

Unveiling Trends in Leading AI and Machine Learning Companies' Stock Price: A Time Series Analysis

MSc Business Analytics

Hamed Soleimani

23699534

Date of submission: 07/17/2024

Word count: 11999

Name of Supervisor: Dr Maksym Koghut

# **Acknowledgement**

I would like to express my sincere gratitude to my supervisor, Dr Maksym Koghut, for his invaluable guidance, patience, and support throughout my dissertation journey. It has been a privilege to work with such a motivated, knowledgeable, meticulous, positive, and approachable individual.

Secondly, I would like to thank my wife and parents for their unwavering support and encouragement, particularly during challenging times during my studies and dissertation writing. Their support has been invaluable to me.

Finally, I extend my gratitude to my colleagues and managers at Datis Arian Qeshm (DOTIN) Company for their understanding and assistance in allowing me to balance my work hours with the demands of full-time master studies.

# **Declaration for Dissertation**

Statement of authenticity:

This Project is an original and authentic piece of work by me. I have fully acknowledged and referenced all secondary sources used. It has not been presented in whole or in part for assessment elsewhere. I have read the Assessment Regulations and am fully aware of the potential consequences of any breach of them.

*Signed: Hamed Soleimani         Date: 07/17/2024*

# **Abstract**

This dissertation explores the potential of time series analysis in predicting significant trends within the historical stock prices of leading AI and ML companies: Alphabet, Meta, and Microsoft. The primary objective is to evaluate the effectiveness of various time series models in forecasting future stock prices, thereby providing useful insights for investors and companies by informing them about the best forecasting models and optimal adjustments to predict future stock prices.

To achieve this aim, the research addresses the following objectives: employing time series models to identify significant long-term and seasonal trends within the historical stock prices of Alphabet, Meta, and Microsoft; comparing the suitability of different time series models; analysing the similarities and differences in the observed trends; and evaluating the accuracy of the models in predicting future trends.

Addressing knowledge gaps in the literature, the research contrasts traditional statistical models like ARIMA with deep learning models such as LSTMs. Traditional models often struggle with non-linear relationships inherent in financial data, while deep learning approaches have shown promise in capturing long-term dependencies. However, discrepancies in comparative studies highlight the need for a comprehensive assessment of these models' effectiveness in the specific context of AI and ML companies' stock prices.

The methodology includes several steps to ensure robustness and accuracy. Initially, data preprocessing and outlier detection are performed to ensure data quality. Exploratory Data Analysis (EDA) follows, involving the use of ACF, PACF plots, and stationarity tests like the Augmented Dickey-Fuller (ADF) and Ljung-Box tests to uncover inherent patterns and ensure data stationarity. Data is then split into training and testing subsets for model evaluation.

Both statistical and deep learning models are employed, including ARIMA-GARCH, LSTM, and GRU. Model performance is evaluated using metrics such as RMSE and MAE. The best-performing models from each category are selected for forecasting mid-term future trends, with comparisons made against actual values to determine prediction accuracy.

Ethical considerations are meticulously addressed, acknowledging the secondary nature of the data sourced from Yahoo Finance and ensuring transparency in all analyses. An Ethos application has been completed to support the ethical conduct of this research.

The findings indicate that all three companies exhibited upward long-term trends, suggesting positive investor outlooks on AI/ML futures. No significant seasonal patterns were identified, pointing to consistent long-term influencing factors on stock prices. Machine learning models, particularly GRUs, outperformed traditional statistical models in capturing underlying trends and predicting mid-term future trends. The GRU model significantly outperformed the ARIMA-GARCH model across all three stocks. Overall, these findings highlight the potential of GRU models for investors and financial analysts seeking to make informed decisions within the AI/ML industry.

However, focusing solely on historical prices might overlook other influencing factors, potentially leading to inaccurate predictions. Future research directions include exploring techniques to mitigate overfitting and improving model generalisability to support high-risk decision-making.

In conclusion, this dissertation demonstrates the superior performance of ML models in forecasting stock prices for leading AI and ML companies, providing useful insights for investors and companies navigating the dynamic AI/ML landscape.

**Table of Contents**

[Acknowledgement 2](#_Toc172064548)

[Declaration for Dissertation 3](#_Toc172064549)

[Abstract 4](#_Toc172064550)

[Table of Contents 6](#_Toc172064551)

[List of Figures 9](#_Toc172064552)

[List of Tables 10](#_Toc172064553)

[1. Introduction 11](#_Toc172064554)

[1.1 Background 11](#_Toc172064555)

[1.2 The Impact of AI and ML on the Stock Market 12](#_Toc172064556)

[1.3 Justifications, Research Aim and Objectives 14](#_Toc172064557)

[1.3.1 Research Aim 14](#_Toc172064558)

[1.3.2 Research Objectives 15](#_Toc172064559)

[1.4 Research Questions 15](#_Toc172064560)

[1.5 Research Approach 15](#_Toc172064561)

[1.6 Dissertation Structure 16](#_Toc172064562)

[2. Literature Review 17](#_Toc172064563)

[2.1 Introduction 17](#_Toc172064564)

[2.2 Time Series Analysis Core Concepts 17](#_Toc172064565)

[2.3 Testing for Stationarity 21](#_Toc172064566)

[2.3.1 Augmented Dickey-Fuller (ADF) 22](#_Toc172064567)

[2.3.2 Ljung-Box 22](#_Toc172064568)

[2.4 Statistical Time Series Forecasting Models 22](#_Toc172064569)

[2.4.1 Moving Average (MA) 22](#_Toc172064570)

[2.4.2 Exponential Smoothing 24](#_Toc172064571)

[2.4.3 Autoregressive Integrated Moving Average (ARIMA) 25](#_Toc172064572)

[2.4.4 ARMA-GARCH (Generalized Autoregressive Conditional Heteroskedasticity) 27](#_Toc172064573)

[2.5 Deep Learning Time Series Forecasting Models 27](#_Toc172064574)

[2.5.1 Recurrent Neural Network (RNN) 27](#_Toc172064575)

[2.5.2 Long Short-Term Memory (LSTM) 29](#_Toc172064576)

[2.5.3 Gated Recurrent Unit (GRU) 31](#_Toc172064577)

[2.6 Comparison and Evaluation of Time-Series Models 32](#_Toc172064578)

[2.7 Knowledge Gaps and Research Questions 34](#_Toc172064579)

[2.8 Conclusion 35](#_Toc172064580)

[3. Methodology 37](#_Toc172064581)

[3.1. Introduction 37](#_Toc172064582)

[3.2. Literature Search 38](#_Toc172064583)

[3.3. Summary of Literature Review 39](#_Toc172064584)

[3.4. Primary data 39](#_Toc172064585)

[3.5. Secondary data 40](#_Toc172064586)

[3.6. Research instrument design 41](#_Toc172064587)

[3.7. Overview of secondary data used for analysis 42](#_Toc172064588)

[3.8. Data analysis 43](#_Toc172064589)

[3.9. Ethics 46](#_Toc172064590)

[3.10. Summary 47](#_Toc172064591)

[4. Data analysis and discussion 48](#_Toc172064592)

[4.1 Introduction 48](#_Toc172064593)

[4.2 Data Preprocessing and Outlier Detection 48](#_Toc172064594)

[4.2.1 Data preprocessing 48](#_Toc172064595)

[4.2.2 Outlier Detection 50](#_Toc172064596)

[4.3 Exploratory Data Analysis (EDA) 52](#_Toc172064597)

[4.3.1 EDA for Alphabet 52](#_Toc172064598)

[4.3.2 EDA for Meta 55](#_Toc172064599)

[4.3.3 EDA for Microsoft 58](#_Toc172064600)

[4.4 Data Splitting 61](#_Toc172064601)

[4.5 Model Implementation 61](#_Toc172064602)

[4.5.1 Alphabet Stock Price Forecast 64](#_Toc172064603)

[4.5.2 META Stock Price Forecast 66](#_Toc172064604)

[4.5.3 Microsoft Stock Price Forecast 69](#_Toc172064605)

[4.6 Forecasting Performance for Mid-Term Unseen Data 72](#_Toc172064606)

[4.7 Summary 75](#_Toc172064607)

[5. Conclusion, Limitations, and Recommendations 77](#_Toc172064608)

[5.1. Introduction 77](#_Toc172064609)

[5.2. Research Conclusion 77](#_Toc172064610)

[5.3. Limitations 82](#_Toc172064611)

[5.4. Recommendations for future research 83](#_Toc172064612)

[5.5. Conclusion 84](#_Toc172064613)

[Reference List 86](#_Toc172064614)

[Appendices 100](#_Toc172064615)

[I. Personal Reflection 100](#_Toc172064616)

[II. Ethos Application 102](#_Toc172064617)

[III. Outliers Data 108](#_Toc172064618)

[IV. Ljung\_Box Test Results After First-Order Differencing 110](#_Toc172064619)

[V. Mid-Term Stock Price Forecasts (Real vs. Predicted) 119](#_Toc172064620)

[VI. Python Code for Time Series Analysis and Forecasting 123](#_Toc172064621)

[VII. Sample of Selected and Analysed Articles for Literature Review 160](#_Toc172064622)

**List of Figures**

[Figure 1: Stationary vs. Non-Stationary time series (source: Santra, 2023) 18](#_Toc172023851)

[Figure 2: Differencing (source: Kai et al., 2022) 19](#_Toc172023852)

[Figure 3: Time Series Components (source: Valse, 2020) 20](#_Toc172023853)

[Figure 4: Autocorrelation Function (ACF) with both trend and seasonality (source: Hyndman et al., 2021) 21](#_Toc172023854)

[Figure 5: Comparison of Moving Averages (source: Kadam, 2012) 24](#_Toc172023855)

[Figure 6: Exponential Smoothing (source: Dash, 2020) 25](#_Toc172023856)

[Figure 7: RNN Architecture (source: Beniwal et al., 2024) 29](#_Toc172023857)

[Figure 8: LSTM Architecture (source: Beniwal et al., 2024) 30](#_Toc172023858)

[Figure 9: GRU Architecture (source: Jing et al., 2021) 32](#_Toc172023859)

[Figure 10: Historical stock prices of Alphabet, META, and Microsoft 43](#_Toc172023860)

[Figure 11: Research Methodology 44](#_Toc172023861)

[Figure 12: Best-fitting Distributions Plot of Alphabet, Meta, and Microsoft 50](#_Toc172023862)

[Figure 13: Original ACF and PACF plots of Alphabet 53](#_Toc172023863)

[Figure 14: ACF and PACF plots of Alphabet after first-order differencing 54](#_Toc172023864)

[Figure 15: Alphabet original trends overlaid by 50-day simple moving average 55](#_Toc172023865)

[Figure 16: Original ACF and PACF plots of META 56](#_Toc172023866)

[Figure 17: ACF and PACF plots of META after first-order differencing 57](#_Toc172023867)

[Figure 18: Alphabet original trends overlaid by 210-day simple moving average 58](#_Toc172023868)

[Figure 19: Original ACF and PACF plots of Microsoft 59](#_Toc172023869)

[Figure 20: ACF and PACF plots of Microsoft after first-order differencing 60](#_Toc172023870)

[Figure 21: Microsoft original trends overlaid by 90-day simple moving average 61](#_Toc172023871)

[Figure 22: Alphabet Baseline and Forecasting Models’ Charts 65](#_Toc172023872)

[Figure 23: META Baseline and Forecasting Models’ Charts 68](#_Toc172023873)

[Figure 24: Microsoft Baseline and Forecasting Models’ Charts 70](#_Toc172023874)

