapter 4. The results of the comparative analysis and further elaboration based on our interpretation are presented in chapter 5. Chapter 6 critically reviews the main results on which this paper focused and offers practical implications for investors and financial analysts together with further theoretical research.

Chapter 2

Literature Review

This section highlights previous work done using classical statistical models like ARIMA and SARIMA, as well as higher-end Machine Learning techniques, including Random Forest, SVR, KNN, Gradient boosting, and deep learning models such as CNN, RNN, and LSTM, their importance, and how they have improved forecasting accuracy.

2.1 Relevance of Time Series Forecasting in Stock Prices Predictions

In the stock market, we can make informative decisions for investment and risk management by predicting future price movements based on relevant data. The time information allows investors to trade that timing and ensure the risk reduction (diversification/hedging/etc.), among best practices[4], [5]. It also makes algorithmic trading possible by automatically executing trades based on reliable forecasts, resulting in better performance and higher returns. Portfolio managers can consider these forecasts to balance potential gains and risks, while long-term investors can align their savings objectives with reliable forecasts. Furthermore, precise predictions can help improve market effectiveness and information asymmetry reduction and integration with global markets [6], [7].

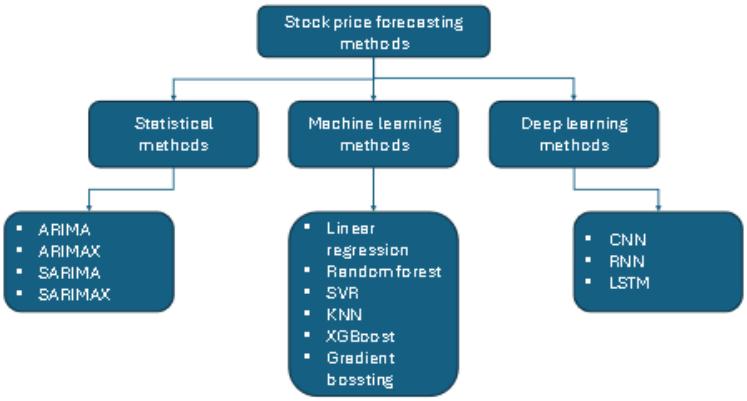

Figure 2.1: A flowchart covering the various methods outlined in the study.

2.2 Relevant Statistical Models for Time Series Analysis of Stock Market Data

There are many models for time series analysis, and extensive functionality exists for handling the time series in Python. The time series data can have strong trends and seasonalities. The models were chosen because of their capabilities in handling complexity surrounding typical time series data, for example, trends or seasonality, and how external variables can affect them. SARIMA is a more powerful extension of ARIMA that accounts for seasonality. ARIMAX considers the external variables in forecasting to make it more accurate. SARIMAX combines both, which are used when you have a seasonal pattern and influence factors. These models provide a powerful and versatile framework for accurate prediction on many types of time series data [8], [9].

2.2.1 Autoregressive Integrated Moving Average (ARIMA)

Autoregressive Integrated Moving Average (ARIMA) models are popular for time series forecasting in economics, finance as well as climatology. Models ARIMA was introduced by Box and Jenkins in the 1970s when the model used a combination of autoregression, differencing, and moving average com-

ponents to predict time series data. One reason for their popularity is the capability to deal with continual non-stationary signals by making them stationary through differencing. It has been documented that ARIMA models work best when the time series data is well understood, and stationary [10]. ARIMA has frequently outperformed other statistical models in complex datasets like stock markets [4] . Hybrid approaches that combine ARIMA with other methods, such as Artificial Neural Networks (ANN), have also been developed to increase the accuracy of a prediction, especially for data showing nonlinear patterns [11] . A number of these hybrid models outperform traditional ARIMA models, illustrating how more complex real-world data requires the adaptation of algorithms for analyses [11].

2.2.2 Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX)

ARIMAX models expand ARIMA by adding external variables, which can improve accuracy when other factors are included. There are several examples in the literature that have successfully used ARIMAX. Hyndman showed that the predictions can be improved by including factors such as economic indicators or weather conditions, which are affecting website visits [12] . ARIMAX, for instance, improved electricity consumption prediction by adding weather information [13], and on public health, it incorporated vaccination rate as an extension variable feature to ARIMA time series model [14]. Under the finance branch, ARIMAX even outperforms GARCH and the seasonal repeated model in exchange rates [13][10] and electricity price predictions by including macroeconomic variables. These are useful properties of specifications in econometric forecasting, as including external variables often increases prediction accuracy [15], [16].

2.2.3 Seasonal AutoRegressive Integrated Moving Average (SARIMA)

The SARIMA model is based on ARIMA but has seasonal components defined by Box and Jenkins as time series analysis [9]. In this sense, although SARIMA is very appropriate for the treatment of certain seasonal data (particularly with few missing values), Makridakis and Hibon [17] identify that in general terms it can become complex so much as there are simpler methods

than outperform it in overall accuracy among various time series. Moreover, combining SARIMA with neural networks Zhang further improved forecasting as it combined the ability to capture linear and non-linear patterns [18]. Cryer and Chan reported SARIMA for seasonal data and provided a guideline to model diagnostics [19] . Hyndman and Athanasopoulos evaluated the appropriateness of SARIMA, its advantages and disadvantages, and some state-of-the-art extensions, such as SARIMAX, which deals with external regressors [12] .

2.2.4 Seasonal AutoRegressive Integrated Moving Average with eXogenous variables (SARIMAX)

These models extend SARIMA by also taking into account external variables, which is important, especially these days: if the time series of interest (such as stock prices) is affected not only by their own dynamics but greatly so because, e.g., economic indicators change dramatically then one can choose to go for a model capable of dealing with this situation [12]. Contreras et al. found that augmenting inputs with weather conditions as exogenous input data increased the robustness of their framework to predict electricity prices [20]. Box and Tiao developed the use of SARIMAX for intervention analysis, providing a means to accurately model how one-off events such as policy changes affect an economic indicator [21][23]. Durbin and Koopman referred to the usefulness of SARIMAX as a state space representation, which allowed for an elastic way of modeling various time series data [22][25].

2.3 Relevant Machine Learning Models for Time Series Analysis of Stock Market Data

There are multiple machine learning models available for time series analysis, and we restricted the possibilities to Linear Regression, Random Forest, SVR (Support Vector Regression), XGBoost and Gradient Boosting which were judged best suitable because of their capabilities in terms of complex patterns. Linear regression: a simple model to describe linear relationships with data [23]. The Random Forest is very well-suited for handling non-linearities and interactions [24] , while the SVR has a high capability of

adapting to learn in large-dimensional spaces as well as being able to handle outliers [25] . XGBoost and Gradient Boosting were chosen because these algorithms performed well predictively and could learn complex relationships in an iterative manner [26], [27] .

2.3.1 Linear Regression

Linear regression is probably the most often learned and used model in equity price predictions because it is simple and intuitive. On the other hand, although Atsalakis and Valavanis [28] mention that the GRNN does fairly represent linear relationships between stock prices with financial indicators; they also say it is problematic to figure out non-linearities of functioning markets. Hybrid models and enhancements have been researched to overcome these limitations. Huang et al. believed that although linear regression serves as a good initial approach, ARIMA and neural networks are empirically more accurate than other methods with the same model structure [29]. Ballings et al. demonstrated performance boosts from using linear regression in tandem with time series analysis. They created historical rank variables that capture the last 5 or so transactions at a customer-item level. Still, these generally will not outperform machine learning approaches (e.g., gradient boosting). have shown that the use of external factors, including sentiment analysis can improve on predictive power [30] . However, baseline linear regression may not be the ideal approach in stock forecasting given that as a simple tool it is still limited by all of these assumptions and sensitivity to outliers [31].

2.3.2 Random Forest

Works in the embedded financial series were mostly concentrated on two different aspects: developing random forests and SVM for modeling volatility, cointegration [30] . But these models require expert feature selection and tuning, as elaborated by Bose et al. [32]. Patel et al. determined that they do not manage to show improvement over human forecasters when markets alter quickly, suggesting that a mixed model could improve results [31]. Though Tsai and Hsiao had provided advantages in crashing-price markets, they were limited during high volatility [33], needing to complement with other methods.

2.3.3 Support Vector Regression

SVR is one of stock price prediction methods and it performs better especially at noisy data in a high frequency. Guo et al. made an adaptive version of SVR that adjusts with time to improve results [34]. Kazem et al. chaos-based firefly algorithm (CFA) to optimise its parameter and capture non-linear patterns to enhance SVR [35]. The choice of kernel functions and parameters is critical for a proper accuracy [36].

2.3.4 K-Nearest Neighbors

KNN is a good choice for stock price prediction because it has the advantage of being very simple and at the same time able to model non-linear relationships without requiring strong assumptions about data. Patel et al. proposed optimised KNN works well in short term stock market predictions with the same accuracy as more complex models [31] . Kumar and Thenmozhi summarized it has an ability to efficiently capture local patterns which was validated by them [37].

2.3.5 XGBoost

It has been demonstrated that XGBoost is very useful for stock price prediction, outperforming traditional models such as ARIMA and simpler machine learning methods by overcoming complex data structures, non-linear relationships, and large datasets with missing values [26]. By fusing XGBoost into other classifiers improves the prediction accuracy, specifically in portraying both short- and long-term market behaviors [31]. Although in unstable market conditions [31] the model becomes difficult to interpret due to its complexity it is still important for feature selection of Desktop XGBoost, optimising this one, mitigating overfitting and raising enough generalisation.

2.3.6 Gradient Boosting

Gradient Boosting is one of the best techniques in stock price prediction, as it can be used to efficiently model nonlinear and non-gaussian relationships. The Fischer and Krauss 2018 study proved its superior predictive power compared to classical models in the context of complex markets, underlining that proper parameter selection and feature engineering are critical

[3]. It is worth quoting the results of Huerta, Corbacho, and Elkan (2013) that Gradient Boosting can provide a good prediction quality for "short-term" predictions, outperform models like SVM, especially when dealing with many features [38]. Krauss, Do, and Huck (2017) provided empirical evidence of its superiority compared to other ensembling methods, such as Random Forests in predicting the turbulent stock market [39].

2.4 Relevant Deep Learning Models for Time Series Analysis of Stock Market Data

Time series analysis deep learning models: In time sequence, there are many DL models, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks(RNNs), and Long Short-Term Memory(LSTM). As CNNs have a proven record of being good at detecting local patterns and features in time series data, especially by treating the input as a one-dimensional signal [40], we opted to use Convolutional Neural Networks for this task. At the time, RNNs were chosen as a type of deep learning paradigms specifically for modeling sequential data (to capture temporal dependencies), by utilising their hidden states [41]. Specifically, LSTMs were chosen because they are particularly good at capturing long-distance interactions and overcoming the vanishing gradient problem — making them suitable for time series data [42].

2.4.1 Convolutional Neural Network (CNN)

Traditionally used in image processing, CNNs have been successfully transformed for stock price prediction purposes and demonstrated their ability to capture local patterns as well as short-term trends from financial time series data such OHLCV (Open, High, Low, Close) [43]. CNNs are able to detect short-term fluctuations faster than LSTMs [44], [45], and this will help them combine well with external data such as sentiment analysis for more accurate predictions. Their flexibility, ability scalability as well and applicability in different areas, price prediction, and market trends evaluation, Machine Learning has been increasingly fast adopted into finance. Yet, capturing long-term dependencies is still a challenge whenever data sources are diverse, and it prompts the need for hybrid models where CNNs can be combined with other architectures [45]–[47] .

2.4.2 Recurrunt Neural Network (RNN)

RNNs, especially LSTMs, perform well in time series predictions, such as stock price prediction, due to their recurrent design and the ability to learn long-term dependencies. As per Fischer and Krauss (2018), LSTMs prove worthwhile over classical models, including logistic regression, for their capacity to manage complex relationships [3] . Guresen et al. confirmed the powerful capability of LSTMs to capture complex stock feature information and also showed that appropriate network architecture and tuning were necessary [48]. For the LSTM model, Bao et al., 2017 also reported the effectiveness of both types with a slight improvement by using LSTMs in certain contexts [49]. Chen et al. showed that LSTMs outperform ARIMA in stock return prediction [50].

2.4.3 Long Short-Term Memory (LSTM)

LSTM models are highly effective for stock price forecasting as they can capture temporal dependencies and general non-linear patterns that most other machine learning algorithms struggle with. Nelson et al. LSTMs have been shown to work extremely well with forecasting movements of markets, especially when they are highly volatile [51]. Shahi et al. While (2020), demonstrated that the use of news sentiment on financial reports improves accuracy for an LSTM [45] so there is also other interesting data into consideration. Akita et al. LSTMs are capable of performing these tasks because they are a more specialised RNN architecture (2016) and Fischer & Krauss [3], [52] both have found LSTMs to actually perform better at handling complex data and making longer-long-term predictions. LSTMs capturing complex stock price patterns was also validated by Bao, Yue and Rao (2017) [49] .

2.5 Additional Literature Reviews

2.5.1 In-depth information for the identified models

Appendix: the complete mathematical formulation of these models and evaluation matrices will be provided in detail for statistical, machine learning, and deep learning models. This includes a detailed treatment of the empiri-

cal models, from their mathematical derivation in Appendices used throughout this study, leading to predictive performance demonstrated within each case dealing.

2.5.2 Literature Supporting the Result Findings

existing literature corroborating the results Further findings from the literature review are described in detail under results, where we also draw on our own study. This section also discusses the limitations of the findings and reviews previous literature that endorsed our results, henceforth providing a holistic assessment to understand what this study has added.

Chapter 3

Design

A literature review in the time series analysis was performed to identify relevant statistical, machine learning, and deep learning models. Methodological considerations were also identified through these reviews, which were used in the implementation phase of the study. The methodological considerations included model selection (details provided in Appendix 1), data preprocessing, exploratory analysis, data cleaning and wrangling, feature engineering, feature selection, normalising, and scaling (3.1).

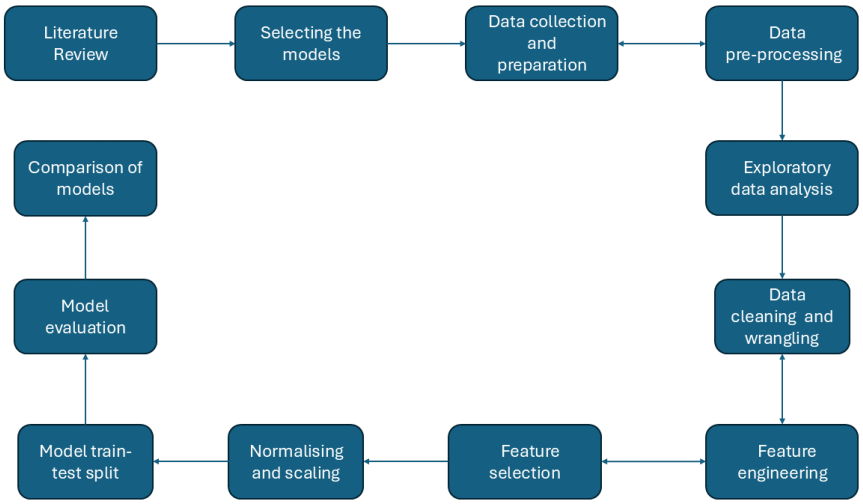

Figure 3.1: Workflow for study design. The figure show detailed description of considerations taken into account to design the overall project

Chapter 4

Methodology

In this section we describe the methodology that was used in our study, beginning from data collection to model preparation. We will demonstrate the process of data collection followed by a few methods used to preprocess our raw databases for accuracy and uniformity. Last, we go through the steps of cleaning up your data for modeling — such as feature selection and dataset transformation into ones that are appropriate with different time series forecasting models.

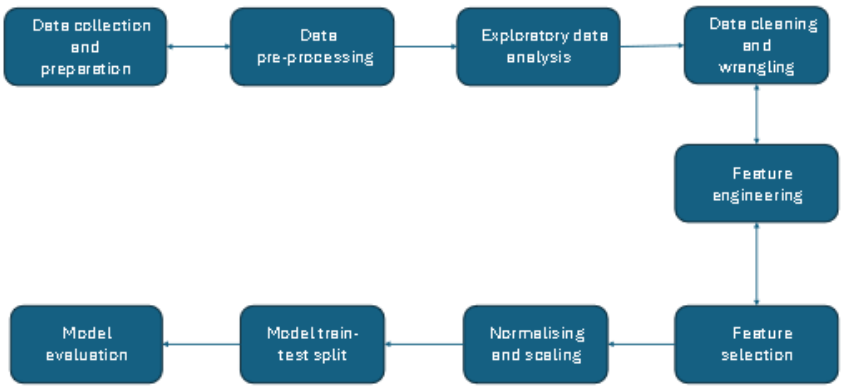

Figure 4.1: Methodology followed for evaluating identified models

4.1 Data Collection

We scraped historical stock data from Yahoo Finance for Meta Platforms, Inc. (Formerly Facebook) from 2014 to 2024 The 10-year stock data has recorded different phases of market from bull to bear and hence can be useful in analysing the past behaviour of stocks prices. The data includes daily metrics: Open, High, Low, Close, Adjusted Close, and Volume, for better understating market dynamics in developing predictive models of stock prices [53][64].

4.2 Preprocessing

A) Data Importation and Examination

The dataset, META. Meta Platforms, Inc. Commercial trading Statistics daily TradeLog csv imported. This is done by examining the standard columns (Date, Open, High, Low,Close, Adj_Close etc.) of df. head() and df. The info() function examines the data types and null values. The date column was converted to datetime for Time series analysis [54]

B) Exploratory Data Analysis (EDA)

- Visual Analysis: Closing prices were plotted over time to visualise trends.
- Statistical Summary: Summary statistics (df.describe()) highlighted data distribution and anomalies.
- Trend, Seasonality, and Residual Analysis: Decomposition methods identified underlying patterns in stock prices [55] .
- Correlation Analysis: A heatmap revealed strong correlations among price-related variables, while Volume showed a mild negative correlation with prices [56] .

C) Data Cleaning and Transformation

- Handling Missing Data: The dataset was checked for missing values, especially after date conversion.

- Outlier Detection: Outliers were identified and removed based on daily returns [57] .

D) Feature Engineering

- Lag Features: Previous close prices were included as lag features so that the model can remember how past performance in order to exploit temporal dependencies and accuracy of prediction.
- Rolling Statistics: Calculated rolling means and variances to smooth out the data and remove noise, which should guide our model in recognising underlying trends rather than interim oscillations.
- Moving Averages: Authorities calculated simple and exponential moving averages, both long-term trends and momentum, so the entire market move should have a clear view.
- Volatility Measures: Rolling standard deviations are used to measure price variability and market volatility as they serve a crucial role when trying to establish the risk behind expected changes in asset prices.
- Percentile Ranks and Seasonality: we use percentile ranks to measure the current position of prices relative to a historical distribution, while season decomposition is used to capture repeating patterns at particular times throughout the year.
- Technical Indicators: These indicators were calculated to understand the momentum, the strength of a trend, and potential reversal points (Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger Bands).
- Liquidity Metrics: Metrics of this type asses three types of indicators related to market liquidity (Simple Moving Average (SMA), Exponential moving average (EMA) trading volumes and bid ask spread) whose stability is a relevant aspect for price stabilisation and efficient trading [56].

4.3 Data Preparation for Modeling

- Train-Test Split: The data was split into training (80%) and testing (20%) sets, preserving chronological order.
- Normalisation and Scaling: Features were normalised using Min-Max scaling to ensure uniformity [54] .

In summary, we have an end-to-end procedural methodology that can be used for stock price prediction by initiating the process with the collection of clean data and then preprocessing it, removing all noise from the raw dataset and maintaining reliability throughout. Preparing the data for modeling helps enhance our models' predictive abilities, which will later be demonstrated by examples from a more detailed analysis underscored in subsequent sections and results.

Chapter 5

Results and Discussion

5.1 Results for Statistical Models

5.1.1 ARIMA and ARIMAX

For this study, we examined many time series forecasting models, such as different types of ARIMA and ARIMAX. The main objective was to check the performance of various model structures, with the aim of improving the accuracy of predictions by applying external variables and statistical methods such as rolling statistics. A summary of the experiments' results is given in Table 5.1.

Table 5.1: Summary of ARIMA and ARIMAX Experiment Results. The table shows the comparison of MSE, MAE, and RMS for various ARIMA and ARIMAX models evaluated

Model Configuration	**MSE**	**MAE**	**RMS**
ARIMA(1, 1, 1)	19140.7033	106.6797	138.3499
Rolling ARIMA(1, 1, 1)	2.1906	0.8684	1.4801
ARIMA(2, 1, 1) with Smoothing	19135.0549	106.6704	138.3295
Rolling ARIMA(2, 1, 1)	49.9574	1.3989	7.0681
ARIMAX(1, 1, 1) with Features	6.9896e-06	0.0022	0.0026
ARIMAX(2, 1, 1)	3.1679e-05	0.0053	0.0056
ARIMAX(2, 1, 2)	6.7135e-06	0.0022	0.0026
AIC-Selected ARIMA(1, 1, 0)	0.0002813	0.0102	0.0117

The ARIMAX(1, 1, 1) combined with feature engineering was the most precise and efficient model for time series forecasting. More specifically, this

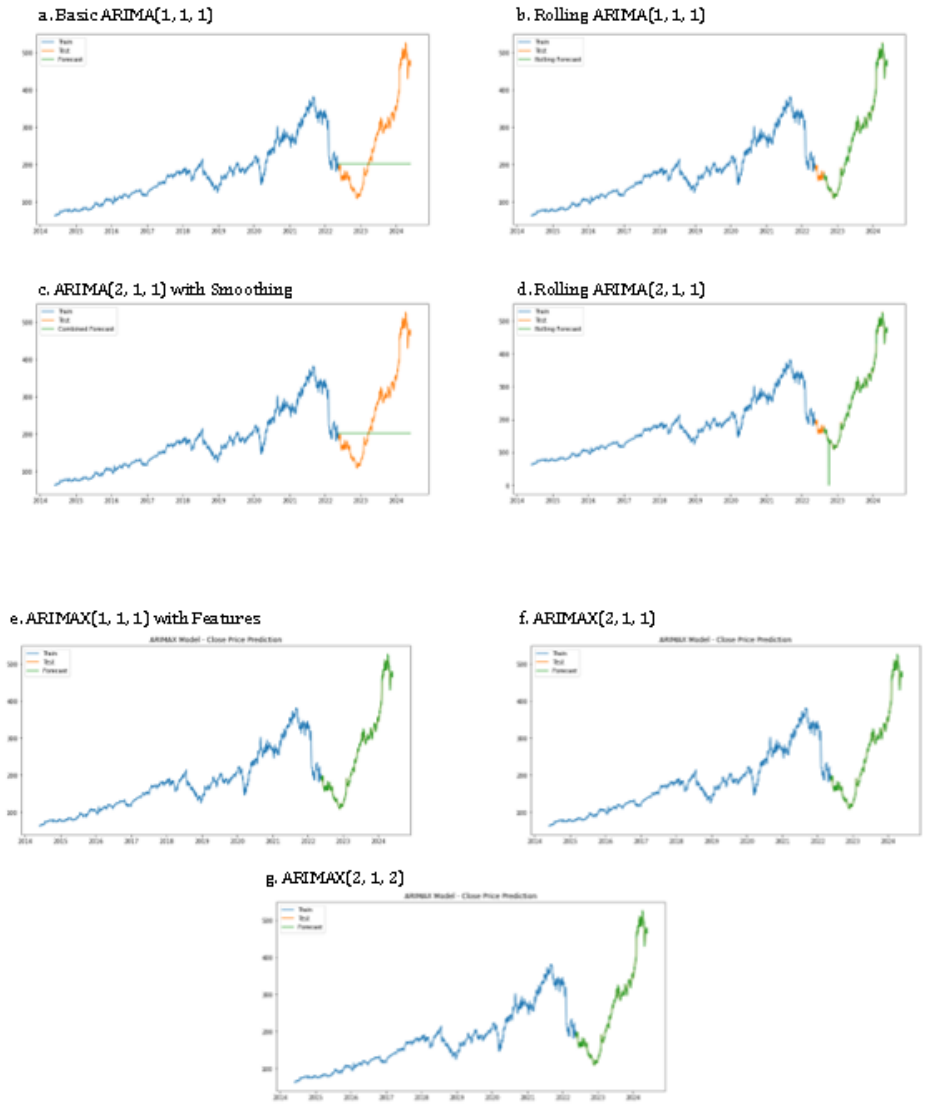

Figure 5.1: Visualisation of the results of ARIMA and ARIMAX. The figure shows the comparison of price forecasting capabilities of evaluated ARIMA and ARIMAX models

configuration scored the smallest Mean Squared Error, Mean Absolute Error, and Root Mean Squared Error. The most significant point that determined the difference in performance was exogenous variables, and their use enabled the model to reflect the impact of external factors that other ARIMA models could not reflect.

Key Factors for Superior Performance

1. Incorporation of Exogenous Variables: Several features could be added beyond simple time series for a deeper look at what may have been the underlying causes of our forecasted values, which ARIMAX models can naturally include. This is valuable, especially for complex systems where multiple external factors influence the outcomes.

2. Balance Between Simplicity and Complexity: The ARIMAX(1, 1, 1) model achieved the right balance between complexity with effectiveness. ARIMA(1, 1, 1) to ARIMAX (2, 1, 2) models were tested on this data, but they did not provide a significant improvement over the simple ARIMAX (1,1,1). Thus, increasing model complexity does not always result in better accuracy.

Key Comparisions

The ARIMAX(1, 1, 1) model with feature engineering significantly improved over the baseline ARIMA(1, 1, 1). The same result happened with the default ARIMA model based only on autoregressive, integrated, or moving-average components and produced much higher errors than these tuning models. The results highlight the need to include relevant external factors in predictive models. To a lesser extent, rolling statistics also enhanced performance, but very interestingly, it was the addition of exogenous variables which had by far the highest impact.

Connection to Prior Studies

Prior studies, it is suggested that the usage of ARIMAX models, including feature selection, could improve forecasting accuracy. Dissanayake and Hemachandra [58] reported ARIMAX as the best-performing model for multi-variate short-term traffic forecasting after feature selection outperforming VAR and LSTM models. Similarly, Chang et al. [59]. Recently

it was shown that integrating feature selection into ARIMAX models substantially increased the accuracy of forecasts compared to the traditional ARIMA. Such results demonstrate that ARIMAX with informative features could well handle intricate time-dependence structures. Our results are consistent with these studies, providing further evidence that incorporating exogenous variables into standard statistical models can lead to substantial gains in prediction accuracy.

5.1.2 SARIMA and SARIMAX

In this study, we have demonstrated the results of forecasting time series data with SARIMA and SARIMAX. We tested specifications enabled by various configurations of the model and enhancements like rolling windows or including exogenous variables. The goal was to find the best predictive model configuration while balancing it with accuracy vs. interpretability [, i.e., less complex models]. Overall, Table 5.2 summarises the results observed from these experiments.

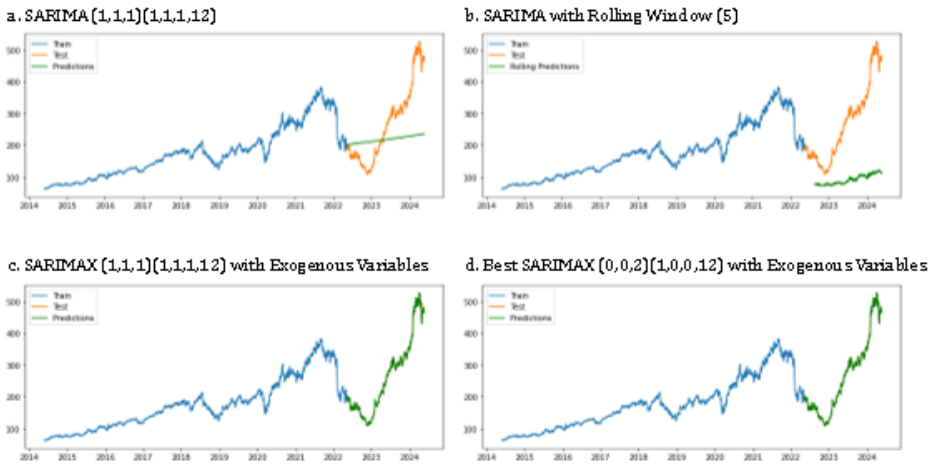

Figure 5.2: Visualisation of SARIMA and SARIMAX model results. The figure shows the stock price forecasting capabilities of evaluated models

Table 5.2: Performance Comparison for SARIMA and SARIMAX models. The table provides MSE, MAE and RMSE for evaluated models.

Model Configuration	**MSE**	**MAE**	**RMSE**
SARIMA (1,1,1)(1,1,1,12)	14933.98	95.18	122.20
SARIMA with Rolling Window (5)	44573.31	183.02	211.12
SARIMA with Rolling Window (60)	48998.29	193.52	221.36
SARIMAX (1,1,1)(1,1,1,12) with Exogenous Variables	0.0117	0.0790	0.1082
Best SARIMAX (0,0,2)(1,0,0,12) with Exogenous Variables	0.00029	0.0135	0.0170
SARIMAX with Rolling Window (60)	48676.26	192.84	220.63
Optimised SARIMAX with Rolling Window (60)	48676.37	192.84	220.63
Cross-Validation on Best SARIMAX Model	Mean MSE: 0.0014, Std MSE: 0.0013	Mean MAE: 0.0271, Std MAE: 0.0140	Mean RMSE: 0.0335, Std RMSE: 0.0169

Detailed Discussion of the Best-Performing Model

All these combinations gave results, but the best was the SARIMAX model with optimised parameters (0, 0, 2)(1, 0, 0,12) and exogenous variables. This model gave the lowest Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root mean squared error(RMSE). This performs better than all other configurations, including no external variable for SARIMA models.

Key Factors for Superior Performance

1. Incorporation of Exogenous Variables: By contrast, the SARIMAX model has several features in addition to just time series data, allowing it to comprehend external impacts better and hence was able to make a more accurate forecast. This was a key consideration because the exogenous variables provided valuable information that the SARIMA model could not capture.

2. Optimal Parameter Selection: Based on the grid search method and further model tuning, the most effective setup was the configuration (0, 0, 2)(1, 0, 0,12). The rationale behind this variant is the proper balance between tracking seasonal data patterns and controlling the complexity of the model, which eventually contributed to the increased efficiency of the predictive performance.

Key Comparisons

The introduction of exogenous variables dramatically improved error metrics compared to the baseline SARIMA(1,1,2)(1,1,1,12) model. Lastly, a pure SARIMA model resulted in much bigger errors. This result teaches that forecasting models should account for comprehensive macroeconomic information beyond overly simplistic heuristics. While gains in computational efficiency and forecast accuracy were observed with rolling windows, adding exogenous variables was most impactful at decreasing forecasting errors.

Connection to Prior Studies

The results of this research are, in general, consistent with other studies that have proved the effects on time series forecasting problems from considera-

tion of exogenous variables and modification applied to model parameters. For example, Hyndman and Athanasopoulos [12] argued that significant gains are almost always obtained through incorporating outside information (hybrid models). De Gooijer and Hyndman [60] found that adding exogenous variables to the SARIMAX greatly improved their accuracy in even faintly more complex/volatile environments. Our results lend further support to these findings, and they serve as evidence of the importance of integrating exogenous variables as well as selecting model parameters carefully to achieve high-accuracy forecasting.

5.2 Results for Machine Learning Models

5.2.1 Linear Regression

In this study, we analysed Linear Regression, Ridge regression, Polynomial Ridge as well Random forest along with ensemble models in this Paper. The main objective was to evaluate their performance with stock prices as time series forecasting including Mean Squared Error (MSE) and R-squared (R^2) on validation dataset & test datasets. Table 5.3 lists the results of these experiments

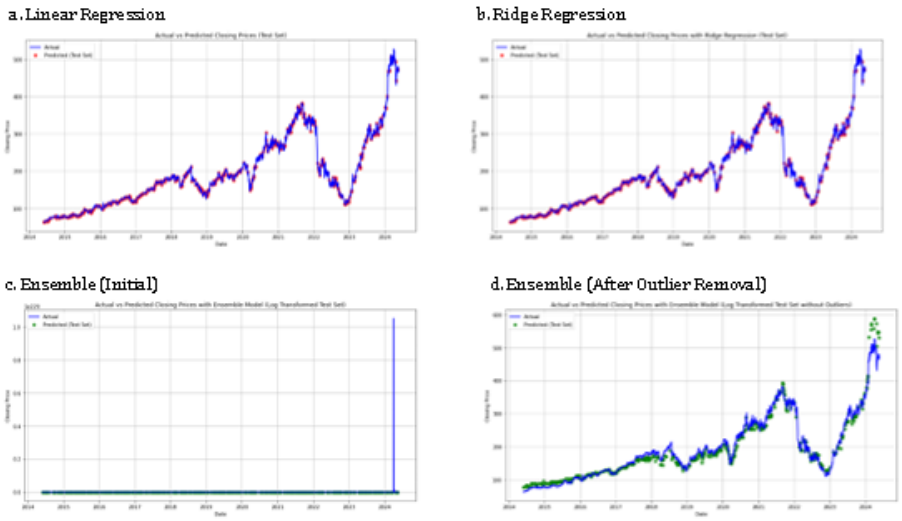

Figure 5.3: Visualisation of Linear Regression the results

Table 5.3: Summary of Linear Regression Experiment Results.

Model	**Validation MSE**	**Validation R^2**	**Test MSE**	**Test R^2**
Linear Regression	0.00533	0.999999	0.00392	0.999999
Ridge Regression	0.00530	0.999999	0.00391	0.999999
Polynomial Regression	2.45782	0.999723	0.28416	0.999968
Random Forest	0.28416	0.999968	0.28416	0.999968
Ensemble (Initial)	0.00705	0.968538	0.00608	0.972815
Ensemble (After Outlier Removal)	176.90	0.978129	231.59	0.973537

Detailed Discussion of the Best-Performing Models

Both Linear Regression and Ridge Regression models did well, with almost perfect R^2 values and extremely low MSEs in both validation as well in test datasets. Since the linear regression instance seemed to be doing very well, that means this data set could naturally have a lot of straight lines going through it, and so there was little need for regularisation. Given the chosen features, using such methods leads to a linear model that, in fact, turns out to be very good for capturing those simple relationships.

Key Factors for Superior Performance

1. Feature Selection and Engineering: Feature engineering, such as lagged prices, moving averages, and combinations with other technical indicators actually helped the linear models. Thus, the models could learn the fundamental patterns in stock prices.

2. Simplicity and Interpretability: Although simple, both Linear Regression and Ridge Regression were shown to produce strong predictions. These results were easy to interpret due to their directness, which is very valuable for virtually every financial forecasting application.

Key Comparisons

Compared to the baseline models, these hyperparameters helped achieve a significant increase in Test MSE and R^2 for the ensemble model after removing outliers. The other simpler models, like Linear and Ridge Regression, actually outperformed or at least matched the more complex Polynomial Regression or Random Forest in some metrics as well, showing that superficially created features are not always good.

Connection to Prior Studies

The results of this report conform to past studies, which also show the superiority of single-input forecasting models in financial time series. As noted in studies such as those by Campbell et al. As evidenced by Breiman [61] and Brooks [62], this combination of features means that constrained linear models can achieve high predictive accuracy, in some cases exceeding the performance of more advanced nonlinear techniques while still retaining interpretability. The improvements in performance of the ensemble model

resulting from outlier preprocessing are consistent with literature highlighting an integral role of data preprocessing on overall predictive quality.

5.2.2 Random Forest

This study was to model the Random Forest in different forms for stock price prediction. The experiments sought to test whether or not Random Forest would predict stock price movements well compared to the baseline, and for that purpose, explored a few improvements (e.g., significant feature engineering and extensive Recursive Feature Elimination (RFE)). These were all experiments, the results of which are tabulated and presented in Table 5.4.

Table 5.4: Summary of Random Forest Experiment Results.

Model	**Validation RMSE**	**Test RMSE**
Initial Random Forest	24.21	81.90
Improved Random Forest	9.48	59.96
Random Forest with Feature Selection	7.28	58.52

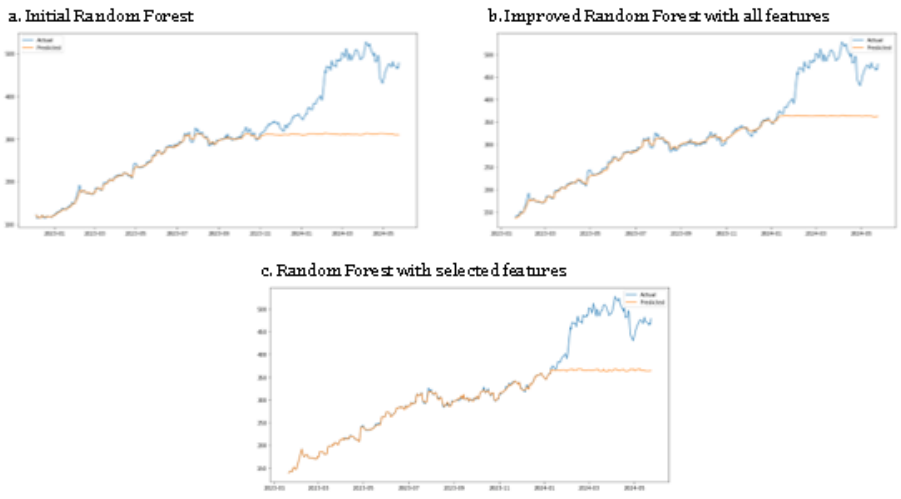

Figure 5.4: Visualisation of Random Forest results. The figure shows the stock price forecasting capabilities of evaluated models.

Detailed Discussion of the Best-Performing Model

What we can assume is probably the best random forest configuration of them all, in case you asked for a shallow tree-level: The one with extensive feature engineering and selected features using RFE. The first problem is that there are too many features in the model, it achieved a test RMSE of 58.52 over the validation RMSE (7.28) but while doing experimentation this was considered with unseen data otherwise I would have easily stuck more layers to reach such performance.

Key Factors for Superior Performance

1. Extensive Feature Engineering: Using a broad set of financial indicators and lagged features increased our model capacity to understand complex stock price dynamics. This helped in improving the predictability of our model as it had more data to use and, hence, better predictions.

2. Feature Selection with RFE: RFE enabled the model to select only important features and thus reduced noise, leading to better performance. This made the model avoid overfitting and had a better generalisation of unseen test data.

Key Comparisons

The previous Random Forest model had a validation RMSE of 24.21 and test RSME of 81.90, while the new model with feature engineering along with RFE once again outperformed in both train and validate datasets compared to its initial equivalent random forest model. The original model; with a wide array of unfiltered features, actually generates and worse RMSE which means not all the features has contributed positve to make our predictions better. Results show that much realised importance in (Random Forest) models comes from feature selection and engineering.

Connection to Prior Studies

These results support the insights discovered in prior analyses on how well-created feature selection and engineering can affect the performance of an ML model to predict stock prices. For example, Ballings et al. [30] showed that feature selection and model optimisation are two important factors

in enhancing the accuracy of financial forecasting. That is a hint to be taken from these two papers, and our results verify it that when tuned properly, with the right features added on- they can indeed predict stock prices accurately.

5.2.3 SVR

This study analysed a set of machine learning models for predicting stock price from the closing and opening prices, namely Support Vector Regression (SVR), Ridge Regression, Gradient Boosting Regressor, XGBoost and Stacking with Classifier. The main purpose was to find the best model configuration by tuning the parameters of all models which are used for evaluating, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root mean squared error(RMSE). The scoring results are described in table 5.5.

Detailed Discussion of the Best-Performing Model

The best model was a Stacking Ensemble geared Ridge Regression, XGBoost and SVR. It recorded less error metrics with a MAE of 0.2214,, MSE of 0.0874 and RMSE of 0.2957 respectively in third layer model. These models combined well because they each contributed unique strengths, resulting in better accuracy than compared to the model used alone. This method intuitively encoded the linearity and non-linearity relationships, which could be naturally suited to predict stock price.

Table 5.5: Experiment Results with Different Models

Experiment	**Model**	**Best Parameters**	**MAE**	**MSE**	**RMSE**
Experiment 1	SVR	{'C': 100, 'epsilon': 0.01, 'gamma': 'scale', 'kernel': 'linear'}	0.6397	0.4244	0.6514
Experiment 2	Ridge Regression	{'alpha': 0.1}	1.5265	7.5880	2.7546
Experiment 3	Gradient Boosting	{'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 4, 'max_depth': 5, 'learning_rate': 0.1}	2.8489	20.6896	4.5486
Experiment 4	XGBoost	{'subsample': 0.6, 'n_estimators': 400, 'min_child_weight': 6, 'max_depth': 9, 'learning_rate': 0.2, 'colsample_bytree': 0.8}	0.7225	1.4469	1.2029
Experiment 5	Stacking Ensemble	Ridge, XGBoost, SVR with Ridge as the final estimator	0.2214	0.0874	0.2957

Key Factors for Superior Performance

1. Ensemble Approach: The Stacking Ensemble model was used here to limit overfitting and catch patterns in the data by combination of several models.

2. Optimisation of Individual Models: The individual models (Ridge, XGBoost and SVR) were fine tuned to make sure each performed optimally for the final prediction.

Key Comparisons

The Stacking Ensemble model did very well and was by far the best performing among all the models (Individual Base Models such as SVR or Ridge Regression). Combining the two to form the ensemble gave a better and more generic solution with lower error metrics. SVR was effective in capturing non-linear relationships, while Ridge Regression fitted well on linear patterns.