[Figure 25: Mid-term Prediction Charts of Alphabet, META, and Microsoft 73](#_Toc172023875)

**List of Tables**

[Table 1: Advantageous and disadvantageous of secondary data 41](#_Toc172068054)

[Table 2: Descriptive Statistics for stock prices of Alphabet, Meta, and Microsoft 49](#_Toc172068055)

[Table 3: Best-fitting distribution and outlier statistics for Alphabet, Meta, and Microsoft stocks 51](#_Toc172068056)

[Table 4: Python Libraries utilised for model implementation 63](#_Toc172068057)

[Table 5: Tuned Parameters of Forecasting Models for Alphabet Stock Price Prediction 64](#_Toc172068058)

[Table 6: Comparative Results of Forecasting Models and Performance Metrics for Alphabet 66](#_Toc172068059)

[Table 7: Tuned Parameters of Forecasting Models for META Stock Price Prediction 67](#_Toc172068060)

[Table 8: Comparative Results of Forecasting Models and Performance Metrics for META 68](#_Toc172068061)

[Table 9: Tuned Parameters of Forecasting Models for Microsoft Stock Price Prediction 70](#_Toc172068062)

[Table 10: Comparative Results of Forecasting Models and Performance Metrics for Microsoft 71](#_Toc172068063)

[Table 11: Summary of real and predicted values for Alphabet 73](#_Toc172068064)

[Table 12: Summary of real and predicted values for META 74](#_Toc172068065)

[Table 13: Summary of real and predicted values for Microsoft 75](#_Toc172068066)

[Table 14: Comparative Table for Mid-term Forecasting of Alphabet, META, and Microsoft 75](#_Toc172068067)

[Table 15: Outliers of Alphabet Stock Prices 109](#_Toc172068068)

[Table 16: Outliers of META Stock Prices 110](#_Toc172068069)

[Table 17: Outliers of Microsoft Stock Prices 110](#_Toc172068070)

[Table 18: Ljung\_Box Test Results of Alphabet 113](#_Toc172068071)

[Table 19: Ljung\_Box Test Results of META 116](#_Toc172068072)

[Table 20: Ljung\_Box Test Results of Microsoft 119](#_Toc172068073)

[Table 21: Mid-Term Stock Price Forecasts (Real vs. Predicted) for Alphabet 120](#_Toc172068074)

[Table 22: Mid-Term Stock Price Forecasts (Real vs. Predicted) for META 122](#_Toc172068075)

[Table 23: Mid-Term Stock Price Forecasts (Real vs. Predicted) for Microsoft 123](#_Toc172068076)

[Table 24: Sample of selected and analysed articles for literature review 161](#_Toc172068077)

**Introduction**

This chapter outlines the aim, objectives and justification of the research. It then delves to the research background to highlight significance of research questions.

* 1. **Background**

Stocks have long been considered a cornerstone asset for investors. They offer the potential for strong returns on investment over time and high liquidity. This is exemplified by the FTSE 100 on the London Stock Exchange, which delivered a total return of 645.2% (equivalent to an annualised return of 5.4%) between 1984 and 2022 (Bright, 2023). Publicly traded companies are required by law to disclose their financial status, providing a wealth of information for investors. Additionally, regulations governing the stock market help mitigate various financial risks.

However, maximising profits from stock investments demands a sound understanding of market dynamics and a well-defined strategy for buying and selling. This, in turn, hinges on accurate prediction of stock prices. Stock price prediction methods can be broadly categorized into two main approaches: fundamental analysis and technical analysis (Devadoss, 2013).

Fundamental analysis focuses on a company's intrinsic value, considering its future potential and financials. A stock trading below its intrinsic value might be a good buy, while an overvalued stock might be a sell candidate. Technical analysis focuses solely on the stock price itself, using charts and mathematical tools to identify trends and predict future price movements. This approach ignores company fundamentals and relies on historical price data (Raghunathan, 2007).

In response to these limitations, time series analysis is a statistical technique used to examine observations collected on a variable of interest over time. This technique allows researchers to uncover patterns within historical datasets, such as seasonality (Montgomery et al., 2008). In the context of this research, time series analysis will be utilised to analyse the historical stock prices of leading Artificial Intelligence (AI) and Machine Learning (ML) companies.

Employing times series analysis, help us uncover long-term patterns within historical stock price data. This involves analysis techniques to identify trend or seasonality patterns within existing data (Shanaev et al., 2020). By analysing these trend components, we can assess the overall growth trajectory of AI/ML companies.

In a general sense, time series analysis will help us to understand if stock price movement in recent years has been impacted by technological advancements. While time series analysis mainly focuses on historical data, its power can be leverage to forecast future trends as well. Techniques such as ARIMA (Hyndman et al., 2021) or machine learning models like LSTM (Asokan, 2022) utilise historical patterns in data to predict future prices.

* 1. **The Impact of AI and ML on the Stock Market**

The rapid advancements in AI and ML are driving a profound revolution across industries, fundamentally transforming their core operations (Rahman, 2023). Artificial Intelligence refers to the development of computer programs or intelligent machines that can mimic or even surpass human cognitive abilities in specific domains (Kaplan, 2016). For instance, AI-powered programs like AlphaGo have achieved superhuman performance in complex strategy games like GO by defeating the world grandmaster, Lee Sedol (Silver et al., 2016), while chess programs like Stockfish can defeat grandmasters consistently (Naroditsky, 2015). Similarly, advancements in Natural Language Processing (NLP) enables machines to understand human-like languages and respond to them with increasing accuracy (Arkhangelskaya et al., 2023).

Machine Learning, on the other hand, empowers these intelligent machines to continually improve by learning from data without the need for explicit programming. Unlike traditional algorithms with fixed set of instructions, ML models can identify patterns and relationships within large datasets, allowing them to adapt and make predictions on new data (Kaplan, 2016). This self-learning capability has fuelled significant breakthroughs in robotics, where AI-powered robots are now performing complex tasks with greater intelligence and adaptability. These tasks include autonomous navigation in transportation systems like self-driving cars, object recognition for applications like inventory management and predictive maintenance to prevent equipment failures (Soori et al., 2023).

The utilisation of AI and ML technologies is significantly impacting the global economy and financial markets. A 2022 PwC report predicts that nearly 45% of total global economic profit by 2030 will be generated through product enhancements driven by AI and ML. This economic surge, driven by three key factors, will undoubtedly impact stock prices (PWC, 2022). AI-powered automation will streamline tasks across industries, boosting productivity. Secondly, AI will augment the workforce, empowering workers with intelligent tools that unlock greater human potential. Finally, AI's ability to analyse vast data sets will personalise consumer experiences, fuelling demand and economic growth across sectors.

Companies developing and utilising these technologies are experiencing substantial growth, attracting investor interest and influencing stock prices (Bughin et al., 2018). Understanding the current trends in how leading companies utilise AI and ML, and the subsequent impact on their stock prices, can provide valuable insights for both companies and their investors. This knowledge can help them assess growth potential, identify related risks, and make more informed investment decisions.

* 1. **Justifications, Research Aim and Objectives**

The economic potential of AI and ML has been extensively explored. Research by scholars such as Chia-Hui (2021) has highlighted their disruptive power, ability to drive innovation, and contribution to economic growth. However, there is a gap in understanding how these developments translate into the concrete stock market performance of leading AI and ML companies themselves. While the broader economic impact of AI and ML advancements is well-documented (Barclays, 2024), a gap exists in understanding how these advancements are reflected in the stock market performance of leading companies driving this revolution.

* + 1. **Research Aim**

The primary objective of this research is to utilise time series analysis to evaluate the potential for predicting significant trends within historical stock prices of leading AI and ML companies. This analysis will involve uncovering long-term and seasonal patterns in the historical data, and subsequently assess the effectiveness of the chosen models for forecasting future trends.

By utilising time series analysis, this research aims to offer valuable insights withing AI and ML landscape for both investors and companies. Time series analysis can help investors make more informed decision by identifying long-term growth trajectories and assessing the impact of AI and ML’s disruptive power. For companies, this analysis can provide a better understanding of how their stock market performance aligns with their technological advancements, allowing them to optimise future stock market strategies.

* + 1. **Research Objectives**

To achieve this aim, the research will address the following specific objectives:

1. Employ time series models to identify significant long-term and seasonal trends within the historical stock prices of Alphabet, Meta, and Microsoft.
2. Compare the suitability of different time series models for uncovering underlying trends in the stock prices of leading AI and ML companies.
3. Analyse the similarities and differences observed between the long-term and seasonal trends exhibited in the chosen companies' stock prices.
4. Evaluate the accuracy of the chosen time series models in predicting future trends in the stock prices of the leading AI and ML companies.
   1. **Research Questions**

In order to achieve the research aim and objectives outlined above, this research will address the following research questions:

1. How well can the chosen time series models identify significant long-term or seasonal trends within the historical stock prices of leading Artificial Intelligence (AI) and Machine Learning (ML) companies (Alphabet, Meta, and Microsoft)?
2. In comparison to each other, how suitable are the chosen time series models for uncovering underlying trends in the stock prices of leading AI and ML companies?
3. Are there any significant similarities or differences observed between the long-term and seasonal trends exhibited in the chosen companies' stock prices?
4. How accurately can the chosen time series models predict future trends in the stock prices of the leading AI and ML companies?
   1. **Research Approach**

This research will utilise secondary data sources for analysis. Historical stock prices will be collected from Yahoo Finance.

To ensure a robust analysis, the data selection process will consider established practices in financial time series analysis. The aim is to capture both long-term trends and potential seasonal patterns, while maintaining relevance to recent developments in the dynamic AI and ML sector. A specific timeframe will be chosen to balance these considerations.

Three leading global companies in the field of AI and ML have been selected for analysis: Alphabet (Google), Meta Platforms (Facebook), and Microsoft. These companies are recognised for their dominant market positions due to their market capitalisation and their well-established applications of AI and ML technologies.

* 1. **Dissertation Structure**

This dissertation is structured into five chapters. Chapter 1 outlines the research aims, objectives, and justification. Chapter 2 explores relevant literature on applying time series analysis to stock market forecasting. Chapter 3 details the research methodology and data collection. Chapter 4 presents the research findings and analysis. Finally, Chapter 5 discusses the results, evaluates the study, and recommends future research directions.

1. **Literature Review** 
   1. **Introduction**

Stock price prediction is crucial for maximising investment returns. Time series analysis, analysing data sequences over time to identify patterns for forecasting, plays a vital role. Selecting the most effective model is essential, and finance boasts a rich history of developing specialised models for financial data.

This chapter delves into the core concepts of time series analysis. We will then explore some of the most commonly applied forecasting models. Finally, we will discuss a comparison and evaluation of these models in the context of real-world stock price prediction, drawing insights from various research studies and literature.

* 1. **Time Series Analysis Core Concepts**

Before diving into specific models, a brief introduction to the core time series concepts is appropriate.

* **Stationarity**

A stationary time series does not depend on the observation time and so will exhibit constant statistical properties (like mean and variance) in the long run. This contrasts with time series exhibiting seasonality or trends, where the time of observation affects the value. In this sense, cyclical behaviour can also be considered stationary because it doesn't have a fixed period within the data (Hyndman et al., 2021).