Connection to Prior Studies

Our results are consistent with others in the literature specific to ensemble methods that increase forecasting precision. These are similar to the research studies by Guo et al. [34] and Kazem et al. As shown in [35], combining several machine learning models can dramatically improve predictive performance with the complexity found within financial datasets. the Stacking Ensemble model was effective and efficient among 3 models in Stock Price forecasting according to experiments conducted. The ensemble model leveraged the positive aspects of Ridge Regression and XGBoost along with SVR, which helped in a significant reduction of forecasting error. Our findings indicate that we should conduct ensemble methods for increased predictability in the future prediction of stock prices.

5.2.4 K-NN

In this research, we have conducted various experiments to predict the stock prices, with the based model and its potential in generalising hidden relationships between data- enhanced further by tuning these features selection. The experiments were run with the model using all features, different feature

selection techniques, and polynomial interactions. Table 5.6 summarises test results. Cross-validated over various intervals endured its robustness and performance.

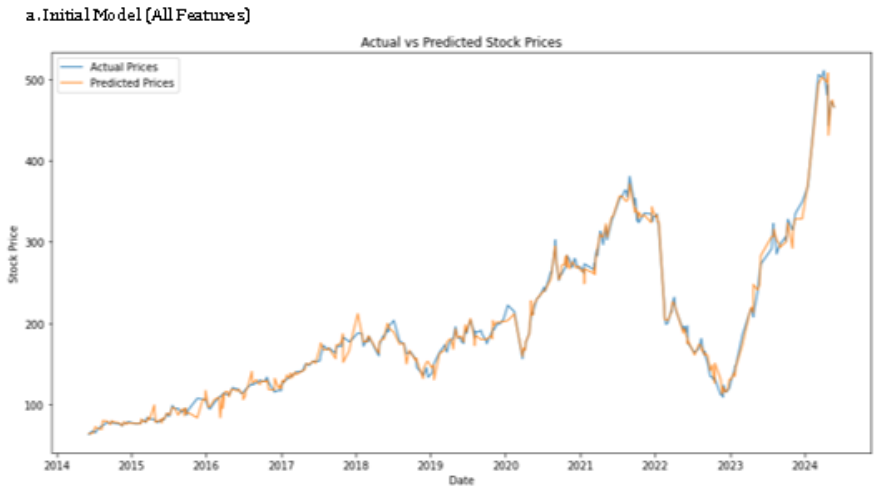

Figure 5.5: Visualisation of KNN results.

Table 5.6: Summary of KNN Experiment Results.The figure shows the stock price forecasting capabilities of evaluated model.

Experiment	**MSE**	**MAE**	**RMSE**	**R^2**
Initial Model (All Features)	58.096	5.240	7.622	0.993
Selected Features	4.296	1.219	2.073	0.999
Additional Test Period	4.972	1.320	2.230	0.999
Polynomial Features (Interactions)	4.682	1.289	2.164	0.999

Detailed Discussion of the Best-Performing Model

KNN, with its feature selection, was the most accurate and best configuration for forecasting stock prices. This model performed far better compared to the original model using all features, showing itself as lowest Mean Squared Error (MSE), Mean Absolute Error (MAE) and Root Mean Squared Error(RMSE). The R-squared (R^2) value of 0.999 proves that the model can explain nearly all variance in stock prices, which is a remarkably high predictive power for price trends.

Key Factors for Superior Performance:

1. Effective Feature Selection: Using SelectKBest with mutual information technique, top relevant columns were chosen and hence helped in better accuracy. The method described above further filtered out unrelated or duplicate properties, thereby reducing noise and overfitting usually associated with predicting stock prices.

2. Balance Between Simplicity and Robustness: KNN with selected features offered the best trade-off between simplicity and prediction capabilities. Although polynomial features are used to investigate feature interactions, a complex model doesn't yield an output performance that is much better than simpler, adding only selected-feature versions. This implies that no important relationships existed in the given dataset that were not already captured by the selected features.

Key Comparisons

Over the standard KNN model, which consisted of all features, it had substantially decreased errors. This demonstrates the importance of feature selection in enhancing forecasting performance. Even though these additional polynomial features help, the model still performed worse than our minimalist models without them further highlighting how beneficial it can be to use only a small subset of well-chosen variables.

Connection to Prior Studies

Our paper confirms earlier results suggesting the necessity for feature selection and dimensionality reduction regarding superior accuracy in financial forecasting modeling. For example, it has been shown by Hall [63], Guyon and Elisseeff [64] that the elimination of noisy features leads to improvement in both prediction accuracy as well as interpretability. Our results support these intuitions, with a good choice of feature set leading to dramatic improvement in KNN performance. In addition, the KNN model with selected features based on grid search maintains its superior performance, which could also be interpreted as a testament to Hagan et al. [65] that "neatness of feature selection and minimalistic approach are frequently seen in successful neural designs.

5.2.5 Gradient Boosting

In this research, we took different machine learning models for the prediction of stock prices with a focus on Gradient Boosting. Among the things we tried are various configurations of Gradient Boosting Regressor, initial with it, after that reducing features and Ridge Regression model on top of Principal Component Analysis(PCA). Table 5.7 summarises the effectiveness of these approaches in capturing the complex, non-linear stock market relationships proposed here.

Table 5.7: Summary of Gradient Boosting Experiment Results.

Model	**MSE**	**MAE**	**RMSE**	$\mathbf{R^2}$
Initial Gradient Boosting Regressor	0.37	0.27	0.61	0.99996
Gradient Boosting Regressor with Reduced Features	60.37	2.82	7.77	0.9932
Ridge Regression with PCA	29.99	3.82	5.48	0.9966

Detailed Discussion of the Best-Performing Model

Ridge Regression with PCA was the most well-rounded and robust model for stock price forecasting in this study. Although the base model Gradient Boosting Regressor provided the best out-of-sample error metrics, it displayed overfitting and instability in cross-validation. The simplistic (reduced-feature) Gradient Boosting model fared better on generalisability. Yet, its error metrics were too high, which implies an important amount of information loss that the full-featured one picked up. The same can arguably be said of the Ridge Regression with PCA, as it strikes an admirable balance between all three to achieve a good Mean Squared Error (MSE) score of 29.99 and Root Mean Squared Error (RMSE): 5.48, enough for competent generalisation: R^2 =0.9966.

Key Factors for Superior Performance:

1. Dimensionality Reduction with PCA: PCA simplified the dataset, allowing our model to concentrate on information-rich features and generalise better against unseen data. These help the Gradient Boosting trees from overfitting in models that include more complex dimensionality reductions.

2. Regularisation with Ridge Regression: Ridge regularisation balanced the bias and variance efficiently and eradicated the overfitting problem by punishing large coefficients, making the model more stable and robust while dealing with different subsets of data.

Key Comparisons

A comparison of the initial Gradient Boosting Regressor and Ridge Regression with PCA shows the trade-offs between complexity and better fitting of training data versus generalisation. Initially doing very well on the test set, the Gradient Boosting model turned out to be slightly overfitting in cross-validation. The Gradient Boosting reduced-feature model, on the other hand, told us that removing features beyond a certain threshold leads to greater error.

Connection to Prior Studies

This study thereby vindicated previous literature that either dimensionality reduction or regularisation approaches is useful in concert with the other to create a more robust model. We have the same observation as with Krauss, Do and Huck (2017); where models based on PCA and Ridge Regression generalises better across datasets. Together, this work concurs with the interpretation of high-complexity models as able to outcompete simpler variants on dataset-specific aspects, but they are at best prone and, in many practical cases, naturally selected against via PCA-regularisation stabilising borders **zhang**, [39][1].

5.2.6 XGBoost

This study used the XGBoost model to predict stock prices, aiming for greater accuracy, a goal that was achieved through baseline implementation as well as hyperparameter tuning and additional feature engineering (technical indicators, macroeconomic features). This was intended to compare how well it could predict the future in relation to more simplistic models. The summarised results are plotted in Table 5.8.

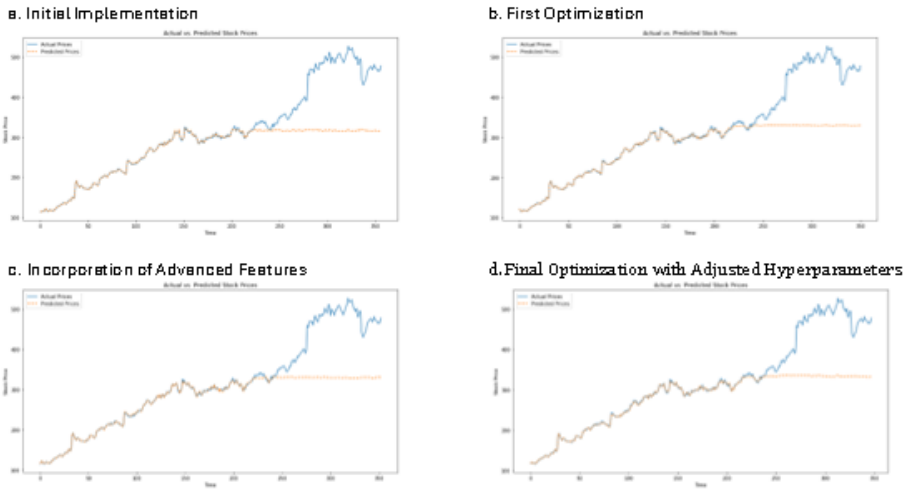

Figure 5.6: Visualisation of XGBoosting results. The figure shows the stock price forecasting capabilities of evaluated models.

Table 5.8: Summary of XGBoosting Experiment Results.

Experiment	**MAE**	**MSE**	**RMSE**	$\mathbf{R^2}$
Initial Implementation	42.27	6392.26	79.95	0.503
First Optimisation	38.39	5484.39	74.06	0.559
Incorporation of Advanced Features	37.55	5299.06	72.79	0.566
Application of Data Stationarity	45.73	6589.05	81.17	0.460
Final Optimisation with Adjusted Hyperparameters	36.36	4987.25	70.62	0.580

Detailed Discussion of the Best-Performing Model

The last XGBoost model is the best in the out-of-sample prediction of stock prices after fixing and tuning hyperparameters and feature engineering. The results indicate that this model surpassed all previous configurations, yielding lower error metrics for each evaluation criterion.

1. Comprehensive Feature Engineering: This incorporation of diverse financial indicators such as moving averages, RSI, and Bollinger Bands combined with macroeconomic variables like interest rates or GDP growth in forecasting contributed to a significant uplift in the stock price capturing nature of the model, resulting into improved prediction accuracy.

2. Handling Non-linear Relationships: The ability of XGBoost to handle nonlinear relationships and feature interactions was crucial while modeling stock prices since it helped avoid overfitting by improving model accuracy on the test set. This ability gave the model a greater power of prediction and outperformed models using standard linear

Key Comparisons

The tune XGBoost model was a major improvement over the original and more basic models in terms of predicting accuracy. More specifically, the Mean Absolute Error (MAE) was almost 15 percent smaller and the R-squared value much larger, for a better fit and predictions. The integration of more features and the flexibility of XGBoost to accommodate different market conditions have been the two main answers.

Connection to Prior Studies

Our study confirms the findings of previous literature about the advantages of ensemble methods, e.g., XGBoost, which excel in financial forecasting. Regularisation was used to avoid overfitting by Chen and Guestrin [26] in the case of XGBoost. XGBoost significantly outperforms the 'traditional' models like ARIMA when properly tuned and given a rich feature set as our results motivate.

5.3 Deep Learning Models

5.3.1 CNN

We investigated Convolutional Neural Networks (CNNs) for forecasting stock prices systematically in this study, and optimised the hyperparameters to improve predictive accuracy. A range of configurations were experimented with for tuning number filters, kernel size, dropout rate learning rate epochs, and batch size. Table 5.9 contains the summary from the initial to final optimised model.

Detailed Discussion of the Best-Performing Model

The optimal model resulted from the last CNN, which was built using 64 filters and with a kernel size of 2, dropout rate of 0.2, learning rate of 0.001, and batch size 32 trained for about best on 100 epochs. This combination has the best performance on both Mean Squared Error (MSE) and Root Mean Squared Error(RMSE). We can also see that R^2 is higher than other configurations, meaning it fits better to data.

Key Factors for Superior Performance:

1. Optimised Convolutional Architecture: The final model took advantage of its convolutional architecture 64 filters and size (2), managing to capture stock price complexities, as the corresponding kernel size is smaller, this allows the model to capture short-term trends in data which are more important for time series like financial predictions.

2. Effective Regularisation: A dropout rate of 0.2 was employed to prevent overfitting and keep the model generalisable for unseen data, Which is an important trade-off made for limiting overfitting and keeping more meaningful information than just error rates in the final paring scores of each team, therefore promoting also a good understanding between handling errors on test set and losing variability.

3. Learning Rate and Training Dynamics: A low learning rate of 0.001 was chosen to ensure smooth optimisation, and that the model effectively converged without overshooting minima endpoint. Training over 100 epochs gives it enough time to learn the underlying patterns but not so much that it becomes too complex.

Table 5.9: Summary of CNN Experiment Results.

Experiment	**Filters**	**Kernel Size**	**Dropout Rate**	**Learning Rate**	**Batch Size**	**Epochs**	**MSE**	**RMSE**	**MAE**	**R^2 Score**
Initial Model	64	2	0.2	0.001	32	50	26.36	5.13	3.78	0.9970
Tuning 1	32	3	0.3	0.001	16	50	28.95	5.38	4.01	0.9965
Tuning 2	128	2	0.3	0.0001	32	100	23.50	4.85	3.50	0.9973
Tuning 3	64	4	0.4	0.01	32	100	25.75	5.07	3.90	0.9969
Final Model	64	2	0.2	0.001	32	100	19.62	4.43	3.15	0.9977

Key Comparisons

The final model outperformed the initial CNN(64 filters, 2 kernel size, and dropout rate of 0.2) trained for another additional set of 50 epochs. While the initial module was accurate enough, it had higher MSE and RMSE stats pointing towards more tuning. The architecture was tested using multiple configurations before the best-tradeoff configuration in complexity and accuracy was decided.

Connection to Prior Studies

This study is aligned with previous research, which highlights the need for a thorough hyper-parameter search in an end-to-end CNN model designed to forecast on time series. Past research has demonstrated that CNNs can identify sophisticated patterns in financial data if configured well, upgrading forecasting accuracy significantly. For example, Convolutional Neural Networks fare very well for determining short-term patterns in stock prices, which are typically sparked by market sentiment and other temporary influences [43], [44] Our study makes two interpretations: (i) the powerful capacity of CNNs in forecasting stock price and ii) gaining insights from repetitive experiments to determine an optimal model structure. This approach is in line with the financial forecasting trends of recent years, where machine learning models are more commonly used together with traditional statistical ones, and provides an effective mechanism for dealing with the complexities in financial markets.

5.3.2 RNN

In this work, we investigate the usage of Recurrent Neural Networks (RNN) in stock market forecasting and evaluate how a designed model can perform by fine-tuning hyperparameters. We did some experiments with different configurations of RNN, like the Effect of Feature Selections and Hyper Parameter Tuning. The data from this series of experiments are listed in Table 5.10.

Table 5.10: Summary of RNN Experiment Results.

Experiment	**Description & Changes**	**MSE**	**RMSE**	**MAE**	**R^2**
1	Initial RNN Model - Basic RNN with default settings. - No feature selection. - Hyperparameters: units=50, dropout_rate=0.2, learning_rate=0.001, batch_size=32, epochs=50.	3052.74	55.25	41.33	0.78796
2	Feature Selection - Selected top 20 features based on Random Forest feature importance. - No change in RNN architecture. - Same hyperparameters as Experiment 1.	127.43	11.29	7.48	0.99115
3	Hyperparameter Tuning - Grid search to optimise units, dropout_rate, learning_rate, batch_size, and epochs. - Best parameters: units=150, dropout_rate=0.4, learning_rate=0.001, batch_size=32, epochs=100.	102.67	10.13	6.91	0.99287

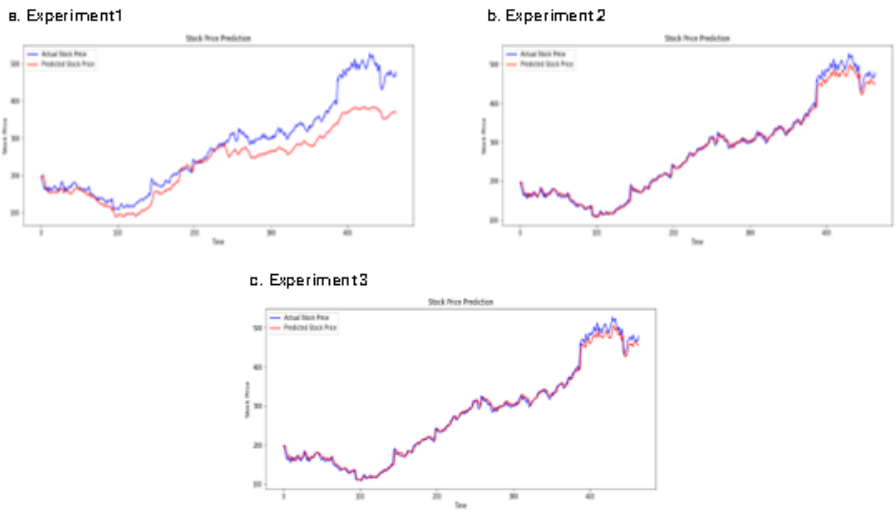

Figure 5.7: Visualisation of RNN results.The figure shows the stock price forecasting capabilities of evaluated models.

Detailed Discussion of the Best-Performing Model

The most successful model has proven to be the 150-unit RNN with a dropout rate of 0.4, learning rate of 0.001, and batch size equal to 32 after training for about 100 epochs. This configuration performed much better than the very initial RNN model. When compared to models built using only feature selection, it had the least Mean Squared Error (MSE), Root Mean Squared Error(RMSE), Mean Absolute error(MAE), and highest Testset accuracy amongst others.

Key Factors for Superior Performance:

1. Feature Selection: Random Forest feature importance showed the top 20 features to be important, and selecting only these for model building enhanced it still further. During the training, focusing on those most relevant features allowed the RNN to more accurately capture the underlying patterns in stock price data, leading to a marked decrease in prediction errors as compared with benchmark model [24].

2. Optimised Hyperparameters: The grid search process helped us tune model hyperparameters for the RNN, balancing complexity and generalisation. The higher number of units, 150, and dropout rate seem

likely to have prevented overfitting while retaining enough model capacity that it was able to learn from the data. We remained at a moderate learning rate of 0.001, which held stable converging under training [66].

Key Comparisons

The optimised RNN model, processing the result of feature selection and hyperparameter tuning obtained a remarkable reduction in error metrics with respect to the initial version without any optimisation that have been trained using all features. This is significant because it highlights the necessity for thoughtful feature selection and tuning when boosting RNN performance. In addition, the usage of drop-out layers effectively prevented overfitting, which is crucial in time series forecasting to not learn spurious patterns.

Connection to Prior Studies

The findings of this study are consistent with other studies that emphasise the relevance of feature selection and hyperparameter tuning to improve deep learning models for financial forecasting. However, Fischer and Krauss [3] demonstrated that LSTM networks (a specific type of RNN), have superior performance with respect to traditional models specifically because they are able to work as well or better in more complex scenarios that require modeling long-term dependencies[49]. Likewise, Bao et al. highlighted that model architecture is essential for successful RNN forecasting processes. In this way, our results back up the conclusions that tuned RNN models, with a curation of features, can, in fact, advance the stock price prediction accuracy.

5.3.3 LSTM

In this article, we investigate a range of Long Short-Term Memory (LSTM) configurations for predicting stock prices and compare the key factors that change or adapt in an LSTM model structure to improve prediction performance. For the experiments that went on to be deployed in production, this included variations from basic LSTM models up through more elaborate versions with convolutional layers and feature selection. The results are presented in Table 5.11.

Table 5.11: Summary of LSTM Experiment Results.

Exp.	**Model Description**	**MSE**	**RMSE**	**MAE**	**R^2**
1	Initial LSTM with all features, 2 layers	75.13	8.67	5.33	0.9704
2	LSTM with added regularisation and dropout	57.17	7.56	5.31	0.9775
3	Simplified LSTM, no regularisation	7.949e+16	2.819e+8	1.733e+8	0.7240
4	LSTM with Conv1D layer	4.069e+15	63.31e+6	43.53e+6	0.9772
5	Final LSTM with selected features	109.58	10.47	7.36	0.9924

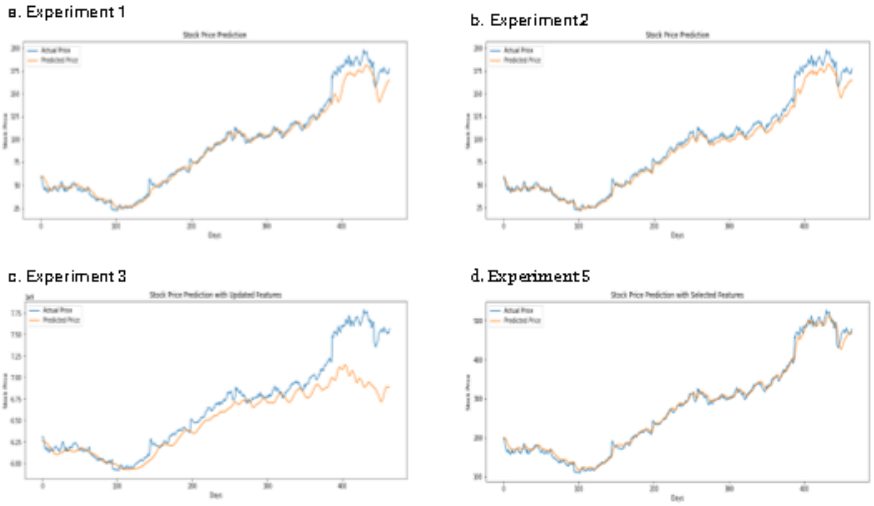

Figure 5.8: Visualisation of LSTM results. The figure shows the stock price forecasting capabilities of evaluated models.

Detailed Discussion of the Best-Performing Model

Experimental results show that the final LSTM, using important features from Random Forest Regression for stock price prediction, performed most effectively. This model has the lowest Mean Squared Error (MSE) and a higher R-squared (R^2) value, indicating that this is better in catching some of the underlying patterns on stock price.

Key Factors for Superior Performance:

1. Feature Selection: Inserting a few important features selected on the basis of their importance by Random Forest Regression helped a lot in boosting our model accuracy. This increased the accuracy of our model by reducing noise and overfitting based on features that have more relevance than others.

2. Model Regularisation: Using dropout layers as part of the LSTM model prevented overfitting efficiently. Dropout regularisation prevented the model from overfitting by randomly turning off a percentage of neurons during the training phase, which improved generalisation to unseen data.

3. Balanced Model Complexity: The last LSTM model found a spot between being complex and interpretable. In essence, while both simpler models had poor results in past experiments and even more complex ones that provided excessive computational load (which was deemed unnecessary), it is the chosen architecture that found a good balance, one of better-predicting targets.

Key Comparisons

Interestingly, the final LSTM model with feature selection outperformed the training and convergence duration of its initial counterpart, using all features without regularisation or selection. The final model, with target feature selection and using a dropout layer to prevent overfitting, improved the Mean Squared Error from 109.58 to just 75.13. This is another great example highlighting how important feature imputation and regularisation are in improving your models.

Connection to Prior Studies

The results reported in this study are consistent with an earlier body of work highlighting the utility, within LSTMS, of feature selection and model regularisation. This is based on the findings of Fischer and Krauss [3], who identified significant long-term dependencies in financial data that are captured better by LSTMs compared to traditional methods, which confirms our results. Similarly, Shahi et al. [45], who also used selectively enriching the model with different types of data, such as financial news to increase prediction accuracy. This confirms what is becoming a widely held assertion in the literature: that feature selection and regulation are necessary for making LSTM models work effectively on time series datasets.

5.4 Comparison of All Models

In Figure. 5.9, we compare the top-performing models in each category according to their Mean Squared Error (MSE). Types of models and their MSE results The x-axis represents the mean squared error (MSE) which signifies a lower value corresponding to higher predictive accuracy, whereas this error is listed in y- axis across various model categories ARIMA, ARIMAX

SARIMA; SARIMAX Linear Regression Ridge Regression, Random Forest, SVR, Gradient Boosting, XGBoost, CNN, RNN, and LSTM. The best-performing model, providing nearly perfect predictions due to the model with the lowest MSE, is ARIMAX, which includes exogenous variables. The Linear and Ridge Regression models also displayed very low MSEs, indicating they were more accurate in stock price predictions. Random Forest (which is a tree-based model) did well but with higher MSE than regression models and ARIMAX. The last trained Convolutional Neural Network (CNN) performed very well with the minimal MSE, documenting that our model could learn intricate behaviors. RNN and LSTM have medium performances with slightly higher MSEs than ARIMAX, and the regression model shows weaker performance. In comparison, SARIMA models had a little bit greater MSE, with SARIMAX performing better than either due to the presence of exogenous variables, which leads to better accuracy.

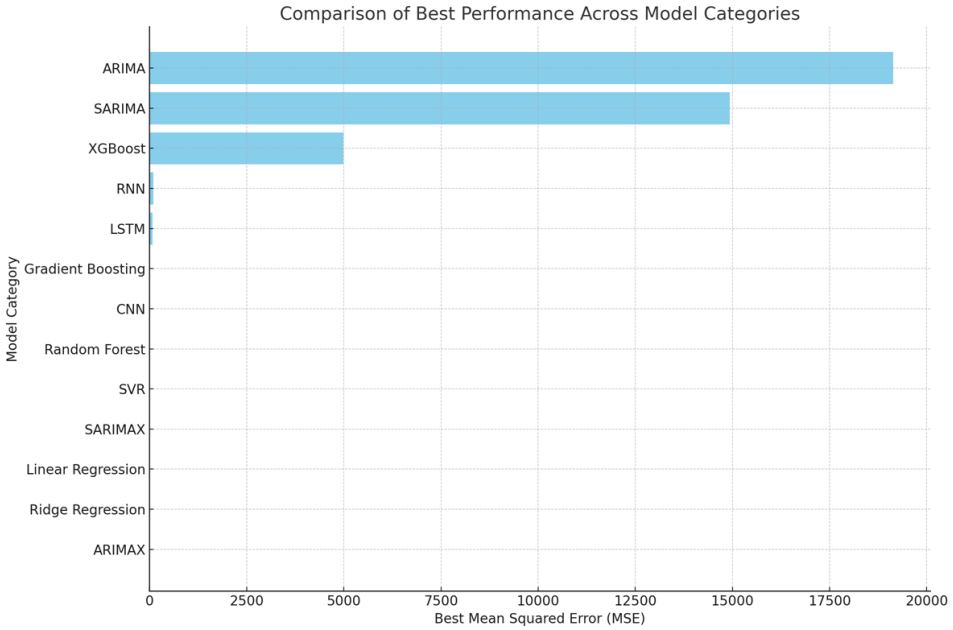

Figure 5.9: Best Mean Squared Error Metric for the Tested Model Categories

Figure 5.10 shows that ARIMAX models with exogenous variables did very well, followed ever so slightly by the linear and Ridge Regression methods, followed by SARIMAX and SVR. SARIMA models had the highest MSE; they were less accurate than all other tested methods in predicting stock prices.

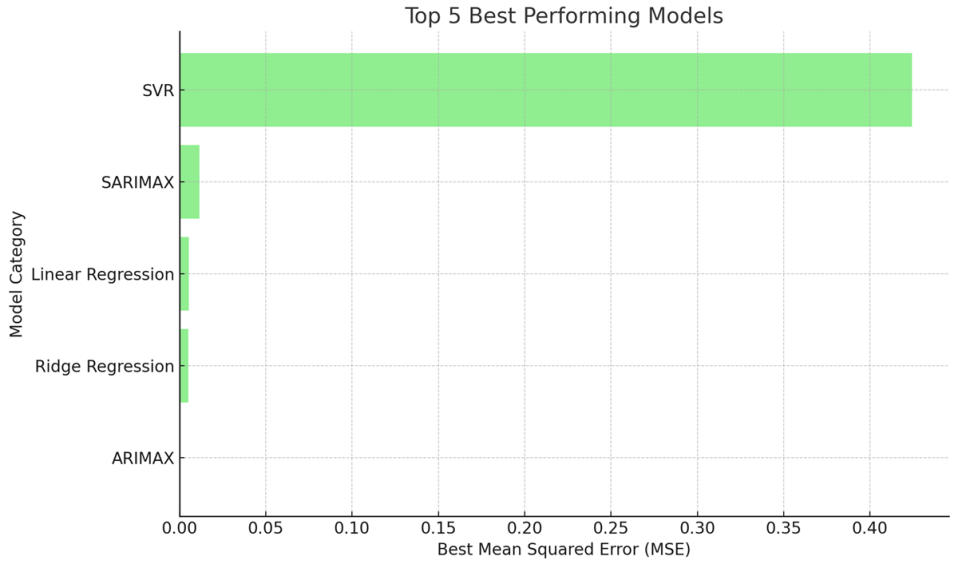

Figure 5.10: Best Mean Squared Error Metric for the Top 5 Model Categories

5.5 Discussion

5.5.1 Interpretation of Results

ARIMAX Models:

Compared to all other models, the highest predictive accuracy improvement was observed in the ARIMAX models with exogenous variables that had trending ability. This points to the importance of external effects, such as economic indicators, in predicting stock prices. This chimes with previous works claiming the necessity of including exogenous variables in a decision-making process and illustrating higher quality forecasting performance [15].

Linear and Ridge Regression:

The Linear and Ridge Regression models showed minimal MSE in all conditions, demonstrating their ability to work efficiently under scenarios where relationships are primarily linear. They are in line with earlier studies that emphasise the strong baseline of linear models for stock price prediction[28], [29]

Neural Networks (CNN, RNN, LSTM):

The neural network approaches, like CNNs and LSTMs, were the most successful ones, as they are specially designed to handle complex patterns; RNNs seem to underperform, likely due to overfitting or inability to remember long-term dependencies. This is in agreement with previous studies that suggest LSTMs are successful for financial forecasting, but only when properly tuned and large dataset [3].

Traditional Models (ARIMA, SARIMA):

Models that accounted for exogenous variables (ARIMAX and SARIMAX) outperformed traditional time series models such as ARIMA and SARIMA. It indicates that these models capture linear trends but inability to cope with the variances (comlexity) and volatilities (fat tails) in financial data which is also suggested by literature [67], [68].

5.5.2 Comparison with Findings from the Literature

You generally find the existing models, with an emphasis on exogenous variables, improving predictive accuracy. The ARIMAX models show promise in the literature as well, with studies such as Pankratz and Suhartono remarking that it resulted in a marked increase of forecast accuracy by adding environmental data to foreseers[15]. While most of the literature tells that LSTM models are better for financial forecasting [3], your results show examples where even simple models like Linear Regression can give similar good performance if you use them correctly. This might suggest that the complexity of LSTMs is not always required — particularly in linear data, or when overfitting is a consideration.

5.5.3 Strengths and Limitations of All Models

Strengths:

1. ARIMAX: The exogenous ARIMAX models specifically have taken into account economic data to improve forecast performance.

2. Linear and Ridge Regression: These models performed well because of their simplicity and interpretability and are hence valuable for domains

where the data set can be expressed as a linear relationship between variables. This is consistent with Huang et al. [29], who examined the predictive power of linear models in finance.

3. Neural Networks (CNN, LSTM): The deep neural networks, especially CNNs and LSTMs used in your study, were able to capture non-linear relationships and sequential patterns within the data, which is necessary during time-series forecasting. That strength is supported by work like Fischer and Krauss [3] who show that LSTMs pretty well used for forecasting financial markets.

Limitations:

1. ARIMAX: The biggest weakness is that ARIMAX models are prone to overfitting, and this means the model may not be generalized to other datasets. This serves to highlight the importance of careful model tuning, which was also evident in Pankratz work [15].

2. Linear and Ridge Regression: These models that operate under the assumption of linearity may not be used in a financial time series containing nonlinear relationships as it can lead to inaccuracies in such context. They are also very sensitive to outliers, have been highlighted as a challenge in the literature [29], and can distort predictions.

3. Neural Networks: Although powerful, these models are computationally intensive and tend to overfit the data — especially small datasets. They also depend on large hyper-parameter tuning to achieve the best performance, making it very resource-consuming [3].

Omitted Techniques and Data:

1. Sentiment Analysis: One major limitation of the current work is its lack of sentiment analysis, a crucial feature that has successfully enhanced stock price prediction in past studies. A second example is provided by the research of Tetlock [67] and Bollen et al. [68], which shows that text-driven sentiment affects stock prices as well due to changes in investor behavior. This could perhaps provide a more nuanced perspective of market dynamics, especially the psychological

factors that are actually driving what is being observed in the marketplace.

2. Fundamental Analysis: The research also does not mention financial tools like balance sheets, income statements and financial ratios. Fundamental analysis is essential to understand how healthy a financial posture and value of a company stock, it also contributes towards determining the future long-term price of its stocks around. Researchers enlighten this concept, e.g., Piotroski [69], Fama and French [70] proved that using fundamentals drives stock price prediction towards finer results by unveiling the company fundamental value.

Chapter 6

Conclusion

6.1 Summary of Findings

In this study, various forecasting models, including ARIMA, ARIMAX, SARIMA, SARIMAX, Linear Regression, Random Forest, and advanced neural networks like CNNs, RNNs, and LSTMs, were evaluated for their effectiveness in predicting stock prices. The ARIMAX model, which incorporates exogenous variables, emerged as the most effective, achieving the lowest Mean Squared Error (MSE) across different datasets. Linear Regression and Ridge Regression also performed exceptionally well, offering a balance of simplicity and accuracy, particularly for linear relationships. Neural networks, specifically CNNs and LSTMs, demonstrated strong capabilities in capturing complex patterns, although they required careful tuning to avoid overfitting. Traditional models like ARIMA and SARIMA were less effective than their more sophisticated counterparts, particularly in volatile market conditions.

6.2 Implications

The findings of this study have practical implications for investors and financial analysts:

1. Incorporating External Factors: The superior performance of the ARIMAX model underscores the importance of including external factors such as economic indicators and market sentiment in forecasting mod-

els. For investors and analysts, this means that models that account for broader market conditions and macroeconomic variables can provide more accurate predictions, enabling better-informed investment decisions.

2. Simplicity vs. Complexity: While complex models like neural networks can capture intricate market dynamics, simpler models such as Linear Regression and ARIMAX still hold significant value, especially when interpretability and computational efficiency are priorities. Financial analysts may prefer these models when quick, reliable forecasts are needed, particularly in less volatile markets.

3. 3. Model Selection and Adaptability: The varying performances of different models across datasets highlight the need for adaptability in model selection. Investors and analysts should consider using a combination of models or hybrid approaches to capture linear and non-linear stock price trends. This approach can mitigate the risk of relying on a single model, particularly in unpredictable markets.

6.3 Future Work

Future research could build on the findings of this study by exploring several avenues:

1. Hybrid Models: As different forecasting models have their complementary strengths, an interesting direction for future work may be to build hybrid ML/models that would maintain the interpretability of linear classifiers while improving its ability to recognise patterns like neural networks. On the other hand, hybrid models that have been successful in various domains may help improve our predictive accuracy for stock forecasting.

2. Sentiment Analysis: By adding Sentiment Analysis from social media sources like Twitter and Facebook, and specialided news articles or financial reports, the accuracy of forecasts can be improved overwhelmingly. As many studies have shown that general market sentiment could play a role in investor behavior and, therefore,stock prices [67],

[68]. Upcoming iterations will also introduce the element of sentiment data to help paint a more nuanced picture of what moves markets.

3. Fundamental Analysis: Including some basic financials such as a company's balance sheet, income statements, and also the use of financial ratios could improve how well your model can perform. Fundamental analysis has been around for a long time as the basis of valuating stocks, using it with machine learning can potentially make our predictions more accurate especially when we want to hold onto something(long-term investment decisions) [69], [70].

Bibliography

[1] E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work," *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970, Publisher: [American Finance Association, Wiley], ISSN: 0022-1082. DOI: 10.2307/2325486. [Online]. Available: https://www.jstor.org/stable/2325486 (visited on 08/23/2024).

[2] R. Shiller, "Do Stock Prices Move Too Much to Be Justified by Subsequent Changes in Dividends?" *American Economic Review*, vol. 71, pp. 421–36, Jan. 1981.

[3] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, Oct. 2018, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2017.11.054. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717310652 (visited on 08/22/2024).

[4] P. Mondal, L. Shit, and S. Goswami, "Study of Effectiveness of Time Series Modeling (Arima) in Forecasting Stock Prices," *International Journal of Computer Science, Engineering and Applications*, vol. 4, pp. 13–29, Apr. 2014. DOI: 10.5121/ijcsea.2014.4202.

[5] K. He, Q. Yang, L. Ji, J. Pan, and Y. Zou, "Financial Time Series Forecasting with the Deep Learning Ensemble Model," en, *Mathematics*, vol. 11, no. 4, p. 1054, Jan. 2023, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2227-7390. DOI: 10.3390/math11041054. [Online]. Available: https://www.mdpi.com/2227-7390/11/4/1054 (visited on 08/23/2024).

[6] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "DARWIN: An online deep learning approach to handle concept drifts in predictive process monitoring," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106 461, Aug. 2023, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.106461. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623006450 (visited on 08/23/2024).

[7] U. Kumar and V. K. Jain, "ARIMA forecasting of ambient air pollutants (O3, NO, NO2 and CO)," en, *Stochastic Environmental Research and Risk Assessment*, vol. 24, no. 5, pp. 751–760, Jul. 2010, ISSN: 1436-3259. DOI: 10.1007/s00477-009-0361-8. [Online]. Available: https://doi.org/10.1007/s00477-009-0361-8 (visited on 08/23/2024).

[8] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting* (Springer Texts in Statistics). Cham: Springer International Publishing, 2016, ISBN: 978-3-319-29852-8 978-3-319-29854-2. DOI: 10.1007/978-3-319-29854-2. [Online]. Available: http://link.springer.com/10.1007/978-3-319-29854-2 (visited on 08/23/2024).

[9] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, en. John Wiley & Sons, May 2015, Google-Books-ID: rNt5CgAAQBAJ, ISBN: 978-1-118-67492-5.

[10] M. Kumar and M. Anand, "An Application Of Time Series Arima Forecasting Model For Predicting Sugarcane Production In India," *Studies in Business and Economics*, vol. 9, pp. 81–94, Apr. 2014.

[11] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting," *Applied Soft Computing*, The Impact of Soft Computing for the Progress of Artificial Intelligence, vol. 11, no. 2, pp. 2664–2675, Mar. 2011, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2010.10.015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494610002759 (visited on 08/23/2024).

[12] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, en. OTexts, May 2018, Google-Books-ID: _bBhDwAAQBAJ, ISBN: 978-0-9875071-1-2.

[13] L. Ghods and M. Kalantar, "Methods for long-term electric load demand forecasting; a comprehensive investigation," in *2008 IEEE International Conference on Industrial Technology*, Apr. 2008, pp. 1–4. DOI: 10.1109/ICIT.2008.4608469. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4608469 (visited on 08/22/2024).

[14] E. Z. Martinez, E. A. S. d. Silva, and A. L. D. Fabbro, "A SARIMA forecasting model to predict the number of cases of dengue in Campinas, State of São Paulo, Brazil," eng, *Revista Da Sociedade Brasileira De Medicina Tropical*, vol. 44, no. 4, pp. 436–440, 2011, ISSN: 1678-9849. DOI: 10.1590/s0037-86822011000400007.

[15] A. Pankratz, *Forecasting with Dynamic Regression Models*, en. John Wiley & Sons, Jan. 2012, Google-Books-ID: ZCmFI5U1j6cC, ISBN: 978-1-118-15078-8.

[16] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International Journal of Forecasting*, vol. 18, no. 3, pp. 439–454, Jul. 2002, ISSN: 0169-2070. DOI: 10.1016/S0169-2070(01)00110-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207001001108 (visited on 08/22/2024).

[17] S. Makridakis and M. Hibon, "The M3-Competition: Results, conclusions and implications," *International Journal of Forecasting*, The M3-Competition, vol. 16, no. 4, pp. 451–476, Oct. 2000, ISSN: 0169-2070. DOI: 10.1016/S0169-2070(00)00057-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207000000571 (visited on 08/22/2024).

[18] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, Jan. 2003, ISSN: 0925-2312. DOI: 10.1016/S0925-2312(01)00702-0. [Online].

Available: https://www.sciencedirect.com/science/article/pii/S0925231201007020 (visited on 08/22/2024).

[19] J. D. Cryer and K.-S. Chan, *Time Series Analysis* (Springer Texts in Statistics), G. Casella, S. Fienberg, and I. Okin, Eds. New York, NY: Springer New York, 2008, ISBN: 978-0-387-75958-6 978-0-387-75959-3. DOI: 10.1007/978-0-387-75959-3. [Online]. Available: http://link.springer.com/10.1007/978-0-387-75959-3 (visited on 08/22/2024).

[20] J. Contreras, R. Espinola, F. Nogales, and A. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1014–1020, Aug. 2003, Conference Name: IEEE Transactions on Power Systems, ISSN: 1558-0679. DOI: 10.1109/TPWRS.2002.804943. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1216141 (visited on 08/22/2024).