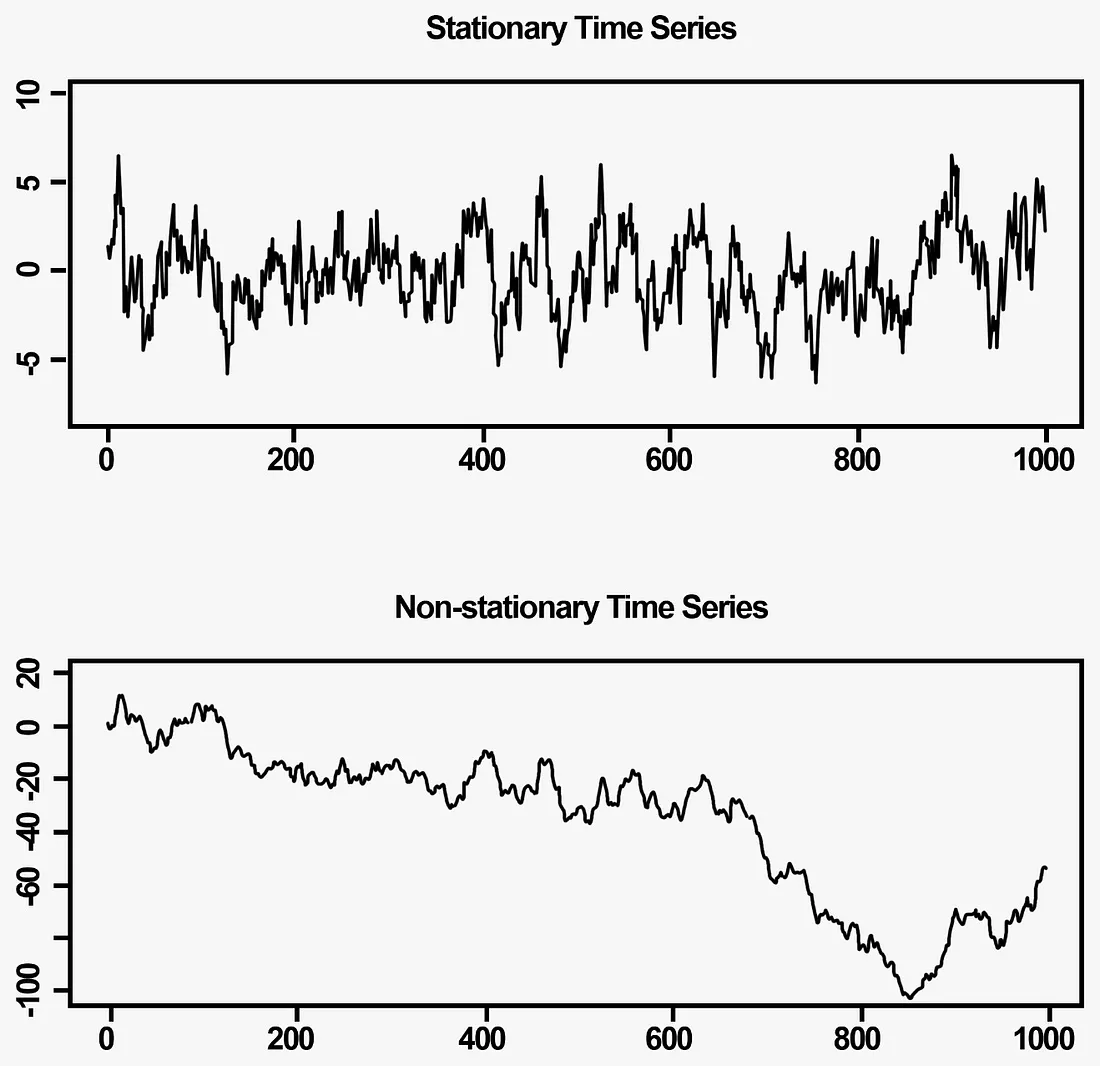


Figure : Stationary vs. Non-Stationary time series (source: Santra, 2023)

* **Differencing**

To transform a non-stationary time series into a stationary one, the difference between the current observation and its previous value (or a lagged value) is computed. This stabilises the variance of the time series. By removing the changes in the series, the mean becomes more stable, and any trends or seasonality patterns are removed (Hyndman et al., 2021).

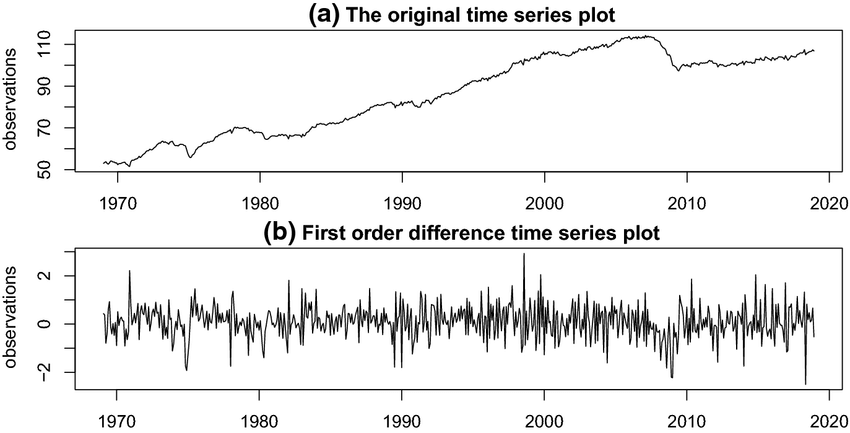


Figure : Differencing (source: Kai et al., 2022)

* **Trends**

A long-term decrease or increase in time series data, regardless of its linearity, suggests the presence of a trend. This trend might even change direction, for instance, from decreasing to increasing (Hyndman et al., 2021).

* **Seasonality**

Seasonality arises when a time series is influenced by recurring seasonal factors like day of the week or time of the year (Hyndman et al., 2021).

* **Cyclic**

A rise and fall in data without any constant frequency is termed a cycle. For instance, business cycles fluctuate due to changing economic conditions (Hyndman et al., 2021).

* **Irregular Variations**

Unpredictable fluctuations in a time series with no discernible pattern are termed irregular variations. Unlike seasonal or cyclical variations, they can occur at any time and often stem from unforeseen events (Valse, 2020), such as unexpected news announcements.

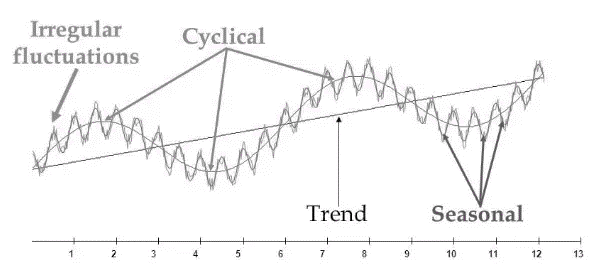


Figure : Time Series Components (source: Valse, 2020)

* **Autocorrelation Function (ACF)**

The ACF measures the correlation between a time series and its lagged observations (past values). Positive or negative values at early lags suggest trends, while spikes at specific lags indicate seasonality. Values close to zero imply little to no correlation. High ACF values at some lags, particularly for pronounced trends, can signal non-stationarity. However, interpreting the ACF for stationarity requires considering the overall pattern and employing additional tests like the Dickey-Fuller test (Hyndman et al., 2021; Brockwell, 2016).

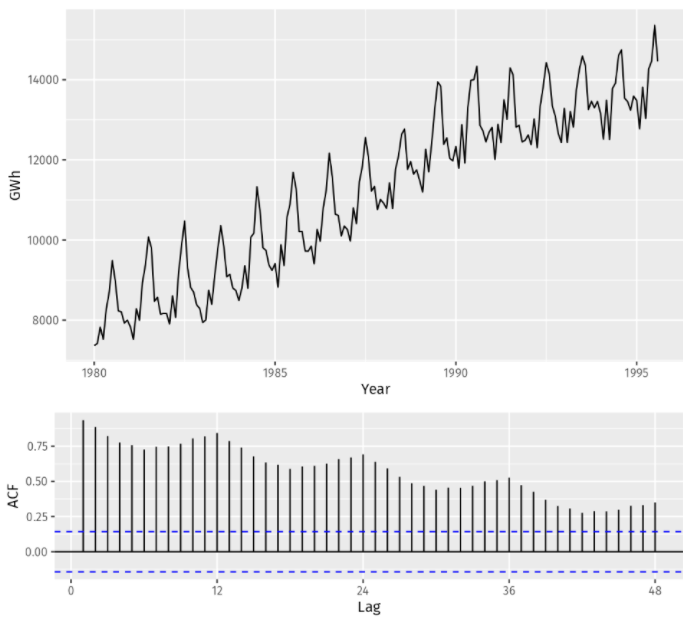


Figure : Autocorrelation Function (ACF) with both trend and seasonality (source: Hyndman et al., 2021)

* **Partial Autocorrelation Function (PACF):**

PACF is similar to the ACF with a difference that unlike ACF, it measures only the direct correlation of time series with its lagged observations excluding the influences of intermediate lags. A high PACF value suggest strong relationship between the current value and its lagged value. Similar to ACF, a value of zero or close to it indicates no direct correlation or a weak one at that lag (Dégerine et al., 2003).

* 1. **Testing for Stationarity**
     1. **Augmented Dickey-Fuller (ADF)**

This statistical test assesses whether a time series is stationary (constant mean, variance, and autocorrelation). The null hypothesis implies non-stationarity, while the alternative is stationarity.

A more negative test statistic indicates stronger evidence against non-stationarity. If the statistic falls below critical values at a chosen significance level (e.g., 5%), we fail to reject the null hypothesis, suggesting potential non-stationarity. Additionally, a p-value less than the significance level (e.g., 0.05) leads to rejection of the null hypothesis, implying stationarity (Dickey et al., 1979; Rizwan, 2011).

* + 1. **Ljung-Box**

In a well-specified model for a stationary time series, residuals should be independent and identically distributed (i.i.d.). This means they behave randomly and have no "memory" of past values. Non-random residuals (serial correlation) suggest issues with the model or data stationarity.

The Ljung-Box test's null hypothesis states no serial correlation in residuals (present residuals are unrelated to past ones). Non-stationary data (trends or seasonality) can lead to serial correlation, violating the null hypothesis. A low p-value from the Ljung-Box test on residuals indicates a potential stationarity issue (Ljung et al., 1978).

* 1. **Statistical Time Series Forecasting Models**
     1. **Moving Average (MA)**

Moving averages (MAs) are a popular technical indicator used in financial trading to identify trends and potential support and resistance levels. Considering stock data as example, this approach assumes that recent prices hold more significance for predicting future prices compared to older data. For example, a 3-period moving average calculates the average data over the last three periods (days, weeks, etc.) to predict the upcoming period's price. This implies that the price of future time periods is weighting by the patterns observed in the most recent data points. (Jain et al., 2005)

There are three main types of moving averages **(Murphy, 1999)**:

* **Simple Moving Average (SMA)**

**This basic MA takes the average of a chosen number of price points (e.g., closing prices). Recent data has equal weight as older data, making it more sensitive to recent price changes.**

* **Linearly Weighted Moving Average (LWMA)**

**This MA assigns higher weights to recent prices within the window, giving them more influence. While recent data is prioritized, some historical data is still considered.**

* **Exponentially Weighted Moving Average (EMA)**

**This MA assigns exponentially decreasing weights to past prices, with the most recent price having the most influence. This approach heavily emphasizes recent information while incorporating some historical context.**

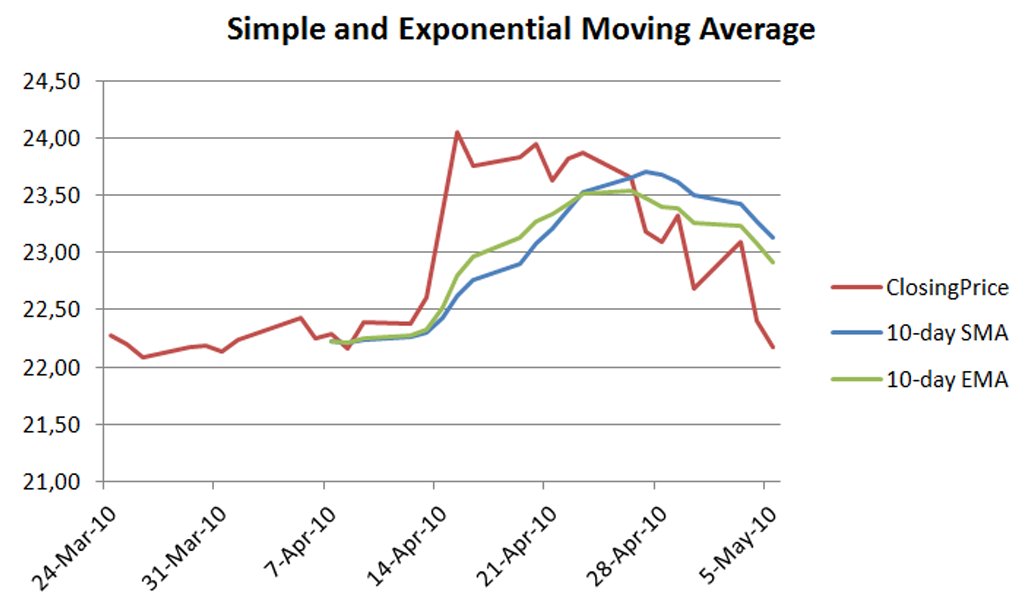
****

Figure : Comparison of Moving Averages (source: Kadam, 2012)

* + 1. **Exponential Smoothing**

Exponential smoothing is a forecasting technique that assigns higher weights to recent data points compared to older ones. It emphasizes the most recent observations while progressively decreasing weight for older data. This approach offers quick and reliable forecasts (Nugus, 2009), making it a valuable tool across various industries since its introduction (Holt, 1957; Brown, 1959; Winters, 1960). Exponential smoothing comes in three main types: single, double, and triple. These methods can model time series data with varying degrees of complexity, accounting for trend level, trend slope, and even seasonal components (Yibin, 2019).

Holt’s linear model excel at forecasting trends but it tends to over-predict if the trend is expected to level off. To address this limitation, researchers like Gardner & McKenzie (1985) introduced a parameter that dampens the trend, making more realistic forecast especially in situation which the trend eventually stabilise. This optimisation has proven highly successful in improving forecasting accuracy (Hyndman et al., 2021).

Holt-Winters builds on exponential smoothing by capturing seasonal patterns with weights in a multiplicative model. Simpler and requiring less data than Box-Jenkins, its strength lies in "back-forecasting" for superior accuracy, though initially computationally expensive (Rumbe et al., 2024).

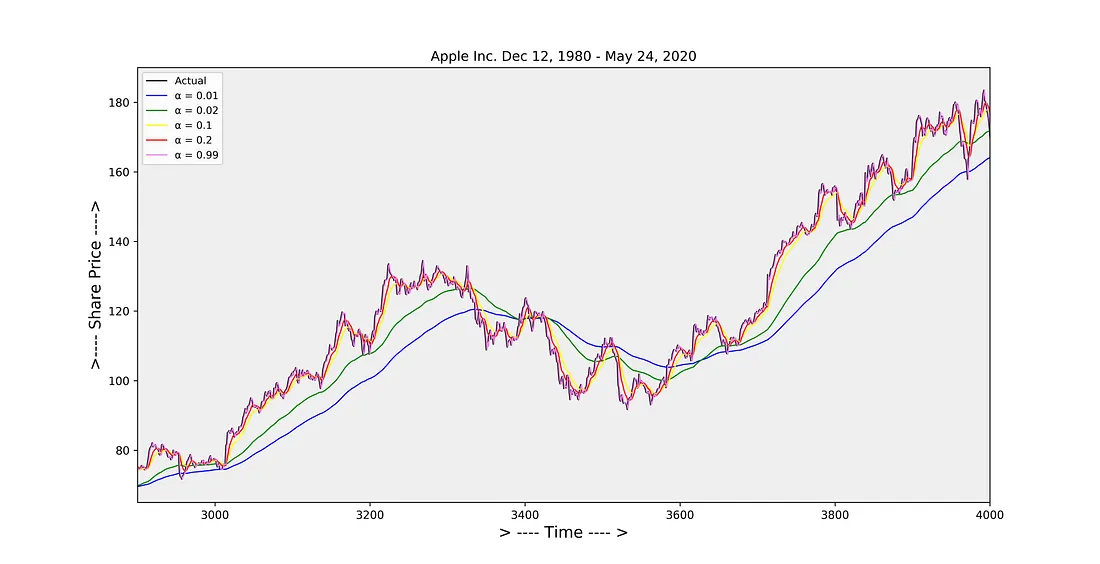


Figure : Exponential Smoothing (source: Dash, 2020)

* + 1. **Autoregressive Integrated Moving Average (ARIMA)**

In the real world, observations of a variable often exhibit a dependence on past values, creating a **serial dependency** across time. For such data, exponential smoothing models become less effective in forecasting. To address this limitation, the ARMA (Auto Regressive Moving Average) model was developed by combining the strengths of Moving Average (MA) and Autoregressive (AR) approaches. Later, Box and Jenkins (Box et al., 1976) introduced the concept of stationarity, leading to the creation of more robust ARIMA models. Finally, to handle seasonal patterns in data, SARIMA (Seasonal ARIMA) models were introduced (Shumway et al., 2015).

ARIMA models (p, d, q) are characterised by 3 key components (Shumway et al., 2015):

* **Autoregression (AR)**

This component leverages past observations of a time series to predict its current value. The parameter 'p' determines the number of past data points included in the model for forecasting.

* **Integrated (I)**

This step ensures the data is stationary**.** If trends or seasonality are present, differencing is applied 'd' times to achieve stationarity.

* **Moving Average (MA)**

MA incorporates past forecast errors -the difference between actual and predicted values. The parameter 'q' determines the number of past errors considered to smooth out fluctuations and potentially improve forecast accuracy.

**ARIMA models require careful selection of p, d, and q parameters, often through statistical tests or validation. A basic example, ARIMA(0,1,0), is essentially a random walk model (d=1 for stationarity) that ignores past values (p=0) and errors (q=0) (Montgomery et al., 2008).**

SARIMA (p, d, q) \* (P, D, Q)s models are an extension of ARIMA models, addressing a critical aspect missing in them: seasonality. They explicitly consider recurring patterns in the data based on time intervals. SARIMA introduces four additional parameters to capture seasonality. ‘P’ determines the number of past seasonal values; ’D’ the number of times data needs to be seasonally differenced; ‘Q’ the number of past seasonal forecast errors; and ‘s’ defines the number of observations in a single seasonal period (example: monthly = 12 and quarterly= 4) (Malki et al., 2022).

* + 1. **ARMA-GARCH (Generalized Autoregressive Conditional Heteroskedasticity)**

To capture volatility clustering of a time series which is variance fluctuation around the mean, GARCH models were introduced (Engle, 1982; Bollerslev, 1986). Since many time-series exhibits periods of high volatility followed by similar periods or vice versa, ARMA based models cannot capture these phenomena alone. However, combining ARMA models with GARCH models can recognise this volatility over time. This allows for better prediction by considering the past squared errors in calculating the variance of forecast errors (Kim, 2022).

GARCH models include two parameters of p and q. these parameters determine how past volatility influences the current conditional variance. Combination of two models, ARMA (p, d, q) – GARCH (p, q) works in two steps: First, ARMA model is applied to predict future values’ mean based on the past values, differencing and forecast errors of past observations in time-series and then using GARCH component of past squared errors (volatility) to predict the future conditional variance. The obtained value explains how much data is likely to fluctuate around the mean (Lee, 2013).

* 1. **Deep Learning Time Series Forecasting Models**
     1. **Recurrent Neural Network (RNN)**

Artificial Neural Networks (ANNs) are a powerful computational approach inspired by the structure and function of the human brain. They consist of a vast network of interconnected processing units, often referred to as artificial neurons or processing elements (PEs). Unlike biological neurons with complex inner workings, these PEs are simpler computational units. Through a training process, ANNs can adjust the connections (synapses) between these PEs and so capture knowledge and patterns from the data they are exposed to (Guresen et al., 2011)

Recurrent neural networks (RNNs) are a specific type of artificial neural network architecture designed to excel at processing temporal sequential data (Guo et al., 2012). Unlike traditional neural networks that process information in a single forward pass, RNNs are adept at handling sequences. They achieve this by employing a feedback loop mechanism that allows the output from one step to be fed back as input to the next step. This creates a form of internal memory within the network, enabling it to learn from past information and improve its understanding of future inputs in the context of the current observation within a sequence. In essence, RNNs overcome the limitation of traditional neural networks, which typically process information in a single forward pass (Ayodele et al., 2021).

RNNs operate by forecasting a sequence one element at a time. During each step, they combine the current input with the prediction from the previous step, retrieved from their internal memory. This combined information is then processed to generate a new output and update the internal memory in preparation for the next element in the sequence (Hewamalage et al., 2021).

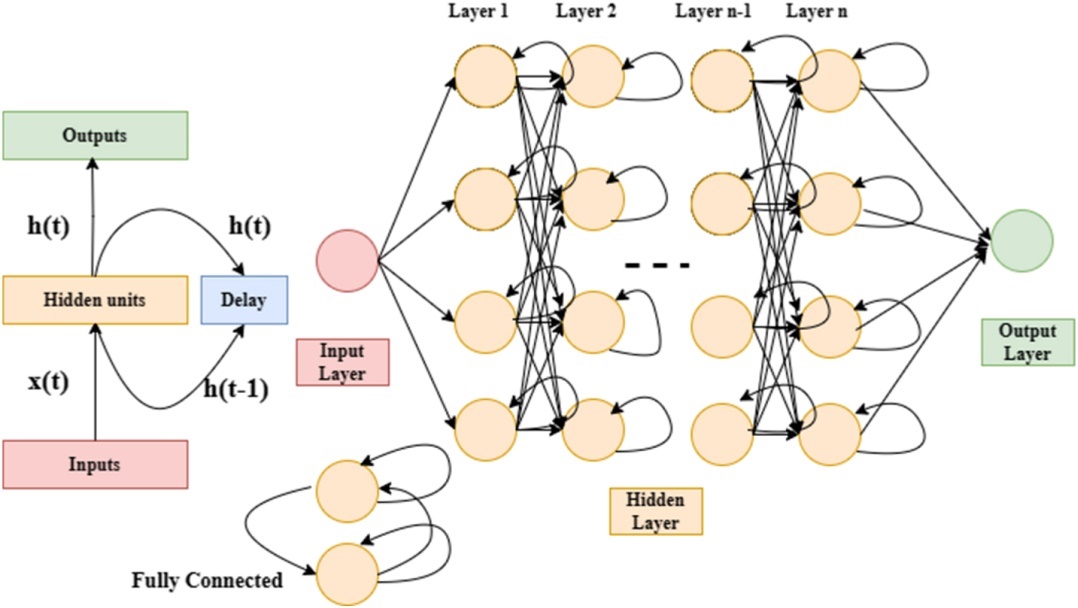


Figure : RNN Architecture (source: Beniwal et al., 2024)

However, RNNs face challenges when dealing with real-world applications involving long sequences. They can struggle to learn dependencies between elements that are far apart in the sequence, a phenomenon known as the vanishing gradient problem. Additionally, RNNs are limited by their internal memory capacity, which restricts their ability to effectively learn from very long sequences (Liu et al., 2023).