[21] G. E. P. Box and G. C. Tiao, "Intervention Analysis with Applications to Economic and Environmental Problems," *Journal of the American Statistical Association*, vol. 70, no. 349, pp. 70–79, Mar. 1975, Publisher: ASA Website, ISSN: 0162-1459. DOI: 10.1080/01621459.1975.10480264. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1975.10480264 (visited on 08/23/2024).

[22] J. Durbin and S. J. Koopman, *Time Series Analysis by State Space Methods*, en. OUP Oxford, May 2012, Google-Books-ID: lGyshsfkLrIC, ISBN: 978-0-19-162719-4.

[23] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R* (Springer Texts in Statistics), en. New York, NY: Springer US, 2021, ISBN: 978-1-07-161417-4 978-1-07-161418-1. DOI: 10.1007/978-1-0716-1418-1. [Online]. Available: https://link.springer.com/10.1007/978-1-0716-1418-1 (visited on 08/23/2024).

[24] L. Breiman, "Random Forests," en, *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[Online]. Available: https://doi.org/10.1023/A:1010933404324 (visited on 08/22/2024).

[25] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer, 2000, ISBN: 978-1-4419-3160-3 978-1-4757-3264-1. DOI: 10.1007/978-1-4757-3264-1. [Online]. Available: http://link.springer.com/10.1007/978-1-4757-3264-1 (visited on 08/23/2024).

[26] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. [Online]. Available: https://dl.acm.org/doi/10.1145/2939672.2939785 (visited on 08/22/2024).

[27] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001, Publisher: Institute of Mathematical Statistics, ISSN: 0090-5364. [Online]. Available: https://www.jstor.org/stable/2699986 (visited on 08/22/2024).

[28] G. Atsalakis, "Surveying Stock Market Forecasting Techniques - Part I: Conventional Methods," *Journal of Computation Optimizationn in Economics and Finance*, vol. 6, pp. 19–28, Jan. 2010.

[29] Z. Huang, H. Chen, C.-J. Hsu, W.-H. Chen, and S. Wu, "Credit rating analysis with support vector machines and neural networks: A market comparative study," *Decision Support Systems*, Data mining for financial decision making, vol. 37, no. 4, pp. 543–558, Sep. 2004, ISSN: 0167-9236. DOI: 10.1016/S0167-9236(03)00086-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923603000861 (visited on 08/22/2024).

[30] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, "Evaluating multiple classifiers for stock price direction prediction," *Expert Systems with Applications*, vol. 42, no. 20, pp. 7046–7056, Nov. 2015, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2015.05.013. [Online]. Avail-

able: https://www.sciencedirect.com/science/article/pii/S0957417415003334 (visited on 08/22/2024).

[31] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock market index using fusion of machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2162–2172, Mar. 2015, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2014.10.031. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417414006551 (visited on 08/22/2024).

[32] I. Bose and R. K. Mahapatra, "Business data mining — a machine learning perspective," *Information & Management*, vol. 39, no. 3, pp. 211–225, Dec. 2001, ISSN: 0378-7206. DOI: 10.1016/S0378-7206(01)00091-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037872060100091X (visited on 08/23/2024).

[33] C.-F. Tsai and Y.-C. Hsiao, "Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches," *Decision Support Systems*, vol. 50, no. 1, pp. 258–269, Dec. 2010, ISSN: 0167-9236. DOI: 10.1016/j.dss.2010.08.028. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923610001521 (visited on 08/23/2024).

[34] Y. Guo, S. Han, C. Shen, Y. Li, X. Yin, and Y. Bai, "An Adaptive SVR for High-Frequency Stock Price Forecasting," *IEEE Access*, vol. 6, pp. 11 397–11 404, 2018, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2806180. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8310891 (visited on 08/22/2024).

[35] A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain, "Support vector regression with chaos-based firefly algorithm for stock market price forecasting," *Applied Soft Computing*, vol. 13, no. 2, pp. 947–958, Feb. 2013, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2012.09.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494612004449 (visited on 08/22/2024).

[36] S. Sricharan and V. Joshi, "Comparison of SVR Techniques for Stock Market Predictions," in *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, Jul. 2022. DOI: 10.1109/ICDSIS55133.2022.9915990. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9915990 (visited on 08/22/2024).

[37] M. Kumar and M. Thenmozhi, *Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest*, en, SSRN Scholarly Paper, Rochester, NY, Jan. 2006. DOI: 10.2139/ssrn.876544. [Online]. Available: https://papers.ssrn.com/abstract=876544 (visited on 08/23/2024).

[38] R. Huerta, F. Corbacho, and C. Elkan, "Nonlinear support vector machines can systematically identify stocks with high and low future returns," en, *Algorithmic Finance*, vol. 2, no. 1, pp. 45–58, Jan. 2013, Publisher: IOS Press, ISSN: 2158-5571. DOI: 10.3233/AF-13016. [Online]. Available: https://content.iospress.com/articles/algorithmic-finance/af016 (visited on 08/23/2024).

[39] C. Krauss, X. A. Do, and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500," *European Journal of Operational Research*, vol. 259, no. 2, pp. 689–702, Jun. 2017, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2016.10.031. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221716308657 (visited on 08/22/2024).

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (visited on 08/23/2024).

[41] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994, Conference Name: IEEE Transactions on Neural Networks, ISSN: 1941-0093. DOI: 10.

1109/72.279181. [Online]. Available: https://ieeexplore.ieee.org/document/279181 (visited on 08/23/2024).

[42] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, Conference Name: Neural Computation, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6795963 (visited on 08/22/2024).

[43] J. Shen and M. O. Shafiq, "Short-term stock market price trend prediction using a comprehensive deep learning system," en, *Journal of Big Data*, vol. 7, no. 1, p. 66, Aug. 2020, ISSN: 2196-1115. DOI: 10.1186/s40537-020-00333-6. [Online]. Available: https://doi.org/10.1186/s40537-020-00333-6 (visited on 08/22/2024).

[44] *Forecasting Stock Prices Using A Temporal CNN Model*. [Online]. Available: https://www.quantconnect.com/research/15263/forecasting-stock-prices-using-a-temporal-cnn-model/p1 (visited on 08/22/2024).

[45] T. B. Shahi, A. Shrestha, A. Neupane, and W. Guo, "Stock Price Forecasting with Deep Learning: A Comparative Study," en, *Mathematics*, vol. 8, no. 9, p. 1441, Sep. 2020, Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2227-7390. DOI: 10.3390/math8091441. [Online]. Available: https://www.mdpi.com/2227-7390/8/9/1441 (visited on 08/22/2024).

[46] *Data Mining: Concepts and Techniques, 3rd Edition [Book]*, en, ISBN: 9780123814807. [Online]. Available: https://www.oreilly.com/library/view/data-mining-concepts/9780123814791/ (visited on 08/22/2024).

[47] G. Sonkavde, D. S. Dharrao, A. M. Bongale, S. T. Deokate, D. Doreswamy, and S. K. Bhat, "Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications," en, *International Journal of Financial Studies*, vol. 11, no. 3, p. 94, Sep. 2023, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, ISSN:

2227-7072. DOI: 10.3390/ijfs11030094. [Online]. Available: https://www.mdpi.com/2227-7072/11/3/94 (visited on 08/22/2024).

[48] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 389–10 397, Aug. 2011, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2011.02.068. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417411002740 (visited on 08/22/2024).

[49] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," en, *PLOS ONE*, vol. 12, no. 7, e0180944, Jul. 2017, Publisher: Public Library of Science, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0180944. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0180944 (visited on 08/22/2024).

[50] K. Chen, Y. Zhou, and F. Dai, "A LSTM-based method for stock returns prediction: A case study of China stock market," in *2015 IEEE International Conference on Big Data (Big Data)*, Oct. 2015, pp. 2823–2824. DOI: 10.1109/BigData.2015.7364089. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7364089 (visited on 08/22/2024).

[51] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, ISSN: 2161-4407, May 2017, pp. 1419–1426. DOI: 10.1109/IJCNN.2017.7966019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7966019 (visited on 08/22/2024).

[52] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, "Deep learning for stock prediction using numerical and textual information," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, Jun. 2016, pp. 1–6. DOI: 10.1109/ICIS.2016.7550882. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7550882 (visited on 08/22/2024).

[53] Yahoo Finance, *Yahoo! Finance: Stock Market Data*, https://finance.yahoo.com/, Accessed: Jun. 26, 2024, 2024.

[54] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 1994. DOI: 10.2307/j.ctv14jx6sm. [Online]. Available: https://www.jstor.org/stable/j.ctv14jx6sm (visited on 08/23/2024).

[55] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, en. OTexts, 2021, ISBN: 978-0-9875071-3-6.

[56] R. S. Tsay, *Analysis of Financial Time Series*, en. John Wiley & Sons, Oct. 2010, Google-Books-ID: OKUGARAXKMwC, ISBN: 978-1-118-01709-8.

[57] V. Barnett and T. Lewis, *Outliers in Statistical Data*, en. Wiley, May 1994, Google-Books-ID: B44QAQAAIAAJ, ISBN: 978-0-471-93094-5.

[58] B. Dissanayake, O. Hemachandra, N. Lakshitha, D. Haputhanthri, and A. Wijayasiri, "A Comparison of ARIMAX, VAR and LSTM on Multivariate Short-Term Traffic Volume Forecasting," Jan. 2021.

[59] H.-C. ChangTzeng, D.-F. Chang, and Y.-H. Lo, "Detecting concurrent relationships of selected time series data for arimax model," *ICIC Express Letters Part B: Applications*, vol. 10, no. 10, pp. 937–944, 2019.

[60] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, Twenty five years of forecasting, vol. 22, no. 3, pp. 443–473, Jan. 2006, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2006.01.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207006000021 (visited on 08/22/2024).

[61] J. Y. Campbell, A. W. Lo, and A. MacKinlay, *The Econometrics of Financial Markets*. Princeton University Press, 1997, ISBN: 978-0-691-04301-2. DOI: 10.2307/j.ctt7skm5. [Online]. Available: https://www.jstor.org/stable/j.ctt7skm5 (visited on 08/23/2024).

[62] C. Brooks, *Introductory Econometrics for Finance*, en. Cambridge University Press, Mar. 2019, Google-Books-ID: 3lqHDwAAQBAJ, ISBN: 978-1-108-42253-6.

[63] *Correlation-based feature selection for machine learning.* [Online]. Available: https://researchcommons.waikato.ac.nz/items/12a40834-bf51-4d87-89ef-d00c2740f0d8 (visited on 08/22/2024).

[64] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, no. null, pp. 1157–1182, Mar. 2003, ISSN: 1532-4435.

[65] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural network design.* USA: PWS Publishing Co., Feb. 1997, ISBN: 978-0-534-94332-5.

[66] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. DOI: 10.48550/arXiv.1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980 (visited on 08/22/2024).

[67] P. C. Tetlock, "Giving Content to Investor Sentiment: The Role of Media in the Stock Market," en, *The Journal of Finance*, vol. 62, no. 3, pp. 1139–1168, 2007, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2007.01232.x, ISSN: 1540-6261. DOI: 10.1111/j.1540-6261.2007.01232.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.2007.01232.x (visited on 08/22/2024).

[68] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, Mar. 2011, ISSN: 1877-7503. DOI: 10.1016/j.jocs.2010.12.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187775031100007X (visited on 08/22/2024).

[69] J. D. Piotroski, "Value Investing: The Use of Historical Financial Statement Information to Separate Winners from Losers," *Journal of Accounting Research*, vol. 38, pp. 1–41, 2000, Publisher: [Accounting Research Center, Booth School of Business, University of Chicago, Wiley], ISSN: 0021-8456. DOI: 10.2307/2672906. [Online]. Available: https://www.jstor.org/stable/2672906 (visited on 08/22/2024).

[70] E. F. Fama and K. R. French, "The Cross-Section of Expected Stock Returns," en, *The Journal of Finance*, vol. 47, no. 2, pp. 427–465, 1992,

_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1992.tb04398.x, ISSN: 1540-6261. DOI: 10.1111/j.1540-6261.1992.tb04398.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1992.tb04398.x (visited on 08/22/2024).

Appendix 1

1. Model Implementation

The "Model Implementation" Section below gives a detailed overview of the mathematical equations and structural diagrams available to each predictive model used in our study. Some of these algorithms are time-series related, such as classical statistical models (ARIMA, ARIMAX, SARIMA, and SARIMAX) for modeling time series data. Meanwhile, linear regression is used to analyze relationships between variables. In addition, we know about machine learning algorithms, including — Random Forest, SVR, KNN XGBoost, and Gradient Boosting with equations and decision mechanisms. In the deep learning section, we explain CNNs, RNNs, and LSTMs (focusing on architecture diagrams of these models) to show how they learn from the data.

1.1 ARIMA

ARIMA (Autoregressive Integrated Moving Average) is a powerful and flexible model for time series forecasting. Autoregression (AR), Integration (I), and Moving Average(MA) are three essential components for extracting different patterns within the time series data, such as trends, seasonality, etc. This section has described the ARIMA model, which provides us with an insightful understanding of its components, mathematical representation, and steps for implementation.

Components of ARIMA

1. **Autoregressive (AR) Component**: This component captures how an observation relates to a number of previous observations. It employs the relationship dependent upon an observation and several preceding values.

    - Mathematical formulation:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \qquad (1)$$

where:

- $\boldsymbol{X_t}$ is the value at time $\boldsymbol{t}$,
- $\boldsymbol{c}$ is a constant,
- $\boldsymbol{\phi_i}$ are the coefficients of the lagged terms,

- $\boldsymbol{p}$ is the number of lagged observations included in the model,
- $\boldsymbol{\epsilon_t}$ is the error term [1].

2. **Integrated (I) Component**: This component involves taking the differences of the observations to stationary a time series, which means that the statistical properties like mean and variance are constant over time.
    - Mathematical formulation for first differencing:

$$X_t' = X_t - X_{t-1} \quad (2)$$

where $\boldsymbol{X_t'}$ is the differenced series [1].

3. **Moving Average (MA) Component**: This component models the relationship between observations and lagged observations in relation to residual errors from a moving average model.
    - Mathematical formulation:

$$X_t = c + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \quad (3)$$

where:

- $\boldsymbol{\theta_j}$ are the coefficients of the lagged error terms,
- $\boldsymbol{q}$ is the order of the moving average [1].

Mathematical Formulation of ARIMA

The ARIMA model is typically denoted as ARIMA(p, d, q), where:

- $\boldsymbol{p}$ is the order of the autoregressive component,
- $\boldsymbol{d}$ is the degree of differencing required to make the series stationary,
- $\boldsymbol{q}$ is the order of the moving average component.

The general form of the ARIMA model is:

$$\phi_p(B)(1-B)^d X_t = \theta_q(B)\epsilon_t \quad (4)$$

where:

- $\boldsymbol{B}$ is the backshift operator,

- $\boldsymbol{\phi_p(B)}$ is the autoregressive polynomial of order $\boldsymbol{p}$,
- $\boldsymbol{(1-B)^d}$ represents differencing $\boldsymbol{d}$ times to achieve stationarity,
- $\boldsymbol{\theta_q(B)}$ is the moving average polynomial of order $\boldsymbol{q}$ [1].

Steps for Implementing ARIMA

1. **Data Preparation**:
    - **Load the Data**: Import the time series data and ensure it is in a suitable format for analysis.
    - **Stationarity Check**: Check stationarity using statistical tests like the Augmented Dickey-Fuller (ADF) test. If the series is non-stationary, then differencing is applied [2].
2. **Model Identification**:
    - **Determine $\boldsymbol{p}$ and $\boldsymbol{q}$**: Use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots to identify appropriate values for $\boldsymbol{p}$ and $\boldsymbol{q}$ [3].
3. **Parameter Estimation**:
    - **Fit the Model**: Use maximum likelihood estimation or least squares estimation to fit the ARIMA model to the data [4].
4. **Model Diagnostics**:
    - **Residual Analysis**: Analyse the residuals. Check white noise residuals that are typically distributed with mean 0 and constant variance [5].
5. **Forecasting**:
    - **Generate Forecasts**: Making forecasts using the fitted ARIMA model Calculate the performance of models using some evaluation metrics such as Mean Squared Error (MSE), and also use Mean Absolute error (MAE)[6].

1.2 ARIMAX

The ARIMAX (Autoregressive Integrated Moving Average with Exogenous Variables) model extends the ARIMA framework by allowing external variable(s) that may impact the primary target series. This section

covers each part of the ARIMAX model in detail, describing its constraints mathematically and procedure conceptualisation.

Components of ARIMAX

Autoregressive (AR) Component: This component models the relationship between an observation and several lagged observations. This would generally exploit the relationship between an observation and various lagged updates in prior values.

- Mathematical formulation:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \quad (5)$$

where:

- $\boldsymbol{X_t}$ is the value at time $\boldsymbol{t}$,
- $\boldsymbol{c}$ is a constant,
- $\boldsymbol{\phi_i}$ are the coefficients of the lagged terms,
- $\boldsymbol{p}$ is the number of lagged observations included in the model,
- $\boldsymbol{\epsilon_t}$ is the error term [7].

1. **Integrated (I) Component**: This component involves differencing the observations to make the time series stationary, which means that the statistical properties of the series are constant over time.
   - Mathematical formulation for first differencing:

$$X_t' = X_t - X_{t-1} \quad (6)$$

where $\boldsymbol{X_t'}$ is the differenced series [8].

2. **Moving Average (MA) Component**: This component models the relationship between an observation and a residual error from a moving average model applied to lagged observations.
   - Mathematical formulation:

$$X_t = c + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \quad (7)$$

where:

- $\boldsymbol{\theta_j}$ are the coefficients of the lagged error terms,

- $\boldsymbol{q}$ is the order of the moving average [9].

3. **Exogenous Variables (X) Component**: This component incorporates external variables that are expected to influence the dependent variable.
    - Mathematical formulation:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{k=0}^{m} \beta_k Z_{t-k} + \varepsilon_t \qquad (8)$$

where:

- $\boldsymbol{Z_{t-k}}$ represents the exogenous variables,
- $\boldsymbol{\beta_k}$ are the coefficients of the exogenous variables,
- $\boldsymbol{m}$ is the number of lagged exogenous observations included in the model [10].

Mathematical Formulation of ARIMAX

The ARIMAX model is typically denoted as ARIMAX($\boldsymbol{p, d, q}$), where:

- $\boldsymbol{p}$ is the order of the autoregressive component,
- $\boldsymbol{d}$ is the degree of differencing required to make the series stationary,
- $\boldsymbol{q}$ is the order of the moving average component,
- $\boldsymbol{m}$ is the number of exogenous variables.

The general form of the ARIMAX model is:

$$\phi_p(B)(1-B)^d X_t = \theta_q(B)\epsilon_t + \sum_{k=0}^{m} \beta_k Z_{t-k} \qquad (9)$$

where:

- $\boldsymbol{B}$ is the backshift operator,
- $\boldsymbol{\phi_p(B)}$ is the autoregressive polynomial of order $\boldsymbol{p}$,
- $\boldsymbol{(1-B)^d}$ represents differencing $\boldsymbol{d}$ times to achieve stationarity,
- $\boldsymbol{\theta_q(B)}$ is the moving average polynomial of order $\boldsymbol{q}$,
- $\boldsymbol{Z_{t-k}}$ represents the exogenous variables [11].

Steps for Implementing ARIMAX

1. **Data Preparation**:

- **Load the Data**: Import the time series data and exogenous variables, ensuring they are in a suitable format for analysis.

- **Stationarity Check**: Perform statistical checks for stationarity (for example, Augmented Dickey-Fuller test). Differencing if the series is non-stationary. [2]

2. **Model Identification**:

   - **Determine p, q, and m**: Use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots to select p, q-order terms for appropriate autoregression model order of ARIMA along with selection of exogenous variables [3].
3. **Parameter Estimation**:
   - **Fit the Model**: This ARIMAX model, including exogenous variables, was fitted to the data using maximum likelihood estimation or least squares estimation [4].

4. **Model Diagnostics**:

   - **Residual Analysis**: Examine the residuals to match white noise, which shows that it follows a normal distribution with an average of zero and constant variance [5].

5. **Forecasting**:

   - **Generate Forecasts**: Assess the model performance in terms of Mean Squared Error (MSE) and Mean Absolute Error [6].

1.3 SARIMA

The Seasonal Autoregressive Integrated Moving Average (SARIMA) model is an ARIMA extension specifically dealing with seasonal data patterns. SARIMA Combines seasonal and non-seasonal components to represent short-term fluctuations (non-seasonal) & long-term cyclic dynamics(Seasonal). The SARIMA model (step-by-step)This section describes the elements of the SARIMA model, its mathematical formulation, and the steps involved in its implementation [3].

Components of SARIMA

1. Autoregressive (AR) Component

It captures the dependence between any observation and several lagged observations. It exploits the dependence of an observation on multiple past observations.