* + 1. **Long Short-Term Memory (LSTM)**

To address the vanishing gradient problem that limits traditional Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) models were introduced (Hochreiter & Schmidhuber, 1997). LSTMs excel at learning long-term dependencies within sequences, allowing them to effectively analyse information even for sequences exceeding 1000 steps. They achieve this by employing a specially designed architecture that incorporates gates within their cell structure. These gates help manage the flow of information and overcome the vanishing gradient problem (Zhang et al., 2023).

An LSTM memory cell is the core component of an LSTM network and contains four major elements:

* Cell State: This acts as a long-term memory unit, storing, reading, writing and forgetting information relevant to the task. It does this by using three gates.
* Input Gate: This gate controls the flow of new information into the cell state.
* Output Gate: This gate determines what information from the cell state is used as output.
* Forget Gate: This gate decides what information to forget from the cell state, allowing the network to selectively retain or discard information over time.

Each gate utilises an activation function to compute a value between 0 and 1. The input gate controls the values entering the cell, the output gate controls the values used for output, and the forget gate controls the information retained within the cell (Gao et al., 2017).

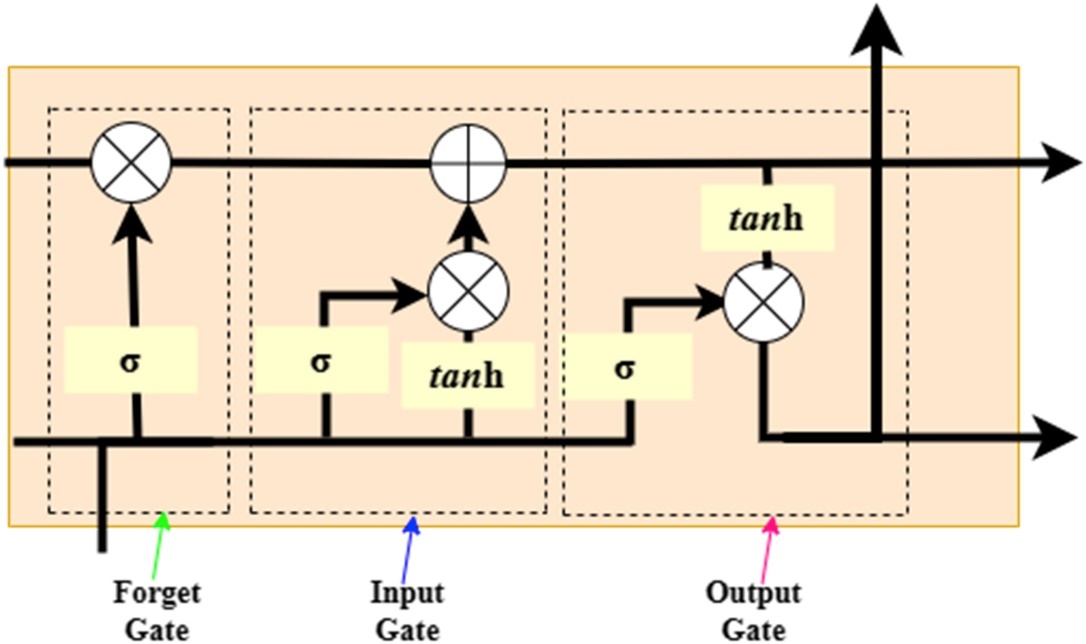


Figure : LSTM Architecture (source: Beniwal et al., 2024)

LSTMs excel at handling financial time series data due to their ability to capture long-term dependencies. However, non-stationary data and noise can challenge LSTMs, especially with sudden changes and extended prediction horizons, potentially reducing performance and forecast accuracy. To address these issues, researchers have developed LSTM-based models like LSTM-BN for financial predictions (Fang et al., 2023).

* + 1. **Gated Recurrent Unit (GRU)**

Gated Recurrent Units (GRUs), proposed by Cho et al. (2014) have emerged as a popular alternative to Long Short-Term Memory (LSTM) networks. While inspired by LSTMs, GRUs offer a more streamlined architecture, making them computationally efficient and well-suited for various applications (Pil-Soo et al., 2018).

One key difference lies in the gating mechanisms. GRUs combine the forget and input gates of LSTMs into a single update gate. This simplifies the information flow within the network. Unlike LSTMs with separate memory cells, GRUs modulate information flow directly through the hidden state, eliminating the need for dedicated memory units (ArunKumar et al., 2021).

This streamlined approach translates to faster training times and almost the same performance compared to LSTMs. The update gate acts as a controller, determining how much of the previous hidden state (information) should be retained and how much should be replaced with new information from the current input. This controlled flow of information allows GRUs to effectively learn long-term dependencies within sequences (Wang et al., 2019).

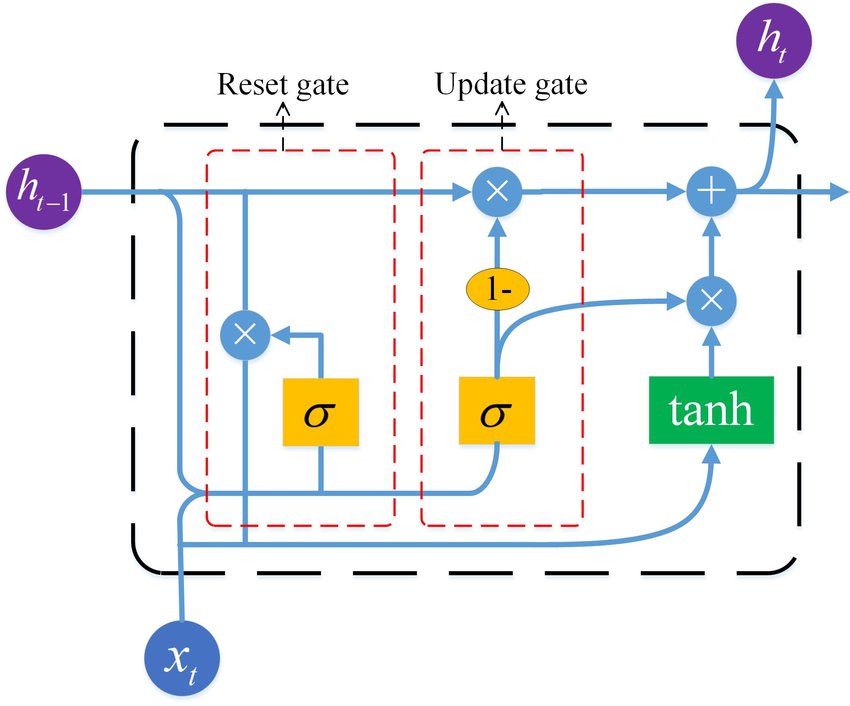


Figure : GRU Architecture (source: Jing et al., 2021)

However, a potential drawback of GRUs is their lack of a dedicated forget gate. While LSTMs can explicitly forget irrelevant information, GRUs rely solely on the update gate to manage the information flow. This could limit their ability to handle certain tasks where selective forgetting might be crucial (Srivatsavaya, 2023).

* 1. **Comparison and Evaluation of Time-Series Models**

The ARIMA model is widely used in finance for time series forecasting. Studies highlight its frequent use for predicting stock prices (Hu et al. 2020; Idrees et al. 2019). ARIMA excels in short-term predictions by leveraging past observations, requiring minimal parameters while balancing complexity and accuracy (Box et al., 1976; Ariyo et al., 2014; Wadi et al., 2018).

However, while ARIMA and similar models excel at capturing linear relationships within data, they can struggle with the complexities introduced by non-linear factors (Earnest et al., 2005) such as unexpected events or market sentiment. This is because ARIMA assumes stationarity, which might not always hold true for stock prices that often exhibit trends and seasonality. The process of differencing data to achieve stationarity can also lead to information loss, potentially overlooking valuable insights (Hyndman et al., 2021).

In contrast, combining ARIMA with models like GARCH can address volatility, a common challenge in stock price data. This is highlighted in a study using GARCH to forecast volatility and return of the four major banks (Hu et al., 2020). However, optimizing these combined models can be computationally complex (Engle, 2001).

Other statistical models have also shown promise in dynamic environments like the Dhaka Stock Exchange. Kim (2022) explores the ARMA-GARCH model with positive results. Additionally, methods like exponential smoothing have been applied to detect share price movements and predict daily prices (Shahid et al., 2020).

Recent advancements in computational power have enabled the use of novel machine learning techniques for financial forecasting, such as GRUs, LSTMs, and RNNs (Selvin et al., 2017). For example, Selvin et al. (2017) successfully applied neural network models to capture the dynamic behaviour of data for predicting the stock prices of conglomerates listed on the National Stock Exchange of India (NSE). Furthermore, Chen et al. (2015) conducted similar research to predict stock price return on the China Stock Market, also achieving good result. A significant advantage of these models, particularly RNNs, is their ability to handle non-linear relationships within data. Unlike traditional time series models like ARIMA, they do not require the data to be stationary (have constant statistical properties over time) (Hyndman et al, 2021). Deep learning techniques based on RNNs, particularly Long Short-Term Memory (LSTM) networks, have shown promise in stock price forecasting due to their ability to learn and remember long-term dependencies within financial data (Sutskever et al., 2014). Studies like Beniwal et al. (2024) further support this, demonstrating that LSTMs outperform other deep learning models in predicting the performance of various global stock indices.

Several comparative studies have been conducted to evaluate the performance of different models in stock price prediction. For instance, Siami-Namini et al. (2018) compared the performance of LSTM, a deep learning model, with ARIMA on 23 years of monthly data (1985-2018) for prominent indices like NASDAQ. Their findings suggest that LSTM outperformed ARIMA by a significant margin, with an error reduction rate of 84%-87% as measured by RMSE (Root Mean Squared Error). However, Kobiela et al. (2022) observed contrasting results when applying ARIMA and LSTM to predict NASDAQ stock exchange data. In their study, ARIMA achieved the best performance while using just one feature for modelling and forecasting multiple periods.

These contrasting outcomes highlight the importance of considering several factors when selecting a model for stock price prediction. Notably, the target population and the characteristics of the collected dataset can significantly impact model performance. Additionally, the prediction window, which refers to the timeframe for which forecasts are made, is another crucial factor. For example, Siami-Namini et al. (2018) focused on one-month predictions, whereas Chen et al. (2015) aimed for 30-day forecasts.

* 1. **Knowledge Gaps and Research Questions**

Having reviewed various time series forecasting models, it becomes evident that while traditional statistical models like ARIMA excel in specific scenarios, they struggle with non-linear relationships and complexities inherent in financial data (Hu et al., 2020). The requirement for stationarity can also lead to information loss through differencing (Hyndman et al., 2021). Deep learning approaches, particularly LSTMs, have demonstrated promising results due to their ability to capture long-term dependencies and handle non-linearity (Beniwal et al., 2024). However, comparative studies yield conflicting findings, with some highlighting LSTM's superiority over ARIMA (Siami-Namini et al., 2018) and others showcasing ARIMA's edge (Kobiela et al., 2022). These discrepancies can be attributed to variations in target data, prediction horizons (short-term vs. long-term), and feature selection.

Despite the advancements in computational power and novel machine learning techniques, there remains a lack of consensus on the most suitable models for predicting stock prices, particularly for leading AI and ML companies. This ongoing debate underscores the need for a comprehensive assessment that evaluates the suitability of various time series models for predicting trends in leading AI and ML company stock prices (Siami-Namini et al., 2018; Kobiela et al., 2022; Beniwal et al., 2024). While deep learning models hold promise due to their ability to capture complex relationships, a systematic comparison with traditional models is necessary to establish their relative effectiveness in this specific context.

Following the knowledge gaps described above, the main aim of this research is to employ a comprehensive evaluation framework to assess the effectiveness of various time series models, including traditional statistical models and deep learning models, in predicting long-term and seasonal trends in the stock prices of leading AI and ML companies.

Accordingly, the guiding research questions that arose from the literature review are the effectiveness of chosen models in identifying significant trends, their comparative suitability, the observed similarities and differences in trends among the companies, and the accuracy of these models in predicting future trends

* 1. **Conclusion**

This chapter has comprehensively explored core time series concepts and delved into various time series forecasting models. The discussion highlighted the strengths and limitations of traditional statistical models and deep learning approaches in predicting stock prices. By outlining the knowledge gaps and identifying the research questions, this review has laid the groundwork for the methodological approach to be presented in the next chapter.

1. **Methodology** 
   1. **Introduction**

This chapter outlines the methodological approach used to forecast future trends in the stock prices of leading AI/ML companies: Alphabet, Meta, and Microsoft. It details the research process, including research design, approach, strategy, and data collection methods, ensuring the study's replicability and establishing the validity and reliability of its findings.

This research adopts a quantitative approach, focusing on analysing numerical data to generate knowledge and understanding (Burrell et al., 2017). Unlike qualitative research, quantitative research is well-suited for examining historical stock price data and identifying potential trends due to its emphasis on numerical precision and statistical analysis.

A deductive approach is employed, utilising established models like ARIMA and LSTM to validate their effectiveness in stock price prediction (Saunders et al., 2019). Additionally, a case study strategy is used, focusing on historical stock prices to provide in-depth insights (Yin, 2018).

The research relies on secondary data, using existing data collected by other sources. Historical stock price data will be obtained from Yahoo Finance, a reputable source of financial information. Unlike primary data, which can be time-consuming and expensive to collect, and may suffer from limitations in sample size or generalizability, secondary research offers access to extensive datasets that might be difficult or costly to collect independently, justified by the availability and comprehensiveness of the required historical stock price data (Johnston, 2014).

The collected data will be analysed using time series forecasting models, including ARIMA and LSTM. This chapter will provide a detailed explanation of the data collection process, data preprocessing steps, and the analytical techniques used to forecast future stock price trends.

The remainder of the chapter is structured into several key sections. Following this introduction, the literature search and a summary of the literature review is presented. The chapter then discusses the theoretical aspects of primary and secondary data, followed by an explanation of the research instruments. A detailed overview of the secondary data utilised for the research is provided, after which the data analysis section is presented. Ethical considerations are subsequently addressed. Finally, the chapter concludes with a summary of the key methodological aspects of the research.

* 1. **Literature Search**

This chapter reviewed time series analysis for stock price prediction. We aimed to assess the effectiveness of various models, including traditional statistical and deep learning approaches, and identify their strengths and weaknesses. Research gaps were then pinpointed, and research questions formulated to address them.

Scholarly databases like ScienceDirect and IEEE Xplore were utilised for their collections in finance, computer science, and statistics. Google Scholar, with its extensive search capabilities, served as the primary search engine. Reviewing abstracts, introductions, and searching for keywords (time series analysis, stock price prediction, ARIMA, ARMA-GARCH, deep learning, RNN, LSTM, etc.) helped identify relevant articles (Appendix VII).

The search focused on recent, peer-reviewed journal articles and conference proceedings in English to ensure current and credible findings, incorporating the latest advancements in time series analysis and stock price prediction techniques.

* 1. **Summary of Literature Review**

The reviewed literature was categorised according to time series concepts, stationarity tests, and time series models (statistical vs. deep learning) employed for stock price prediction. We analysed their strengths and weaknesses in capturing trends and forecasting. Themes emerged regarding the effectiveness of deep learning models (LSTMs) in certain scenarios and the utility of traditional statistical models (ARIMA) in others (Appendix VII). This understanding informs our conceptual framework by emphasising that feature selection, prediction windows, and model adjustments are crucial when choosing a model for stock price prediction.

* 1. **Primary data**

Primary data refers to information gathered directly by researchers through various methods like surveys, interviews, experiments, or observations. It is collected with a specific research question in mind, acting as a customized lens to explore a phenomenon. Unlike secondary data, which might be pre-existing or not perfectly aligned with the study's needs, primary data offers the distinct advantage of being tailored to the researcher's specific inquiry. This direct approach allows researchers to design and control the data-gathering process, ensuring its relevance in relation to the research objectives (Creswell, 2023).

Collecting primary data becomes necessary when existing information is scarce, outdated, or does not quite match with the research questions. By gathering primary data, researchers can directly test specific hypotheses and generate unique insights that might not be possible with secondary data alone (Fisher, 2010). This approach is particularly valuable in exploratory research, where understanding new or emerging trends necessitates fresh data that captures the current context's nuances (Bell, 2022), which is not the main aim of the present study.

* 1. **Secondary data**

The primary objective of utilising secondary data in research is to leverage existing information to address new research questions or validate findings without the need for primary data collection. This approach is particularly valuable in financial forecasting, where extensive historical datasets are often required. According to Bell et al. (2022), secondary data can provide a rich source of information that may otherwise be inaccessible due to constraints in time, cost, or logistical feasibility. By utilising secondary data, researchers can capitalise on previously gathered information to conduct comprehensive analyses, thereby enhancing the robustness and validity of their findings.

Secondary data can be classified into qualitative and quantitative types. Qualitative data is descriptive and often involves thematic analysis to uncover patterns and insights from non-numeric information such as interviews, texts, or visual media (Creswell et al., 2022). In contrast, quantitative data is numerical and is analysed using statistical methods to identify trends, correlations, or causal relationships. This research employs quantitative secondary data, specifically historical stock price data, as it aligns with the objective of forecasting future stock price trends of AI and ML companies. Quantitative data is advantageous for its precision and the ability to apply mathematical models to predict future outcomes (Fisher et al., 2010). The table below highlights the general strengths and limitations of using secondary data in research (Bell et al., 2022; Johnston, 2014).

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Provides access to extensive datasets, facilitating comprehensive analysis. | May lack specific data tailored to the research needs. |
| Saves time and reduces costs compared to primary data collection. | Limited control over data collection process and potential biases inherent in original data |
| Enhances robustness and validity by leveraging previously gathered information | Potential issues with data completeness or gaps in historical data |
| Suitable for financial forecasting, where historical data is essential | Secondary data may be outdated or not align perfectly with the research timeline |

Table : Advantageous and disadvantageous of secondary data

The type of secondary data utilised in this research includes historical stock price data. This data is sourced from Yahoo Finance. This platform provides extensive historical datasets for publicly traded companies, ensuring the reliability and accuracy required for financial forecasting. The selection of Yahoo Finance as the data source is justified by its extensive coverage and the granularity of the data it offers.

* 1. **Research instrument design**

This study's research instrument aimed to collect historical stock price data for leading AI/ML companies (Alphabet, Meta Platforms, Microsoft). This data is essential for analysing and forecasting future stock prices.

Key design considerations included:

* Relevance: Ensuring data directly supports price analysis and forecasting.
* Accuracy: Selecting Yahoo Finance for its established reputation and extensive datasets.

The instrument focused on historical adjusted closing prices as numerical values for a defined date range. These variables were chosen for their relevance to financial forecasting and their impact on time series model accuracy.

Data was recorded in a standardised CSV format, ensuring consistency for analysis. Adjusted closing prices and dates were logged numerically and consistently.

* 1. **Overview of secondary data used for analysis**

The dataset spans from 1st January 2015 to the end of May 2024, encompassing approximately nine years and five months, which translates to 2,368 working days. For the purposes of model evaluation and analysis, data from the beginning of the period until the end of February 2024 was utilised, resulting in 2,305 records. The remaining 63 working days, from 1st March 2024 to the end of May 2024, were reserved for validating the forecasting models.

The primary variables extracted from the dataset are the date and the adjusted close price. The adjusted close price is chosen due to its adjustment for corporate actions, providing a more accurate reflection of the stock's value over time.

The data collection process was straightforward: the historical stock prices were downloaded directly from Yahoo Finance on the 1st of June 2024, following the last business day of May. This ensures that the dataset is up-to-date.

The data collection period begins in 2015, a strategic choice reflecting the significant developments in AI and deep learning during that time. Around 2015, Google’s advancements in deep learning began to gain substantial attention, marking a period of rapid innovation in the field (LeCun et al., 2015). Similarly, Facebook intensified its efforts in AI research, establishing the Facebook AI Research (FAIR) lab in 2013 and achieving notable progress by 2015 (BBC, 2015). This period is therefore pivotal for analysing the stock performance of companies heavily invested in AI and ML.

To provide a comprehensive understanding of the dataset, the following figures illustrate the trends in the adjusted close prices for Alphabet, Meta Platforms, and Microsoft from January 2015 to May 2024:

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure : Historical stock prices of Alphabet, META, and Microsoft

* 1. **Data analysis**

The data analysis process for this research involves several meticulous steps, each designed to ensure the robustness and accuracy of the findings. These steps are applied independently to the stock prices of Alphabet, Meta, and Microsoft.

Figure : Research Methodology

* **Data Preprocessing and Outlier Detection**

The initial stage focuses on data preprocessing and outlier detection. This stage ensures the quality and suitability of the data for subsequent analysis. Data is inspected for missing values and anomalies using descriptive statistics and visual inspection (Aggarwal, 2015). To detect outliers, statistical distribution methods were employed. This approach leverages the best-fitted distribution, according to statistical analysis to define data points deviating from the norm (Aggarwal, 2017).

* **Exploratory Data Analysis (EDA)**

During the EDA phase, we examine potential patterns and trends within the dataset. This includes preliminary tests using the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots to detect any inherent patterns (Hyndman et al., 2021; Brockwell, 2016; Dégerine et al., 2003). Stationarity tests, specifically the Augmented Dickey-Fuller (ADF) test (Dickey et al., 1979) and the Ljung-Box test (Ljung et al., 1978), are conducted to assess whether the time series data is stationary. If non-stationary, differencing is applied to achieve stationarity. After that, further analysis using ACF, PACF, and the Ljung-Box test is conducted to determine any seasonality within the dataset.

Finally, a simple moving average is overlaid on the original data to smooth out volatility and identify underlying trends. This helps in assessing trends, seasonality and other temporal dependencies in the data (Box et al., 2015). Determining the presence of seasonal patterns is vital for deciding whether to use seasonal models for forecasting.

* **Data Splitting**

To train and evaluate models, datasets are commonly divided into training and testing subsets, usually in an 80:20 proportion. This method ensures that a significant portion of the data is used for training, while a separate part is kept for testing the models' predictive performance. This division facilitates a thorough assessment of how well the models perform on new, unseen data (Goodfellow et al., 2016).

* **Model Implementation**

To evaluate the performance of advanced forecasting models, simple and exponential moving averages are utilised as benchmark models. These baselines serve as reference points, allowing for the comparison of how effective the more complex models are in predicting outcomes (Makridakis et al., 2008).

Both statistical and deep learning models are employed for forecasting. These include:

* Statistical Models: Exponential Smoothing, ARIMA, and ARIMA-GARCH.
* Deep Learning Models: Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU).

The accuracy and reliability of each model are assessed through training and evaluation using metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). These metrics serve as benchmarks to gauge how well the models perform in forecasting tasks (Chatfield et al., 2018).

To enhance model performance, a tuning algorithm is implemented to optimize the parameters of models. This step involves iterative adjustments to find the best configurations that minimize forecasting errors (Box et al., 2015).