Mathematical formulation:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \epsilon_t \quad (10)$$

where:

- $\boldsymbol{Y_t}$ is the value at time $\boldsymbol{t}$,
- $\boldsymbol{c}$ is a constant,
- $\boldsymbol{\phi_i}$ are the coefficients of the lagged terms,
- $\boldsymbol{p}$ is the number of lagged observations included in the model,
- $\boldsymbol{\epsilon_t}$ is the error term [3].

2. Integrated (I) Component

The difference part of this component involves differencing the observations (subtracting an observation from a previous time step) to make the time series stationary or use well-behaved statistical properties.

Mathematical formulation for first differencing:

$$Y_t' = Y_t - Y_{t-1} \quad (11)$$

Where $\boldsymbol{Y_t'}$ is the differenced series [3].

3 Moving Average (MA) Component

This part captures the association between an observation and residual error of moving average model applied at a lag in observed series.

Mathematical formulation:

$$Y_t = c + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \quad (12)$$

where:

- $\boldsymbol{\theta_j}$ are the coefficients of the lagged error terms,
- $\boldsymbol{q}$ is the order of the moving average [3].

4. Seasonal Components (SAR, SMA)

A series of transformations are available, including elements that model seasonal effects (including autoregressive terms and moving average elements that capture seasonality), which are handled via differencing.

Mathematical formulation for seasonal autoregressive (SAR) terms:

$$(1 - \phi_1 L^s - \phi_2 L^{2s} - \cdots - \phi_P L^{P_s}) \quad (13)$$

where $\boldsymbol{\phi_P}$ are the seasonal AR coefficients and $\boldsymbol{s}$ is the seasonal period [6].

Mathematical formulation for seasonal moving average (SMA) terms:

$$\left(1 - \Theta_1 L^s - \Theta_2 L^{2s} - \cdots - \Theta_Q L^{Qs}\right) \quad (14)$$

where$\boldsymbol{\Theta_Q}$ are the seasonal MA coefficients and $\boldsymbol{s}$ is the seasonal period [6].

Seasonal differencing:

$$(1 - L^s)^D Y_t \quad (15)$$

where $\boldsymbol{D}$ is the order of seasonal differencing [6].

Mathematical Formulation of SARIMA

The SARIMA model is typically denoted as SARIMA(p,d,q)(P,D,Q)s , where:

- $\boldsymbol{p}$ is the order of the non-seasonal autoregressive component,
- $\boldsymbol{d}$ is the degree of non-seasonal differencing,
- $\boldsymbol{q}$ is the order of the non-seasonal moving average component,
- $\boldsymbol{P}$ is the order of the seasonal autoregressive component,
- $\boldsymbol{D}$ is the degree of seasonal differencing,
- $\boldsymbol{Q}$ is the order of the seasonal moving average component,
- $\boldsymbol{s}$ is the length of the seasonal cycle.

The general form of the SARIMA model is:

$$\phi_p(B)\Phi_P(B^s)(1 - B)^d(1 - B^s)^D Y_t = \theta_q(B)\Theta_Q(B^s)\epsilon_t \quad (16)$$

where:

- $\boldsymbol{B}$ is the backshift operator,
- $\boldsymbol{\phi_p(B)}$ is the non-seasonal autoregressive polynomial of order $\boldsymbol{p}$,
- $\boldsymbol{\Phi_P(B^s)}$ is the seasonal autoregressive polynomial of order $\boldsymbol{P}$,
- $\boldsymbol{(1-B)^d}$ represents non-seasonal differencing,
- $\boldsymbol{(1-B^s)^D}$ represents seasonal differencing,
- $\boldsymbol{\theta_q(B)}$ is the non-seasonal moving average polynomial of order $\boldsymbol{q}$,
- $\boldsymbol{\Theta_Q(B^s)}$ is the seasonal moving average polynomial of order $\boldsymbol{Q}$ [12].

Steps for Implementing SARIMA

1. Data Preparation

- **Load the Data:** Import the time series data and ensure it is in a suitable format for analysis.
- **Stationarity Check:** Apply statistical tests (e.g. Augmented Dickey-Fuller [ADF] test) to determine whether a time series is stationary/ non-stationary. Apply differencing if the series is not stationary [2]

2. Model Identification

- **Determine p, d, q, P, D, Q, and s:** analyse plots of the Autocorrelation Function (ACF) and Partial_autocorrelation function for parameters p, *d, q, P, D, Q,* and *s* [3].

3. Parameter Estimation

- **Fit the Model:** Fit the SARIMA model to the data using maximum likelihood estimation or least squares estimators [13].

4. Model Diagnostics

- **Residual Analysis:** Study the residuals should be white noise or normally distributed with a mean of zero and uniform variance [14].

5. Forecasting

- **Generate Forecasts:** Use the fitted SARIMA model to make forecasts. Calculate model performance using metrics like Mean Squared Error (MSE), Mean Absolute Error(MAE) [6].

1.4 SARIMAX

The Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors (SARIMAX) is also an extension of the ARIMA model, including exogenous variables, i.e., factors from outside that affect time series behavior. In this section, we will describe in detail the SARIMAX model, its components and mathematical formulation required to implement our steps [3].

Components of SARIMAX

1. Autoregressive (AR) Component

This captures the autoregression component, whereby an observation is related to a number of lagged observations. This one takes an observation and combines it with more than a single past values.

Mathematical formulation:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \epsilon_t \qquad (17)$$

where:

- $\boldsymbol{Y_t}$ is the value at time $\boldsymbol{t}$,
- $\boldsymbol{c}$ is a constant,
- $\boldsymbol{\phi_i}$ are the coefficients of the lagged terms,
- $\boldsymbol{p}$ is the number of lagged observations included in the model,
- $\boldsymbol{\epsilon_t}$ is the error term [3].

2. Integrated (I) Component

This component differencing the observations makes the time series stationary, meaning that the statistical properties of a process generating a time series do not change over time.

Mathematical formulation for first differencing:

$$Y_t' = Y_t - Y_{t-1} \qquad (18)$$

Where $\boldsymbol{Y_t'}$ is the differenced series [3].

3.0 Moving Average (MA) Component

This component represents how an observation relates to a residual error in the predictive model, described as the moving average of observations at previous time steps.

Mathematical formulation:

$$Y_t = c + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \quad (19)$$

where:

- $\boldsymbol{\theta_j}$ are the coefficients of the lagged error terms,
- $\boldsymbol{q}$ is the order of the moving average [3].

4. Seasonal Components (SAR, SMA)

These seasonal components capture periodic variations in the time series (i.e., seasonality) through autoregressive terms, moving average terms and differences at particular lags.

Mathematical formulation for seasonal autoregressive (SAR) terms:

$$(1 - \phi_1 L^s - \phi_2 L^{2s} - \cdots - \phi_P L^{P_s}) \quad (20)$$

where $\boldsymbol{\phi_P}$ are the seasonal AR coefficients and $\boldsymbol{s}$ is the seasonal period [6].

Mathematical formulation for seasonal moving average (SMA) terms:

$$\left(1 - \Theta_1 L^s - \Theta_2 L^{2s} - \cdots - \Theta_Q L^{Qs}\right) \quad (21)$$

where$\boldsymbol{\Theta_Q}$ are the seasonal MA coefficients and $\boldsymbol{s}$ is the seasonal period [6].

Seasonal differencing:

$$(1 - L^s)^D Y_t \quad (22)$$

where $\boldsymbol{D}$ is the order of seasonal differencing [6].

5. Exogenous Variables (X) Component

Exogenous Variables which allow the model to take into this account external factors that might influence your time series.

Mathematical formulation with exogenous variables:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{k=0}^{K} \beta_k X_{t-k} + \epsilon_t \quad (23)$$

where:

- $\boldsymbol{\beta_k}$ are the coefficients of the exogenous variables $\boldsymbol{X_{t-k}}$,
- $\boldsymbol{K}$ is the number of exogenous variables [12].

Mathematical Formulation of SARIMAX

The SARIMAX model is typically denoted as SARIMAX(p, d, q)(P,D,Q)s, where:

- $\boldsymbol{p}$ is the order of the non-seasonal autoregressive component,
- $\boldsymbol{d}$ is the degree of non-seasonal differencing,
- $\boldsymbol{q}$ is the order of the non-seasonal moving average component,
- $\boldsymbol{P}$ is the order of the seasonal autoregressive component,
- $\boldsymbol{D}$ is the degree of seasonal differencing,
- $\boldsymbol{Q}$ is the order of the seasonal moving average component,
- $\boldsymbol{s}$ is the length of the seasonal cycle,
- $\boldsymbol{X}$ represents the exogenous variables.

The general form of the SARIMAX model is:

$$\phi_p(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D Y_t = \theta_q(B)\Theta_Q(B^s)\epsilon_t + \sum_{k=0}^{K}\beta_k X_{t-k} \quad (24)$$

where:

- $\boldsymbol{B}$ is the backshift operator,
- $\boldsymbol{\phi_p(B)}$ is the non-seasonal autoregressive polynomial of order $\boldsymbol{p}$,
- $\boldsymbol{\Phi_P(B^s)}$ is the seasonal autoregressive polynomial of order $\boldsymbol{P}$,
- $\boldsymbol{(1-B)^d}$ represents non-seasonal differencing,
- $\boldsymbol{(1-B^s)^D}$ represents seasonal differencing,
- $\boldsymbol{\theta_q(B)}$ is the non-seasonal moving average polynomial of order $\boldsymbol{q}$,
- $\boldsymbol{\Theta_Q(B^s)}$ is the seasonal moving average polynomial of order $\boldsymbol{Q}$,
- $\boldsymbol{\beta_k}$ are the coefficients of the exogenous variables $\boldsymbol{X_{t-k}}$ [15].

Steps for Implementing SARIMAX

1. Data Preparation

- **Load the Data:** Import the time series data and ensure it is in a suitable format for analysis.
- **Stationarity Check:** Test for stationarity by performing statistical tests such as the Augmented Dickey-Fuller (ADF) test. Take first difference is that the series is non-stationary [2].

2. Model Identification

- **Determine p, d, q, P, D, Q, and s:** using Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to identify appropriate values for ***p, d, q, P, D, Q,*** and ***s*** [3].

3. Exogenous Variables Identification

- **Select Relevant Exogenous Variables:** Identify and preprocess exogenous variables that might influence the time series [13].

4. Parameter Estimation

- **Fit the Model:** Fitting the SARIMAX Model to Data using Maximum Likelihood Estimation or Least Squares estimation for non-stationary data [14].

5. Model Diagnostics

- **Residual Analysis:** Check the residuals to verify that is random i.e. white-noise (specifically normal distributed with a centered mean and equal variance) [14].

6. Forecasting

- **Generate Forecasts:** Make Predictions based on the fitted SARIMAX model. Measure how well your model is doing with Mean Squared Error (MSE) and/or Mean Absolute Error (MAE) [6].

1.5 Linear Regression

Linear Regression is a basic statistical method that builds the relationship between the dependent variable and one or more independent variables. The feature is especially useful when modelling stock forecasting

problems. It can capture the linear relationships between stock prices and different financial attributes, including historical trading volumes or prices such as close and high. Linear Regression is one of the simplest models, yet it has remained a commonly used tool in financial analysis because it's easy to implement and well-defined [3].

Components of Linear Regression

1. Assumption of Linearity: Linear Regression assumes the linear relationship between independent (predictors) and dependent variables (target). Although this simplification can be helpful in providing more interpretable results, it also restricts the model from accurately capturing potentially non-linear relationships that are not uncommon within financial markets [16]. This assumption is important because without it the model predictions are no longer guaranteed to be both reliable and interpretable.

2. Independent Variables (Features): The independent variables used for stock forecasting can be the full inventory of financial indicators (e.g., historical stock prices, trading volumes) or just moving averages on different ranges of data and other relevant numbers with a high impact in the finance world, which may directly influence share price fluctuations. Each feature is chosen because it appears that it can meaningfully contribute to predicting the variation in, usually, future stock price or return[17].

3. Dependent Variable (Target): The dependent variable in Linear regression is the thing that we want the model to predict. Normally, this is the stock closing price or return over that particular period in the case of Stock Market forecasting. The model should learn to predict this value as well as possible according to the features selected [18].

Mathematical Formulation of Linear Regression

The Linear Regression model is mathematically expressed as:

$$Y = \beta_0 + \sum_{i=1}^{n} \beta_i X_i + \epsilon \qquad (26)$$

Where:

- $\boldsymbol{Y}$ is the dependent variable.
- $\boldsymbol{\beta_0}$ is the intercept.
- $\boldsymbol{\beta_i}$ are the coefficients of the independent variables.
- $\boldsymbol{X_i}$ are the independent variables.

- ϵ is the error term [19].

The intercept (β_0) represents the expected value of Y when all X_i are zero, and the coefficients (β_i) represent the expected change in Y for a one-unit change in X_i, assuming all other variables remain constant. The error term ϵ accounts for the variability in Y that cannot be explained by the linear relationship with X_i [20].

Steps for Implementing Linear Regression

1. Data Preparation: When it comes to the implementation of Linear Regression, you take the initial step towards preparing the data i.e., loading the dataset, handling missing values, and performing feature engineering. Stock forecasting requires a lot of feature engineering, which is the process where you create new useful financial indicators to enhance your model and increase its predictive power [21]. The data is divided into Training and Test sets, and a partial Fit Is Used to train a model based on the Train Data. The trained models are then evaluated based on the predicted "Y" attribute from the test Data.

2. Model Training: A Linear Regression model is trained by calculating the coefficients β, on selected training split and fitted to it e.g using Ordinary Least Squares (OLS). Ordinary Least Squares (OLS) method of estimation tries to minimize the sum of squared differences between observed values and predicted one from particular model [22]. This process of fitting the curve to the data helps in capturing linearity well between independent and dependent variables.

3. Model Evaluation: After the model has finished training, The model is tested on performance measures like R^2, MSE, and MAPE. R^2 is the percentage of variance in a dependent variable that can be explained by independent variables, while MSE and MAPE test how well you were forecasting [23]. In addition, Residual analysis is performed to verify that the residuals should follow normality with constant variance, which would not have any pattern between predicted and residual data for the best model fit [24].

4. Model Diagnostics: After examination, it is critical to run diagnostics to test the assumptions of these models. Specifics such as linearity, independence of errors, homoscedasticity (constant variance of residuals), and normal assumptions for the error terms should be checked. Not meeting these assumptions means that the model's predictions are spurious, and linear regression is not appropriate for data [25].

5. Forecasting: After obtaining the model with good evaluation metrics, your job is forecasting a test set or new data. The predictions are scored so

the model can be evaluated on performance with unseen data. The model can be further tuned by selecting features or using regularization to prevent overfitting [26].

6. Model Refinement: This step optimises features by enhancing the model, which might be done through LASSO Ridge regression. Governance ways also are helpful, particularly in stopping overfitting, in particular for large datasets with many capabilities [27].

1.6 Random Forest

The Random Forest is a strong ensemble learning method aggregating multiple decision trees to form a stronger and broader model. Especially in stock price forecasting, random forests are very helpful because they can manage a pretty large dataset, catch some complex interactions between features, and let us know the importance of features. However, despite their robustness, Random Forests have limitations, including possible overfit and inability to capture time dependencies indispensable in financial forecasting [28].

Components of Random Forest

1. **Ensemble of Decision Trees:** Random Forest is generated by using many decision trees, each getting trained in the form of a different amount of data. Executing the ensemble helps in controlling both model variance and improving accuracy as a way of handling overfitting challenges that are inherent when using single decision trees [29]. The final prediction is made by combining the predictions from all trees, for regression tasks this can be averaging and for classification tasks it would take a majority vote.

2. **Bagging (Bootstrap Aggregating):** This is also a popular technique used heavily in random forests, called bagging, where the different datasets are created by bootstrapping — which means we take a lot of random samples with replacements from the original dataset as instructed. Random Forest is a group of decision trees in which each tree is trained on a different subset, and then it generalizes better to the unseen data. [30].

3. **Feature Randomness:** Apart from random sampling of data, Random Forests also add randomness by selecting a subset of features randomly for each decision tree. Feature bagging, as this process is termed, serves to decorate the trees and allows for many different facets of the data set through boosting, which was found

incredibly helpful when working with high-dimensional financial datasets [31].

Mathematical Formulation of Random Forest

The Random Forest algorithm can be described mathematically as follows:

- **Prediction for Regression:** For a given input $\boldsymbol{x}$, the prediction of a Random Forest $\hat{\boldsymbol{y}}$ is the average of the predictions from all individual decision trees:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \qquad (27)$$

  where $\boldsymbol{T}$ is the total number of trees, and $\boldsymbol{h_t(x)}$ is the prediction from the $\boldsymbol{t}$ decision tree.

- **Feature Importance:** The importance of a feature in random forest is usually calculated as the decrease in node impurity (Gini for classifications and variance form regressions) when nodes using that feature are compared to any other one averaged over all trees. A higher score in importance leads to a key significance about the feature when predicting stock prices [32].

Steps for Implementing Random Forest

1. **Data Preparation:** Similar to any other model, we need to prepare data before using Random Forest. In simple words, this means loading the dataset, handling missing values, and doing some feature engineering. For stock forecasting, we look at methods to create lagged variables, moving averages, and other types of technical indicators in the feature engineering step. The data set is then divided into train-test sets so that the model can be assessed on unseen data [33].

2. **Model Training:** The Random Forest model is a type of ensemble algorithm trained by constructing multiple decision trees at training time. It gives the class that has the majority of votes from constructed trees. The training process mostly works by setting proper parameters like the number of trees **T**, the maximum depth of the trees, and features to consider for splitting at each node, etc. Fitting the model on the training data by minimising prediction error [34].

3. **Model Evaluation:** The model may be evaluated on various metrics such as Mean Squared Error (MSE), R-squared or R^2, and Mean Absolute Error(MAE) after training. Performance metrics

evaluate how well the model predicted on the test set. We also analyse the importance of features to identify which variables have the greatest impact on the model and its predictions [35].

4. **Model Diagnostics:** Validation with the diagnostics helps ensure that you understand how this model makes its predictions and if it is still making valid predictions. Overfitting should be checked So to make sure the model generalise well techniques such as cross-validation and pruning of trees can be used [32].

5. **Forecasting:** The Random Forest model is validated and finally applied to new/test data for making predictions. You can see data on these forecasts and how well the model performed after a stock price actually happened to determine confidence. The model can also be improved by hyperparameter tuning or other feature selection. [36].

6. **Model Refinement:** Hyperparameter tuning using techniques like Grid Search or Random search to fine-tune the model. Moreover, ensemble methods like stacking with Random Forest using other models (like boosting or neural networks) may be considered for better predictive accuracy as well as resilience [37].

1.7 Support Vector Regression (SVR)

Support Vector Regression (SVR) is a powerful technique for regression and stock price prediction falls in the same domain of predictive modeling based on numerical data set features providing an SVR analogy. SVR is an extension of a similar algorithm Support Vector Machines (SVM) which was originally introduced for classification and has the capability to handle both linear and non-linear functions since it maps input features into higher dimensions using kernel methods [38]. This is a particularly powerful feature as financial data can be noisy and difficult to frame with linear regression [39].

Components of SVR

1. **Non-Linear Relationship Handling**: To model non-linear relationships, we use Kernel functions such as Radial Basis Function (RBF) or polynomial kernels which map the original data into some higher-dimensional space where linear regression is applicable. This allows SVR to capture complex patterns that

are observed in financial markets, which makes it a better candidate for making predictions [39].

2. **Support Vectors and Margin**: The model's decision function is essentially determined by a subset of the training data points known as support vectors (i.e., they are the closest to each other) lying on/within max distance from hyperplane or margin. The margin is the span of values in which SVR can fit data while staying within a predefined tolerance set by epsilon (ε), to avoid overfitting on noise through concentrating on relevant points only [38].

3. **Kernel Functions**: Kernel function plays a crucial role in SVR by stipulating the way input feature vectors are transformed into higher dimensions. There are kernels, which include linear kernel (for essentially linear data) and rbf kenel. Which captures the nonlinear variations. Aggressive. SVR can efficiently perform a non-linearly transformation without actually calculating the coordinates in another dimension by using kernel trick [40].

Mathematical Formulation of SVR

SVR aims to find a function $\boldsymbol{f(x)}$ that deviates from the actual observed outputs by a value no greater than epsilon (ε) for each training point, while ensuring that the model is as flat as possible. The SVR model is expressed as:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + b \qquad (28)$$

Where:

- $\boldsymbol{\alpha_i}$ and $\boldsymbol{\alpha_i^*}$ are Lagrange multipliers that define the weight of each support vector.
- $\boldsymbol{K(x_i, x)}$ is the kernel function.
- $\boldsymbol{b}$ is the bias term that adjusts the decision boundary [39].

Steps for Implementing SVR

1. **Data Preparation**: Implementing SVR, we must first load our data and then prepare the data by dealing with missing values or feature scaling. Feature scaling is very important in SVR; this is because, the model depends on distance metrics that are incorporated by kernel function and results into being sensitive to feature magnitudes [40].