* **Forecasting and Validation**

The best-performing models from each category (statistical and deep learning) are selected for mid-term future forecasting. Various time windows are analysed to assess the performance of the deep learning models, ensuring a comprehensive evaluation of their predictive capabilities (Wang et al., 2018).

The final step involves comparing the forecasted values against the actual, unseen values. This comparison, based on RMSE and MAE metrics, identifies the model with the best prediction accuracy. The results are then used to draw conclusions about the most effective model for stock price forecasting (Montgomery et al., 2015).

* 1. **Ethics**

This research adheres to rigorous ethical guidelines to ensure its integrity and ethical conduct. As the study utilises secondary data, specifically historical stock prices from Yahoo Finance, there is no direct interaction with human participants. This minimises ethical risks associated with privacy and consent. However, it remains crucial to acknowledge the data source accurately and ensure its appropriate use within the research scope (Bell et al., 2022; Fisher, 2010). All analyses and interpretations are conducted with transparency and objectivity, upholding academic integrity and avoiding any data manipulation or misrepresentation. Additionally, an Ethos application has been completed, and the supporting documents is included in Appendix II.

* 1. **Summary**

This chapter outlines the methodology for forecasting stock prices of Alphabet, Meta, and Microsoft. It details a quantitative approach using established models (ARIMA, LSTM) and a case study strategy with historical data (2015-May 2024) from Yahoo Finance. The analysis involves data preprocessing, exploratory data analysis, model implementation and finally forecasting and validation. Ethical considerations are addressed through an Ethos application (details in appendices). The next chapter dives into data analysis for insights and conclusions.

1. **Data analysis and discussion**
   1. **Introduction**

This chapter details the systematic approach taken to analyse the stock price data for Alphabet, Meta, and Microsoft. The analysis begins with data preprocessing and outlier detection, followed by exploratory data analysis to reveal inherent patterns. ADF and Ljung-Box tests assess data stationarity. ACF and PACF plots identify patterns, and simple moving averages smooth volatility to expose trends. The data, covering adjusted closing prices from January 1st, 2015, to April 30th, 2024, is then split for training and testing models.

Forecasting models, including simple and exponential moving averages as baselines, and advanced models (Exponential Smoothing, ARIMA, ARIMA-GARCH, RNN, LSTM, GRU), are implemented in Python (Appendix VI). Model accuracy is evaluated using RMSE and MAE metrics, with parameter tuning to improve performance.

Finally, mid-term forecasting is conducted from March to May 2024, comparing forecasted values to actual prices to identify the most effective model for each company. These results guide conclusions on the best models for this research context.

* 1. **Data Preprocessing and Outlier Detection**
     1. **Data preprocessing**

The data preprocessing stage plays an essential role in ensuring the quality of the data. This phase involved a series of checks to confirm data integrity.

* **Missing** **Values**

An inspection of the data revealed no missing values (null or empty entries) within the adjusted closing stock price data for all three companies from January 1st, 2015, to April 30th, 2024.

* **Descriptive Statistics**

Descriptive statistics were calculated for each company's stock price data using excel. These initial observations did not reveal any significant anomalies or inconsistencies within the data.

|  |  |  |  |
| --- | --- | --- | --- |
| **Statistics** | **Alphabet** | **Meta** | **Microsoft** |
| Mean | 74.87707653 | 194.199443 | 160.5447804 |
| Standard Error | 0.780019228 | 1.704249905 | 2.184334189 |
| Median | 60.275002 | 177.761383 | 130.450317 |
| Mode | 26.834499 | 137.274353 | 56.675861 |
| Standard Deviation | 37.44904725 | 81.82174611 | 104.8707921 |
| Sample Variance | 1402.43114 | 6694.798137 | 10997.88303 |
| Kurtosis | -1.057007205 | 0.125018762 | -1.075224714 |
| Skewness | 0.5628959 | 0.80461222 | 0.486417161 |
| Range | 130.279926 | 416.158486 | 384.194637 |
| Minimum | 24.56007 | 73.971519 | 34.823277 |
| Maximum | 154.839996 | 490.130005 | 419.017914 |
| Sum | 172591.6614 | 447629.7161 | 370055.7188 |
| Count | 2305 | 2305 | 2305 |

Table : Descriptive Statistics for stock prices of Alphabet, Meta, and Microsoft

* **Date Inspection**

A visual inspection of the date data confirmed no inconsistencies or unexpected entries. This ensures the temporal integrity of the time series data.

* + 1. **Outlier Detection**

Outliers were detected using the Python Fitter library, which evaluates over 16 statistical distributions (e.g., Normal, Exponential, Weibull\_Min) to find the best fit. Data points outside the threshold of the best-fit distribution were identified as outliers (Appendix III). Below are diagrams and results showing the best-fitting distributions, their statistics, and outliers for each stock:

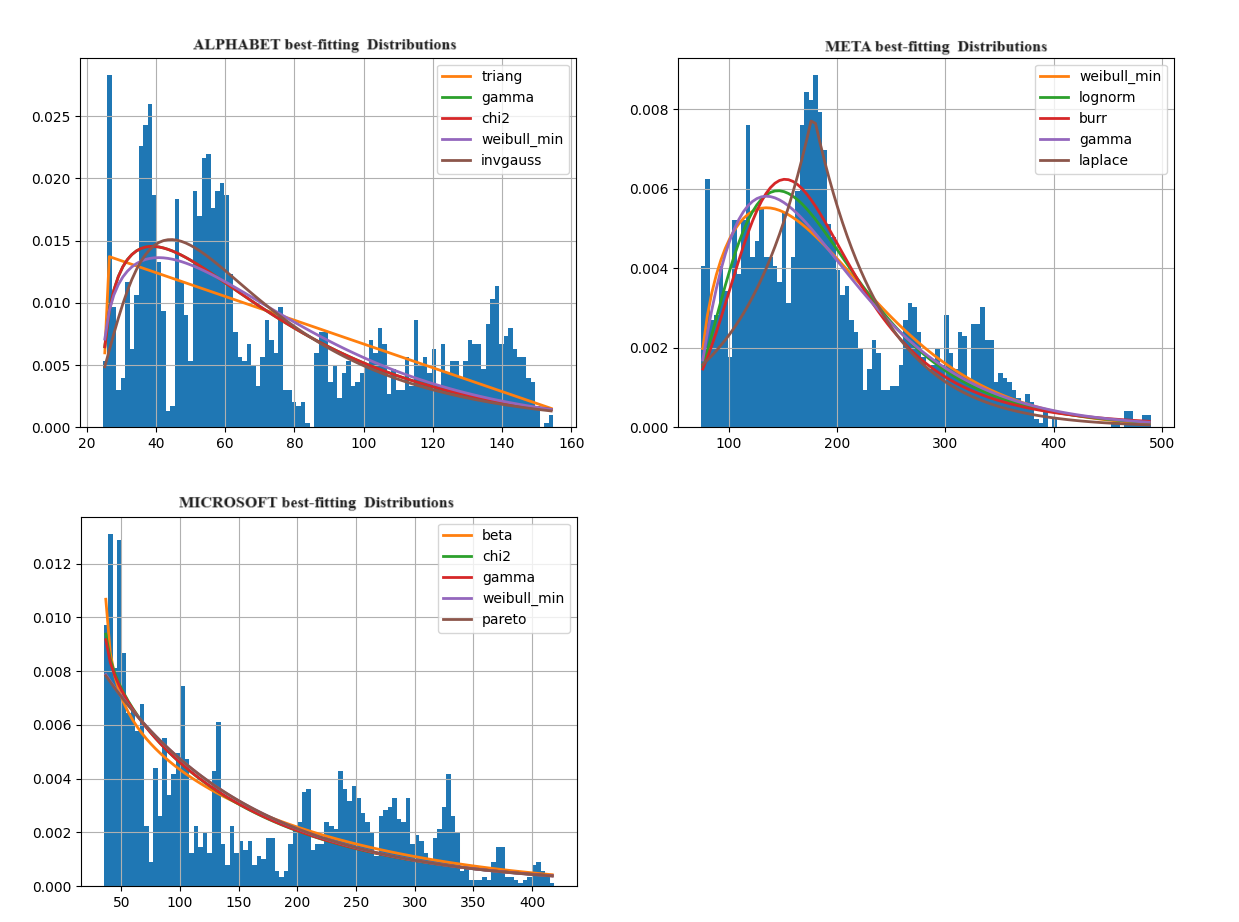


Figure : Best-fitting Distributions Plot of Alphabet, Meta, and Microsoft

|  |  |  |  |
| --- | --- | --- | --- |
| **Statistics** | **Alphabet** | **Meta** | **Microsoft** |
| Best-Fitting Distribution | Triangular | Weibull\_Min | Beta |
| Distribution Parameters | 'c': 0.013965398369329092, 'loc': 24.325991499760093, 'scale': 145.6463625529612 | 'c': 1.476317889619243, 'loc': 72.3352781557903, 'scale': 34.36230849716202 | 'a': 0.8267846118714879, 'b': 4.442673066892139, 'loc': 34.82327699999999, 'scale': 782.2503021914882 |
| Threshold Percentile | 0.95 | 0.95 | 0.95 |
| No of outliers | 192 | 79 | 30 |
| Min outliers | 137.653503 | 355.073273 | 392.43512 |
| Max outliers | 154.839996 | 490.130005 | 419.017914 |

Table : Best-fitting distribution and outlier statistics for Alphabet, Meta, and Microsoft stocks

Upon inspection, Alphabet's outliers correspond to sharp growth from mid-2021 to early 2022 and from late 2023 onwards. Meta's outliers appear in Q3 2021 and late 2023 onwards. Microsoft's outliers are noticeable from early 2024.

Despite this, retaining outliers is justified due to the following factors:

* **Significant Events**

All three stocks showed significant growth with peaks in 2021-2022 and late 2023, likely influenced by major events. Retaining these outliers captures the market's behaviour accurately.

* **Market Volatility**

Stock prices are inherently volatile, and these fluctuations are essential for understanding stock behaviour.

* **Robust Modelling**

The models used are robust and can effectively handle the volatility introduced by outliers, capturing extreme values reliably.

Retaining outliers and using robust forecasting models provide a comprehensive view of stock prices, accounting for both routine fluctuations and significant market events.

* 1. **Exploratory Data Analysis (EDA)**

The exploratory data analysis (EDA) phase examined the data of all three stocks to identify patterns such as seasonality and trends. This was achieved using the Augmented Dickey-Fuller (ADF) and Ljung-Box tests, as well as ACF and PACF plots. To conduct these tests, we utilised Python libraries including ‘statsmodels.tsa.stattools’, ‘statsmodels.stats.diagnostic’, and ‘matplotlib.pyplot’. Finally, visual inspection of the original trends overlaid with a simple moving average confirmed the presence of patterns.

* + 1. **EDA for Alphabet**

The ADF test results (statistic: -0.2321, p-value: 0.9346) indicated that the series was non-stationary, as the p-value was significantly higher than the usual significance level of 0.05.

The Ljung-Box test was performed to assess the presence of autocorrelation at various lags. The results consistently showed significant p-values (p < 0.05) across multiple lags, suggesting that the data exhibited autocorrelation, further confirming non-stationarity.

To further validate these findings, ACF and PACF plots were examined. Our aim was to identify spikes at regular intervals. Based on our daily data, the lags were considered equal to 252.