2. **Model Training**: SVR is trained with **αi** and **b** coefficients that are optimized so as to minimise the prediction error against an epsilon (ε) defined margin. The training progresses by computing solutions of a convex optimisation problem, like quadractic programming, to guarantee the global minimum [39].

3. **Model Evaluation**: Once trained the model is evaluated with such metrics as: R-squared(R^2), Mean Squared Error(MSE) and Root mean squared error to determine how well your model performs or generalized. These statistics offer an idea about how well the model is predicting and accommodating with data [38].

4. **Model Diagnostics**: Diagnostics are performed to ensure that a model is not overfitting (with the help of diagnostics) and validate assumptions made by the model. This involves looking at the residuals, looking into the distribution of support vectors, and making sure that no single data point is playing an outsized (it would indicate overfitting) influence on predictions [39].

5. **Forecasting**:
   SVR can be used to predict future stock prices after the model is accepted. The predictions of the model, are then reviewed for accuracy and reliability. The model parameters can be further tuned for better performance in forecasting if required [40].

6. **Model Refinement**: The process of refinement requires altering parameters like the kernel, epsilon (ε), and including new financial indicators in order to enhance model accuracy when faced with rapidly changing market conditions [39].

1.8 K-Nearest Neighbors

K-Nearest Neighbors (KNN): K-nearest neighbors is a versatile machine learning algorithm used for classification as well as regression problems with stock price prediction among one of them. Regarding stock prediction, KNN does this by first taking 'k' most similar historical data points (neighbors) and then using them to predict stock price in the future.

This, coupled with the model's simplicity and ability to capture local data patterns have made it a popular choice in financial modelling [32].

Components of K-Nearest Neighbors

1. **Assumption of Similarity**: In contrast to an algorithm that assumes a particular function character between the X variables and our Y, KNN assumptions are based on the expectation that close regions in feature space correspond with similar outcomes. For stock forecasting, this means days with the same financial indicators (historical prices, volume etc) are more likely to end at similar closing values. This assumption means that k-nn (k-nearest neighbors) can model some truly complex, non-linear relationships that are common between securities in financial markets [35].

2. **Independent Variables (Features)**: In stock price prediction with KNN, the features used are usually financial indicators representing a certain important aspect of those companies, like historical prices or moving averages/technical ones. These features are chosen based on their relevance to the target variable, typically the closing price of stock. It means selecting features is very important as KNN follows garbage in and garbage out mechanism [41].

3. **Dependent Variable (Target)**: For in stock forecasting, the dependent variable wants to predict; a simple example would be Stock Price or Return. In simpler words, KNN ultimately predicts the target value of a given point based on an average of its k-nearest neighbors, which guarantees prediction to take into account those data points that are most similar to this particular instance and not all [42].

Mathematical Formulation of K-Nearest Neighbors

The KNN algorithm can be described mathematically as follows:

Given a query point, the KNN algorithm identifies the 'k' closest points in the training data based on a chosen distance metric (e.g., Euclidean distance). The prediction $\hat{\boldsymbol{y}}_{\boldsymbol{q}}$ for the query point is then calculated as the average of the target values of these nearest neighbors:

$$\hat{y}_q = \frac{1}{k}\sum_{i=1}^{k} y_i \qquad (29)$$

Where:

- $\boldsymbol{y_i}$ represents the target values of the 'k' nearest neighbors.

- ***k*** is the number of neighbors used for prediction [43].

The distance metric (e.g., Euclidean, Manhattan) plays a critical role in determining the 'closeness' of the neighbors and, consequently, the accuracy of the predictions.

Steps for Implementing K-Nearest Neighbors

1. **Data Preparation**: Before using KNN for modelling, the data needs to be cleaned and prepared- this is called Data Preprocessing. Feature Engineering includes creating technical indicators, lagged values, and moving averages so that appropriate input to the model is given in sense stock forecasting. Partition the data to test how well our model performs on new, unseen tweets [44].

2. **Model Training**: There is no conventional training phase, as in many other machine learning models for KNN. Rather, the training data is retained, and prediction takes place to compare query points with this training store. The topic modeling process is computationally expensive, especially for a large training dataset. It becomes so much important to make feature selection/dimensional reduction [45].

3. **Model Evaluation**: The evolution of the KNN model was performed by looking at several different metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared(R^2). These measures help determine how accurate the predictions are and to what degree these models can account for variations in stock prices. In addition, cross-validation can be used to create a graph showing the optimal number of neighbors 'k' that reduces error much better [46].

4. **Model Diagnostics:** Model diagnostics KNN is about tuning the ‘k’ value and selecting a distance metric. If the 'k' becomes too small, getting overfitting where this model is highly sensitive to noise in training data. Conversely, a value of 'k' that is too high will oversmooth the predictions and cause underfitting. Another major determinant of the performance is the choice of distance metric, with Euclidean distance being most frequently used [28].

5. **Forecasting**: After validating the model, we can use it to predict our test set or new data. For each of these forecasts, we evaluate the accuracy with which our model generalises to data not seen. If need be, changes will likely involve returning to the feature set or tuning the hyperparameters (e.g. k, distance metric) [47].

6. **Model Refinement**: Where we are trying to tune the feature set and number of neighbors. This in turn will lead to a better predictive model, and feature selection can involve techniques like RFE (Recursive Feature Elimination), Select k Best, etc. Further, software such as Principal Component Analysis (PCA), which can reduce feature dimensions of X and keep prediction accurate [48].

1.9 Gradient Boosting

Gradient Boosting is a machine-learning method for both regression and classification problems. For stock prediction, Gradient Boosting does very well correlating complicated but non-linear relationships between the movement of stock prices and changes in various financial indicators. This model utilizes the idea of ensembling many weak learners (usually decision trees) to form a strong predictive model. This iterative behavior of Gradient Boosting assists it in correcting the mistakes made by initial models and leads to an extremely accurate, robust forecasting model [49].

Components of Gradient Boosting

1. **Weak Learners (Base Models):** Gradient Boosting uses a decision tree as a base learner. The common assumption is that we are building each tree in the sequence to correct the errors of all preceding trees. The weak learner is the individual tree, as each one alone does not have much explanatory power. But when aggregated as an ensemble, these weak learners make up a strong model that can capture complex patterns in the stock market data [36].

2. **Sequential Learning:** This model is developed in a sequential manner such that the new tree tries to minimize the residual errors from preceding trees. The methodology is designed so that the model will iteratively learn and correct its prediction with every iteration, therefore suitable for forecasting tasks where the pattern is complex to catch by simpler models [50].

3. **Learning Rate:** As I mentioned earlier in Gradient Boosting, the important model parameter is the learning rate, which constrains the contribution of each tree to the final ensemble. Essentially, a lower learning rate makes the model more robust, with every tree's contribution being minimised but it also needs the addition of many trees to achieve the same level of performance. Optimisation of the learning rate must be balanced with the

number of trees to obtain optimal performance while avoiding overfitting [32].

4. **Loss Function:** It basically minimizes the loss for it learns to converge it, like Mean Squared Error (MSE) in regression tasks, and uses gradient-descent as an increment or decrement of this functions. The type of loss function to use is defined by the nature of your prediction task (regression, classification) for which there are different loss functions [51].

Mathematical Formulation of Gradient Boosting

The Gradient Boosting model is mathematically formulated as follows:

$$F_m(x) = F_{m-1}(x) + \gamma h_m(x) \tag{30}$$

Where:

- $\boldsymbol{F_m(x)}$ is the updated model after adding the **m-th** tree.
- $\boldsymbol{F_{m-1}}$ is the ensemble model after $\boldsymbol{m-1}$ trees.
- $\boldsymbol{\gamma}$ is the learning rate.
- $\boldsymbol{h_m(x)}$ is the new tree fitted to the residuals of the previous model.

The model starts with an initial prediction, and each subsequent model is built to correct the residual errors of the previous model [52].

1.10 XGBoost

Extreme planting increases the efficiency and scale of these gradient-boosting algorithms specifically for structured or tabular data. Since it has the ability to deal with complex data structures, manage non-linear relationships, and process large-scale datasets efficiently, XGboost has become very popular in stock price prediction. This makes XGBoost popular for financial forecasting — its design reduces overfitting and is geared toward high-performance [49],[53].

Components of XGBoost

1. **Gradient Boosting Framework**: Gradient boosting is an approach where the models are added sequentially to correct errors made by previous models, and XGBoost (based on gradient boosting) does this great. Light GBM works to train the residuals (errors) of previous models, so here, each new model is trained, and its work is combined with others as the final prediction. It is possible to choose the loss function that you want to minimise;

this is optimised via an efficient approximate second-order gradient [49].

2. **Tree-Based Learning:** XGBoost runs using decision trees as base learners. LGBM constructs these trees level-wise instead of leaf-wise, as used by other gradient-boosting algorithms. XGBoost has taken the level-wise approach instead of weight-based shrinking, which helps in smoother growth by lowering over-fitting. Trees are constructed to optimize a loss function that incorporates an error term and regularisation for complexity [49], [53].

3. **Regularization**: An embodied difference between XGBoost and other algorithms is that it is a regularized algorithm. In general, XGBoost has a regularization term such as L1 (Lasso) and L2 (Ridge). These regularization terms decrease the overall complexity of a model, assuming that simple models are more appropriate for any given data set and hence they help to prevent over-fitting in environments where there is likely stock market noise. [53].

4. **Handling Missing Values**: XGBoost also has the capability to handle missing data natively. In training mode this algorithm will automatically learn which way to go in the tree for missing values, optimizing the split. This is especially helpful for financial datasets which frequently do have missing values [49].

5. **Feature Importance and Interpretability**: XGBoost has inbuilt methods to find out feature importance measures between different features averaged over all the trees. This can help portray the importance of each feature, and be used in visualizing important factors that best contribute to predicting stock prices hence encouraging interpretability. This is helpful for bias measures, interpretation of feature importances or explanations (and therefore compliance), and when the understanding how the model actually makes a decisions [49], [53].

Mathematical Formulation of XGBoost

The objective function of XGBoost is designed to optimize both the predictive accuracy of the model and its complexity. It is expressed as:

$$Y = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (31)$$

Where:

- $\boldsymbol{L(y_i, \hat{y}_i)}$ is the loss function (such as mean squared error) that measures how well the model predicts the target variable.

- $\boldsymbol{\Omega(f_k)}$ is the regularization term that penalizes the complexity of the model, which helps prevent overfitting.

- $\boldsymbol{Y}$ is objective.

The regularization term $\boldsymbol{\Omega(f_k)}$ is defined as:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \qquad (32)$$

Where:

- $\boldsymbol{T}$ is the number of leaves in the tree.

- $\boldsymbol{\omega_j}$ is the weight of the **jth** leaf.

- $\boldsymbol{\gamma}$ and $\boldsymbol{\lambda}$ are hyperparameters that control the amount of regularization.

Steps for Implementing XGBoost

1. **Data Preparation**: Implementation starts with data preparation, in which one needs to clean the dataset, handle missing values, and do feature engineering. Feature engineering means doing some lag transformation and technical indicators and maybe taking the external macroeconomic factors into account at the data level. Important to have enough information on stock prices for patterns to become vivid [49], [53].

2. **Model Training**: The XGBoost algorithm trains by adding each decision tree sequentially such that when added, it will attempt to correct the errors of its predecisors. During the training process, we seek to optimize the objective function defined above by minimizing both loss and regularization terms. Model performance is improved by tuning hyperparameters like learning rate, max depth, and number of trees [49].

3. **Model Evaluation**: Post training, the performance of the model is compared using different metrics, i.e., Mean Squared Error(MSE), Mean Absolute Error (MAE), and R-squared(R^2 consist). Cross-validation: The MDN model is robustified using cross-validation methods and walk-forward validations. This is an important step to check the performance of a model on new unseen data [53].

4. **Model Tuning**: Hyperparameter Tuning Using these new hyperparameters for model building, Grid Search, or Randomized

Search is used to generate the ideal pair of parameters so that better and non-overfitting models can be made [49].

5. **Forecasting**: After training and tuning the model, it forecasts stock prices. Finally, these predictions are evaluated against a handful of known correct values to determine their accuracy. With this, the model may need retraining every period to adjust its predictions as some numbers change yearly market conditions [49].

6. **Model Refinement**: Refining the model requires ongoing monitoring and refinement. This could involve reconsidering the importance of features, inserting new ones, or changing values for regularization terms. The object is to hold onto the strong predictive abilities while limiting training set over-fitting of it [49], [53].

1.11 CNN

Convolutional Neural Networks (CNNs), which are a subset of deep learning models that were first introduced as an architecture for image recognition and have since been successfully applied to time series data such as stocks. CNNs are built to find local patterns in the input data, which Convolutional Layers efficiently do. Hence it excels when modelling for short-term dependencies like stock price movements as well [54], [55].

Components of CNN

1. **Convolutional Layers**:

   Convolutional layers are the main blocks of CNNs. At the lowest level of CNNs, you can think of them as a set of learnable filters (or kernel) that slide across input data to create feature maps. These filters learn specific patterns or features in the data like trends, and stock price behaviors over time. These layers help with detecting quite similar patterns that can predict future price movements when it comes to stock forecasting [56], [57].

2. **Pooling Layers**:

   Commonly, pooling layers follow convolutional layers to decrease the spatial dimension of feature maps by averaging or max-pooling. This process helps summarise the features

detected by convolutional layers and makes this model more robust to minor variations in input data. All the model proceedings, such as max pooling in stock forecasting, are essential to preserving valuable features from a particular feature map that can lead toward generalisation of the model [57], [58].

3. **Flattening and Dense Layers**:

   The feature maps are flattened into a one-dimensional vector after Convolutions and pooling layers before going through fully connected (dense) layers. The output of these layers is then used as features in the forecasting task, stacking on top to predict future stock prices from input stocks. These dense layers of a CNN model are responsible for combining the features learned in the previous layer to make its prediction with better accuracy [54], [56].

Mathematical Formulation of CNN

A CNN processes the input data X through multiple layers as follows:

Convolution Operation:

$$Z^{(l)} = W^{(l)} * X^{(l-1)} + b^{(l)} \quad (33)$$

Where:

- $\boldsymbol{Z^{(l)}}$ is the output of the convolutional layer $\boldsymbol{l}$.
- $\boldsymbol{W^{(l)}}$ represents the filters or kernels.
- $\boldsymbol{X^{(l-1)}}$ is the input from the previous layer.
- $\boldsymbol{b^{(l)}}$ is the bias term [57].

Steps for Implementing CNN

1. Data Preparation:

   First, preparing the dataset, which involves normalising stock price data and transforming it into a format that could work for CNN model. This often involves restructuring the data into 3D arrays (samples, time steps, features) to fit these input requirements of convolutional layers [54], [58].

2. Model Training:

   To train the CNN, we pass our formatted data through a network that has convolutional layers that learn features specific to identifying stock price movements and dense layers that put these together as an output of future stock prices. The model parameters are optimized with backpropagation and gradient descent [55], [57].

3. Model Evaluation:

   Portions of the dataset can be used in training and others for testing to get metrics such as MSE, RMSE, R^2 Score that indicate how close predictions are to actual results on a separate test set upon which the model is trained. The evaluation process is made to validate that the model generalizes well on unseen data and does not overfitting [56], [58].

4. Model Diagnostics:

   Model diagnostics require verifying the performance of a model using approaches such as residual analysis, and visualizing learned filters to evaluate which representations were discovered by this network from the data. Step crucial to the enhancing model and predictive power of it [57], [58].

5. Forecasting:

   Once validated, the CNN model forecasted stock prices on unseen data. These predictions are then compared against the real prices to measure how well such a model performs in practice [56], [57].

6. Model Refinement:

   It involves tuning hyperparameters, exploring new CNN architectures (adding extra convolutional layers or modifying pooling strategy) and adding other factors such as trading volume/external economic indices to improve model accuracy [58], [59].

1.12 RNN

Recurrent Neural Networks (RNNs) are a class of artificial neural networks dedicated to working with sequential data among which, time series forecasting seems to be abundant such as Stock Prices. RNNs can remember past input which makes it well suited to learn the long-term patterns in time series data such as stock prices and predicting future values of a variable from historical inputs [60].

Components of Recurrent Neural Networks

1. **Sequential Data Processing:** RNNs are designed to work with sequences of data — they have an internal state that keeps track of previous time steps. The ability to process sensor and time-series data enables RNNs trained on stock price forecasting (given prices from the past, predict prices in the future) because knowing previous minute(s) of trading can help forecast new trades [61]. This ordering is the key distinction between RNNs and traditional feedforward neural networks, which essentially have no memory of previous inputs.

2. **Hidden States and Memory:** In each time step, RNNs is updated by a hidden state which take in the current input and previous variable. This way the network can remember information from previous time steps since some of it is retained when updates occur. But standard RNNs can face the problem of having a short term memory and not being able to learn long range dependencies (for example, vanishing gradients) due to which we have more advanced models like Long Short-Term Memory(LSTM) networks or Gated Recurrent Units (GRUs) [62].

3. **Non-linear Activation Functions:** Similarly RNNs makes use of non-linear activation functions such as Tanh or ReLU(Rectified Linear Unit) to introduce such complexity in the model. This non-linear feature helps RNN in capturing nonlinear relationships between inputs and outputs which are quite common in financial data. We use these activation functions to enable the network so as to approximate non-linear dynamisms present among stock prices [63].

4. **Backpropagation Through Time (BPTT):** Training RNNs involves a special type of backpropagation called Back propagation Through Time (BPTT). BPTT rolls out the RNN for k time steps and calculates gradients w.r.t every timestamp so that we can update its weights. It was computationally intensive but necessary for the model to learn temporal dependencies in data

[64]. It is important to tune BPTT parameters, such as the unrolling length of RNNs so that the gradients do not explode or vanish.

Mathematical Formulation of RNN

The RNN model can be mathematically described by the following equations:

1. **Hidden State Update:**

$$h_t = \sigma(W_h \cdot x_t + U_h \cdot h_{t-1} + b_h) \quad (34)$$

Where:

- $\boldsymbol{h_t}$ is the hidden state at time ***t***.
- $\boldsymbol{x_t}$ is the input at time ***t***.
- $\boldsymbol{W_h}$ and $\boldsymbol{U_h}$are weight matrices.
- $\boldsymbol{b_h}$ is the bias term.
- $\boldsymbol{\sigma}$ is the activation function, typically Tanh or ReLU [65].

2. **Output Calculation:**

$$y_t = \sigma\big(W_y \cdot h_t + b_y\big) \quad (35)$$

Where:

- $\boldsymbol{y_t}$ is the output at time ***t***.
- $\boldsymbol{W_y}$ is the weight matrix for the output layer.
- $\boldsymbol{b_y}$ is the bias term [66].

Each of these equations characterize how the RNN reads an input sequence and returns a list of outputs, where each output depends on both current input as well as hidden state read from previous time step.

Steps for Implementing RNN

1. **Data Preparation:** The implementation of an RNN for stock prediction is to begin with data preparation. This requires loading the dataset, scaling (normalizing) features to an appropriate range

and shaping data into sequences that can be dealt with by RNNs. For example, if we wanted to predict stock price over the next day based on 60 days of historical data, then input sequences would be composed of those continuous 60 days only [67].

2. **Model Architecture:** The RNN model is created by stacking the same type of layers that its input sequences can process, where it remembers some information for a period to make accurate predictions. A deep RNN is the creation of stacking two or more layers just as done in a fully connected feed forward network, some times it may somewhat increases performance depending on the complexity of task. To avoid overfitting it is common to add dropout layers between RNN(Recurrent Neural Network) layers by randomly setting a fraction of the output units to zero during training [68].

3. **Model Training:** Now that the RNN model has been built, it is trained using Backpropagation Through Time (BPTT), an algorithm where back-propagation works by unfolding a given network in time and then applying Chain Rule recursively for all instances of cells so created to adjust weights with respect goal function typically Mean Squared Error (MSE) for regression tasks. The model is trained to modify its weights based on the historical changes and structures inherent in data, so that it can predict future stock prices [69].

4. **Model Evaluation:** Finally, the model is trained and evaluated with MSE, RMSE, MAE and R^2 (R-square) on our test set. Measurement of these metrics helps to understand the predictive power and generalization capability of your model. Specifically, the R^2 score measures how well the model is able to explain variation in stock prices [28].

5. **Forecasting:** With a trained model, forecasting is simply making predictions of what future stock prices could be using most recent sequences. In finance, the ability of a model to predict unseen data is paramount for it to be useful in practice [70].

6. **Model Refinement:** The model may be fine-tuned, more hyper parameters can be studied out or other RNNs like LSTMs of GRUs could have been considered in little bit detail with some add-on features to make prediction eligible. But we may need to update the model periodically as market conditions change, so that it continues to predict well [71].

1.13 LSTM

A Long Short-Term Memory (LSTM) is an artificial RNN architecture designed to manage sequence data, by capturing long-term dependencies and solving gradient vanishing problems. Since LSTM can learn behavioral patterns, it is very suitable for forecasting stock prices because they have time series which depends on past events. LSTM Networks are the flavor of choice for financial modeling due to their fantastic performance on time series data and a particularly relevance in cases where historical context matters immensely to foreseeing future predictions [72].

Components of LSTM

1. **Memory Cells:**

   At the basic level an LSTM networks are made up of cells, and those cells store information across time-steps. The memory cells are fitted with gates that control the information flow, allowing it to remember or forget as desired. The structure plays a crucial role in helping the LSTM models to capture and learn long-term dependencies within this data hence making it efficient for forecasting stock prices over quite some periods with high precision [66].

2. **Gates in LSTM:**

   The three major gates in LSTM networks are the input gate, forget gate and output gates. These gates regulate information to enter, exit or reside within the memory cell:
   - **Input Gate:** Determines which values from the input should be updated in the memory state.
   - **Forget Gate:** Decides which information should be discarded from the cell state.
   - **Output Gate:** Controls the output based on the cell state and the current input.

   LSTM networks are capable of learning complex patterns and relationships in sequential data as a result of such gate operation, which leads to LSTM being used widely in financial forecasting tasks [56].

3. **Sequential Data Handling:**

LSTM networks are different from traditional feedforward neural network as they have a hidden state to process sequences of data points that are ordered in time steps. This property is important when dealing with stock price forecasting as the historical order of prices heavily determines future prediction. As LSTM can handle data in sequence form, so it is much better than simply models when we need to deal with financial data [73].

Mathematical Formulation of LSTM

The LSTM model processes the input data through the following equations:

- **Forget Gate:**

$$f_t = \sigma\left(w_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (36)$$

Here, $\boldsymbol{f_t}$ is the forget gate's output, $\boldsymbol{w_f}$ represents the weights, $\boldsymbol{h_{t-1}}$ is the previous hidden state, $\boldsymbol{x_t}$ is the current input, and $\boldsymbol{b_f}$ is the bias term. The forget gate decides which information should be discarded from the memory cell [60].

- **Input Gate:**

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \quad (37)$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (38)$$

The input gate $\boldsymbol{i_t}$ controls which new information will be added to the memory, while $\boldsymbol{\widetilde{C}_t}$ is the candidate value for updating the cell state [62].

- **Cell State Update:**

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (39)$$

The cell state $\boldsymbol{C_t}$ is updated by combining the previous state and the candidate values, weighted by the forget and input gates respectively [74].

- **Output Gate:**

$$O_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o) \quad (40)$$

$$h_t = O_t * tanh(C_t) \quad (41)$$

The output gate $\boldsymbol{O_t}$ determines the next hidden state $\boldsymbol{h_t}$ based on the updated cell state, which in turn is used for the final prediction [60].

Steps for Implementing LSTM

1. **Data Preparation:**

   To implement an LSTM model, you preprocess the data which means normalisation of it, creating sequences and splitting your data into training dataset and test set. This stage in input format has a lot to do with how the model learns and interprets data [75].

2. **Model Architecture Design:**

   As LSTM model has a standard architecture that includes one or more LSTM layers and dense layer, with the final output going to all possible values for classification. Dropout layers could also be added to help reduce overfitting. Layers, units and other hyperparameters are then tuned to improve the performance of the model [76].

3. **Training the Model:**

   The model is trained using Backpropagation through time (BPTT) where the LSTM learns changing its weights so that errors between predicted output and actual values are reduced. Performance of the model is evaluated over a validation set to stop overfitting and enhance Generalization [77].

4. **Evaluation and Forecasting:**

   For testing the model, they are used metrics like Mean Squared Error (MSE), Root Mean Square Deviation(R^2) and R-squared. These are the metrics that give an indication of how well whether our model is predicting stock prices. Lastly this model serves as to predict future prices and is tested on actual stock data [78].

2. Evaluation Metrics

Mean Squared Error (MSE)

Mean Squared Error (MSE) is a common metric for assessing the performance of regression models. It measures the average of the squared differences between the actual and predicted values. The formula for MSE is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (42)$$

where $\boldsymbol{y_i}$ is the actual value**,** $\boldsymbol{\hat{y}_i}$ is the predicted value, and $\boldsymbol{n}$ is the number of data points. MSE penalizes larger errors more severely, making it sensitive to outliers. A lower MSE indicates a better model performance [79].

Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is the square root of the MSE. It provides an error metric in the same units as the original data, making it more interpretable in practical terms. The formula for RMSE is:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (43)$$

RMSE is also sensitive to large errors due to its squaring effect, but it gives a clear sense of how far the predictions deviate from the actual values on average [80].

Mean Absolute Error (MAE)

Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions, without considering their direction. The formula for MAE is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_1 - \hat{y}_i| \quad (44)$$

MAE is a linear score, meaning that all the individual differences are weighted equally in the average. It is less sensitive to outliers compared to MSE and RMSE [79].

Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) measures the average of the absolute percentage errors of predictions. The formula for MAPE is:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_1}\right| \quad (45)$$

MAPE is useful when comparing the predictive accuracy across different datasets, as it is scale-independent. However, it has limitations when actual values are close to zero, leading to inflated percentage errors [81].

Coefficient of Determination (R^2)

The Coefficient of Determination, denoted as $\boldsymbol{R^2}$, indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. The formula for $\boldsymbol{R^2}$ is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_j - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (46)$$

where $\boldsymbol{\bar{y}}$ is the mean of the observed data. An $\boldsymbol{R^2}$ value closer to **1** indicates that the model explains a large portion of the variance, while a value closer to **0** suggests poor explanatory power [82].

Bibliography:

[1] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples*. in Springer Texts in Statistics. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-52452-8.

[2] D. A. Dickey and W. A. Fuller, "Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root," *Econometrica*, vol. 49, no. 4, Art. no. 4, 1981, doi: 10.2307/1912517.

[3] G. M. Jenkins, "Time series analysis : forecasting and control," *No Title*, Accessed: Aug. 21, 2024. [Online]. Available: https://cir.nii.ac.jp/crid/1130000797820830336

[4] J. Durbin and S. J. Koopman, *Time Series Analysis by State Space Methods*. OUP Oxford, 2012.

[5] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*. in Springer Texts in Statistics. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-29854-2.

[6] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018. Accessed: Aug. 21, 2024. [Online]. Available: https://books.google.co.uk/books?hl=en&lr=&id=_bBhDwAAQBAJ&oi=fnd&pg=PA7&dq=2.+R.+J.+Hyndman+and+G.+Athanasopoulos,+%22Forecasting:+Principles+and+Practice,%22+2nd+ed.,+OTexts,+2018.+Available:+https://otexts.com/fpp3/arima.html.&ots=TjgZviTIHP&sig=yTXG2hxUodSFz_gD_vtcpGiFf-A

[7] A. Pankratz, *Forecasting with Dynamic Regression Models*. John Wiley & Sons, 2012.

[8] S. Suhartono, "Time Series Forecasting by using Seasonal Autoregressive Integrated Moving Average: Subset, Multiplicative or Additive Model," *J. Math. Stat.*, vol. 7, pp. 20–27, Jan. 2011, doi: 10.3844/jmssp.2011.20.27.

[9] J. Wang, X. Li, L. Jin, J. Li, Q. Sun, and H. Wang, "An air quality index prediction model based on CNN-ILSTM," *Sci. Rep.*, vol. 12, p. 8373, May 2022, doi: 10.1038/s41598-022-12355-6.

[10] H. Sun, Y. Yang, Y. Chen, X. Liu, and J. Wang, "Tourism demand forecasting of multi-attractions with spatiotemporal grid: a convolutional block attention module model," *Inf. Technol. Tour.*, pp. 1–29, Apr. 2023, doi: 10.1007/s40558-023-00247-y.

[11] Z.-Q. Li, H.-Q. Pan, Q. Liu, H. Song, and J.-M. Wang, “Comparing the performance of time series models with or without meteorological factors in predicting incident pulmonary tuberculosis in eastern China,” *Infect. Dis. Poverty*, vol. 9, no. 1, p. 151, Nov. 2020, doi: 10.1186/s40249-020-00771-7.
[12] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, “A state space framework for automatic forecasting using exponential smoothing methods,” *Int. J. Forecast.*, vol. 18, no. 3, Art. no. 3, Jul. 2002, doi: 10.1016/S0169-2070(01)00110-8.
[13] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 2020.
[14] G. M. LJUNG and G. E. P. BOX, “On a measure of lack of fit in time series models,” *Biometrika*, vol. 65, no. 2, Art. no. 2, Aug. 1978, doi: 10.1093/biomet/65.2.297.
[15] “Inventions | Free Full-Text | A Seasonal Autoregressive Integrated Moving Average with Exogenous Factors (SARIMAX) Forecasting Model-Based Time Series Approach.” Accessed: Aug. 24, 2024. [Online]. Available: https://www.mdpi.com/2411-5134/7/4/94
[16] J. Y. Campbell, A. W. Lo, A. C. MacKinlay, and R. F. Whitelaw, “THE ECONOMETRICS OF FINANCIAL MARKETS,” *Macroecon. Dyn.*, vol. 2, no. 4, Art. no. 4, Dec. 1998, doi: 10.1017/S1365100598009092.
[17] G. Atsalakis, “Surveying Stock Market Forecasting Techniques - Part I: Conventional Methods,” *J. Comput. Optim. Econ. Finance*, vol. 6, pp. 19–28, Jan. 2010.
[18] C. Brooks, *Introductory Econometrics for Finance*. Cambridge University Press, 2019.
[19] T. Hastie, J. Friedman, and R. Tibshirani, *The Elements of Statistical Learning*. in Springer Series in Statistics. New York, NY: Springer, 2001. doi: 10.1007/978-0-387-21606-5.
[20] C. Chatfield and H. Xing, *The Analysis of Time Series: An Introduction with R*, 7th ed. New York: Chapman and Hall/CRC, 2019. doi: 10.1201/9781351259446.
[21] J. Y. Campbell and R. J. Shiller, “The Dividend-Price Ratio and Expectations of Future Dividends and Discount Factors,” *Rev. Financ. Stud.*, vol. 1, no. 3, pp. 195–228, Jul. 1988, doi: 10.1093/rfs/1.3.195.
[22] J. H. Stock and M. W. Watson, “Introduction to econometrics (3rd updated edition),” *Age X3*, vol. 3, no. 0.22, 2015, Accessed: Aug. 24, 2024. [Online]. Available: https://swh.princeton.edu/~mwatson/Stock-Watson_3u/Students/RTC/Stock_Watson_3U_AnswersToReviewTheConcepts.pdf
[23] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. John Wiley & Sons, 2021.
[24] R. S. Tsay, *Analysis of Financial Time Series*. John Wiley & Sons, 2010.
[25] C. R. Nelson, *Applied Time Series Analysis for Managerial Forecasting*. San Francisco: Holden Day, 1973.
[26] P. Khuwaja, S. A. Khowaja, I. Khoso, and I. A. Lashari, “Prediction of stock movement using phase space reconstruction and extreme

learning machines," *J. Exp. Theor. Artif. Intell.*, Jan. 2020, Accessed: Aug. 24, 2024. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/0952813X.2019.1620870
[27] H. A. David and H. N. Nagaraja, *Order Statistics*, 1st ed. in Wiley Series in Probability and Statistics. Wiley, 2003. doi: 10.1002/0471722162.
[28] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, Art. no. 1, Oct. 2001, doi: 10.1023/A:1010933404324.
[29] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*, Berlin, Heidelberg: Springer, 2000, pp. 1–15. doi: 10.1007/3-540-45014-9_1.
[30] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, Art. no. 2, Aug. 1996, doi: 10.1007/BF00058655.
[31] A. Liaw and M. Wiener, "Classification and Regression by RandomForest," *Forest*, vol. 23, Nov. 2001.
[32] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. in Springer Series in Statistics. New York, NY: Springer, 2009. doi: 10.1007/978-0-387-84858-7.
[33] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd edition. Morgan Kaufmann, 2011.
[34] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2022.
[35] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. in Springer Texts in Statistics. New York, NY: Springer US, 2021. doi: 10.1007/978-1-0716-1418-1.
[36] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Ann. Stat.*, vol. 29, no. 5, Art. no. 5, 2001.
[37] C. C. Aggarwal, *Data Mining: The Textbook*. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-14142-8.
[38] Y. Guo, S. Han, C. Shen, Y. Li, X. Yin, and Y. Bai, "An Adaptive SVR for High-Frequency Stock Price Forecasting," *IEEE Access*, vol. 6, pp. 11397–11404, 2018, doi: 10.1109/ACCESS.2018.2806180.
[39] A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain, "Support vector regression with chaos-based firefly algorithm for stock market price forecasting," *Appl. Soft Comput.*, vol. 13, no. 2, Art. no. 2, Feb. 2013, doi: 10.1016/j.asoc.2012.09.024.
[40] S. Sricharan and V. Joshi, "Comparison of SVR Techniques for Stock Market Predictions," in *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, Jul. 2022, pp. 1–6. doi: 10.1109/ICDSIS55133.2022.9915990.
[41] "Financial time series forecasting using independent component analysis and support vector regression - ScienceDirect." Accessed: Aug. 24, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923609000323
[42] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock market index using fusion of machine learning techniques," *Expert*

Syst. Appl., vol. 42, no. 4, Art. no. 4, Mar. 2015, doi: 10.1016/j.eswa.2014.10.031.
[43] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, Art. no. 1, Jan. 1967, doi: 10.1109/TIT.1967.1053964.
[44] "Correlation-based feature selection for machine learning." Accessed: Aug. 23, 2024. [Online]. Available: https://researchcommons.waikato.ac.nz/items/12a40834-bf51-4d87-89ef-d00c2740f0d8
[45] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J Mach Learn Res*, vol. 3, no. null, Art. no. null, Mar. 2003.
[46] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural network design*. USA: PWS Publishing Co., 1997.
[47] "Principal component analysis - Abdi - 2010 - WIREs Computational Statistics - Wiley Online Library." Accessed: Aug. 23, 2024. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.101
[48] I. Jolliffe, "Principal Component Analysis," in *Encyclopedia of Statistics in Behavioral Science*, John Wiley & Sons, Ltd, 2005. doi: 10.1002/0470013192.bsa501.
[49] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[50] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Front. Neurorobotics*, vol. 7, Dec. 2013, doi: 10.3389/fnbot.2013.00021.
[51] M. J. Silva de Souza *et al.*, "Can artificial intelligence enhance the Bitcoin bonanza," *J. Finance Data Sci.*, vol. 5, no. 2, pp. 83–98, Jun. 2019, doi: 10.1016/j.jfds.2019.01.002.
[52] J. H. Friedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal.*, vol. 38, no. 4, Art. no. 4, Feb. 2002, doi: 10.1016/S0167-9473(01)00065-2.
[53] W. Zou *et al.*, "SmartDL: energy-aware decremental learning in a mobile-based federation for geo-spatial system," *Neural Comput. Appl.*, vol. 35, no. 5, pp. 3677–3696, Feb. 2023, doi: 10.1007/s00521-021-06378-9.
[54] J. Shen and M. O. Shafiq, "Short-term stock market price trend prediction using a comprehensive deep learning system," *J. Big Data*, vol. 7, no. 1, Art. no. 1, Aug. 2020, doi: 10.1186/s40537-020-00333-6.
[55] "Forecasting Stock Prices Using A Temporal CNN Model," Forecasting Stock Prices Using A Temporal CNN Model - QuantConnect.com. Accessed: Aug. 25, 2024. [Online]. Available: https://www.quantconnect.com/research/15263/forecasting-stock-prices-using-a-temporal-cnn-model/
[56] T. B. Shahi, A. Shrestha, A. Neupane, and W. Guo, "Stock Price Forecasting with Deep Learning: A Comparative Study,"

Mathematics, vol. 8, no. 9, Art. no. 9, Sep. 2020, doi: 10.3390/math8091441.

[57] A. Srivastava, A. Srivastava, Y. B. Singh, and M. K. Misra, "Deep Learning Models for Stock Market Forecasting: GARCH, ARIMA, CNN, LSTM, RNN," in *Cryptology and Network Security with Machine Learning*, A. Chaturvedi, S. U. Hasan, B. K. Roy, and B. Tsaban, Eds., Singapore: Springer Nature, 2024, pp. 827–839. doi: 10.1007/978-981-97-0641-9_56.

[58] G. Sonkavde, D. S. Dharrao, A. M. Bongale, S. T. Deokate, D. Doreswamy, and S. K. Bhat, "Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications," *Int. J. Financ. Stud.*, vol. 11, no. 3, Art. no. 3, Sep. 2023, doi: 10.3390/ijfs11030094.

[59] J. Shen and M. O. Shafiq, "Short-term stock market price trend prediction using a comprehensive deep learning system," *J. Big Data*, vol. 7, no. 1, Art. no. 1, Aug. 2020, doi: 10.1186/s40537-020-00333-6.

[60] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *Eur. J. Oper. Res.*, vol. 270, no. 2, Art. no. 2, Oct. 2018, doi: 10.1016/j.ejor.2017.11.054.

[61] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Syst. Appl.*, vol. 38, no. 8, Art. no. 8, Aug. 2011, doi: 10.1016/j.eswa.2011.02.068.

[62] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLOS ONE*, vol. 12, no. 7, Art. no. 7, Jul. 2017, doi: 10.1371/journal.pone.0180944.

[63] K. Chen, Y. Zhou, and F. Dai, "A LSTM-based method for stock returns prediction: A case study of China stock market," in *2015 IEEE International Conference on Big Data (Big Data)*, Oct. 2015, pp. 2823–2824. doi: 10.1109/BigData.2015.7364089.

[64] M. Ala'raj and M. F. Abbod, "A new hybrid ensemble credit scoring model based on classifiers consensus system approach," *Expert Syst. Appl.*, vol. 64, pp. 36–55, Dec. 2016, doi: 10.1016/j.eswa.2016.07.017.

[65] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, Art. no. 2, Mar. 1994, doi: 10.1109/72.279181.

[66] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, Art. no. 8, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[67] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, Art. no. 5786, Jul. 2006, doi: 10.1126/science.1127647.

[68] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, Art. no. 56, 2014.

[69] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, Art. no. 6088, Oct. 1986, doi: 10.1038/323533a0.
[70] I. Sutskever, J. Martens, and G. E. Hinton, “Generating text with recurrent neural networks,” in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 1017–1024. Accessed: Aug. 25, 2024. [Online]. Available: https://www.cs.toronto.edu/~jmartens/docs/RNN_Language.pdf
[71] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Jan. 29, 2017, *arXiv*: arXiv:1412.6980. doi: 10.48550/arXiv.1412.6980.
[72] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, “Stock market’s price movement prediction with LSTM neural networks,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 1419–1426. doi: 10.1109/IJCNN.2017.7966019.
[73] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, “Deep learning for stock prediction using numerical and textual information,” in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, Jun. 2016, pp. 1–6. doi: 10.1109/ICIS.2016.7550882.
[74] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, “A Comparison of ARIMA and LSTM in Forecasting Time Series,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2018, pp. 1394–1401. doi: 10.1109/ICMLA.2018.00227.
[75] A. Moghar and M. Hamiche, “Stock Market Prediction Using LSTM Recurrent Neural Network,” *Procedia Comput. Sci.*, vol. 170, pp. 1168–1173, Jan. 2020, doi: 10.1016/j.procs.2020.03.049.
[76] S. Smyl, “A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting,” *Int. J. Forecast.*, vol. 36, no. 1, Art. no. 1, Jan. 2020, doi: 10.1016/j.ijforecast.2019.03.017.
[77] S. Rao, “STOCK PRICE PREDICTION USING LSTM,” May 2024.
[78] X. Pang, Y. Zhou, P. Wang, W. Lin, and V. Chang, “An innovative neural network approach for stock market prediction,” *J. Supercomput.*, vol. 76, no. 3, pp. 2098–2118, Mar. 2020, doi: 10.1007/s11227-017-2228-y.
[79] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Clim. Res.*, vol. 30, no. 1, Art. no. 1, Dec. 2005, doi: 10.3354/cr030079.
[80] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature,” *Geosci. Model Dev.*, vol. 7, no. 3, Art. no. 3, Jun. 2014, doi: 10.5194/gmd-7-1247-2014.
[81] S. Makridakis and M. Hibon, “The M3-Competition: results, conclusions and implications,” *Int. J. Forecast.*, vol. 16, no. 4, Art. no. 4, Oct. 2000, doi: 10.1016/S0169-2070(00)00057-1.

[82]N. J. D. NAGELKERKE, “A note on a general definition of the coefficient of determination,” *Biometrika*, vol. 78, no. 3, Art. no. 3, Sep. 1991, doi: 10.1093/biomet/78.3.691.