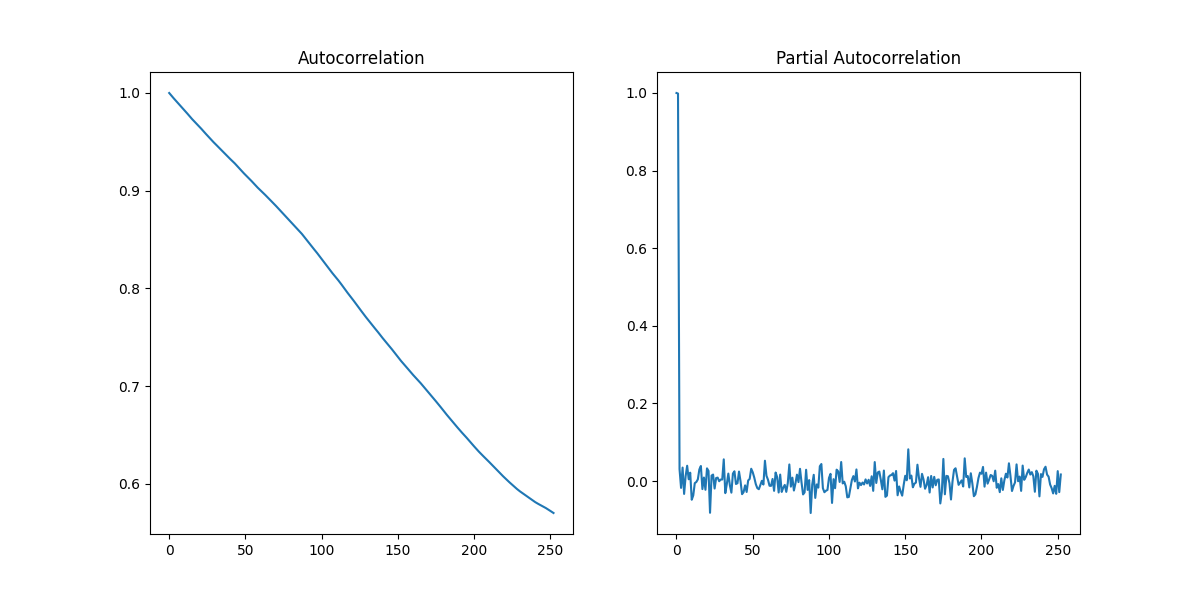
****

Figure : Original ACF and PACF plots of Alphabet

The ACF plot showed a slow decay, indicating non-stationarity and a strong long-range autocorrelation, but no clear seasonality. The PACF plot had a strong initial spike at lag 1, suggesting strong short-term correlation, with most subsequent lags falling within significance bounds, implying a weakened relationship after the first lag. Overall, no clear signs of seasonality were evident in the ACF and PACF plots.

Given these findings, differencing was applied to the data to unmask presence of seasonality. Subsequent analyses with ADF and Ljung-Box tests, ACF and PACF plots were conducted to identify any seasonality within the dataset.

The ADF test results indicated that after differencing, the time series was now stationary. The ADF statistic was significantly negative (-11.7524), and the p-value was extremely low (1.2003e-21), much lower than the usual significance level of 0.05. This confirmed that the null hypothesis of non-stationarity could be rejected, indicating that the differenced series was stationary.

In the Ljung-Box test statistics, lower lags showed p-values higher than the significance level (0.05), indicating no significant autocorrelation at these lags. However, as the lags increased, the p-values became extremely low (p < 0.05), indicating significant autocorrelation at these higher lags (Appendix IV). This suggested that the differenced series still retained some long-term dependencies. Further analysis with ACF and PACF could fully address these dependencies.

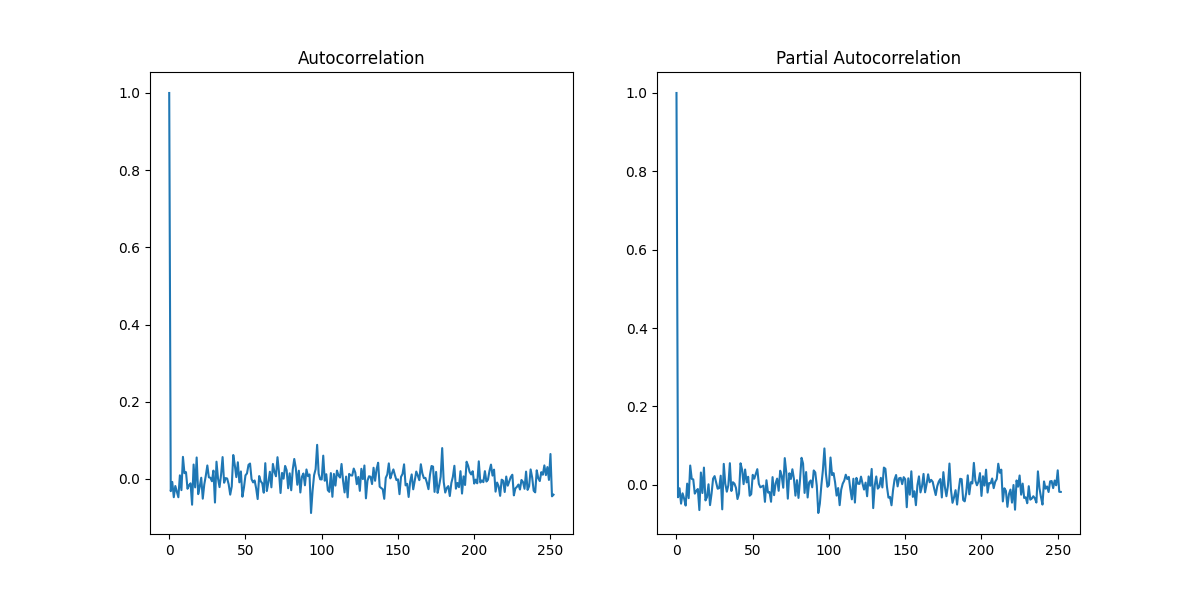
****

Figure : ACF and PACF plots of Alphabet after first-order differencing

The ACF showed a strong initial spike at lag 1 due to first differencing, then dropped to near zero, indicating white noise. The PACF also showed no significant partial autocorrelation beyond the first lag. The ADF and Ljung-Box tests, along with the ACF and PACF plots, suggested that the series was now stationary. There were no visible periodic spikes or regular patterns, indicating a lack of strong seasonality in the Alphabet data.

Since there was no detectable seasonality, we focused on identifying trends using the original stock price overlaid with a simple moving average of 50 days (selected based on the best metric results of RMSE and MAE).

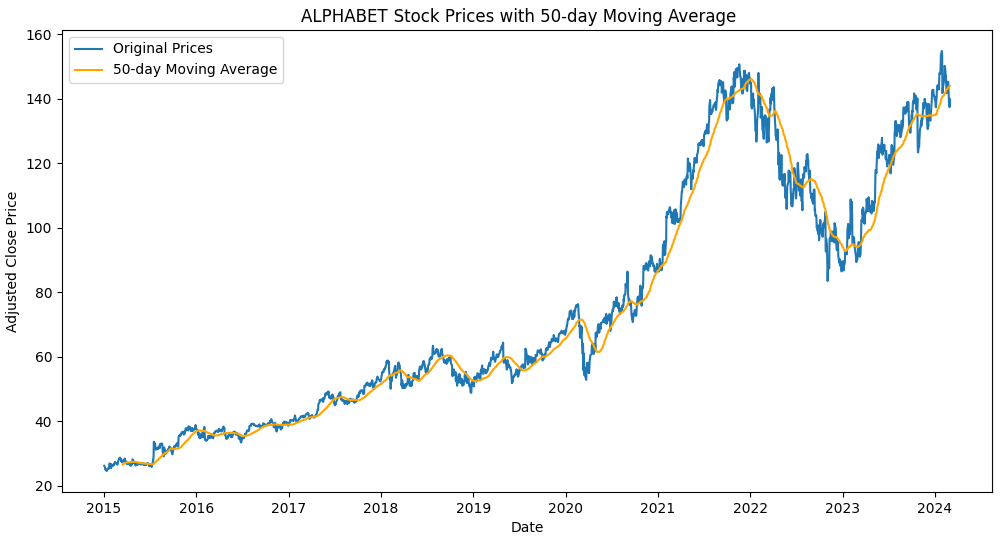
****

Figure : Alphabet original trends overlaid by 50-day simple moving average

Visual inspection of original and 50-day moving average trends revealed a clear upward trend in Alphabet’s stock price from 2015 to 2024. This long-term trend indicates that the stock price has generally increased over this period.

* + 1. **EDA for Meta**

The ADF test (statistic: 0.0463, p-value: 0.9622) indicated that the series was non-stationary (a high p-value suggests this). The Ljung-Box test confirmed this non-stationarity, as it revealed statistically significant autocorrelation (p-values less than 0.05) across multiple lags. To strengthen the evidence for non-stationarity, ACF and PACF plots were examined.

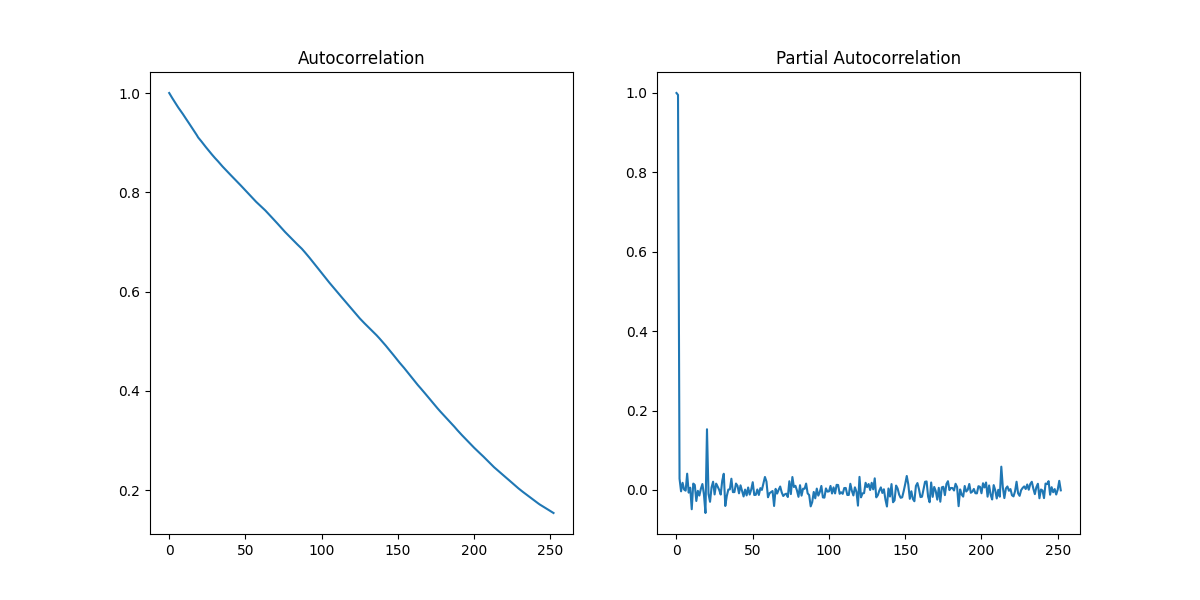


Figure : Original ACF and PACF plots of META

META's plots suggest characteristics of non-stationarity. The ACF plot exhibits strong autocorrelation across most lags, indicating no recurring patterns that might suggest strong seasonality. Similarly, the PACF plot shows a pronounced spike at lag one, with most other lags close to zero, although a few outliers exist. Overall, there's no clear evidence of seasonality.

Based on these findings, first-order differencing was performed. The ADF test results (statistic: -8.6973, p-value: 3.8947e-14) strongly suggest that the data is now stationary. While the Ljung-Box test confirms this, it revealed some remaining autocorrelation at higher lags (Appendix IV). To investigate these remaining dependencies in the differenced data, ACF and PACF plots were examined.

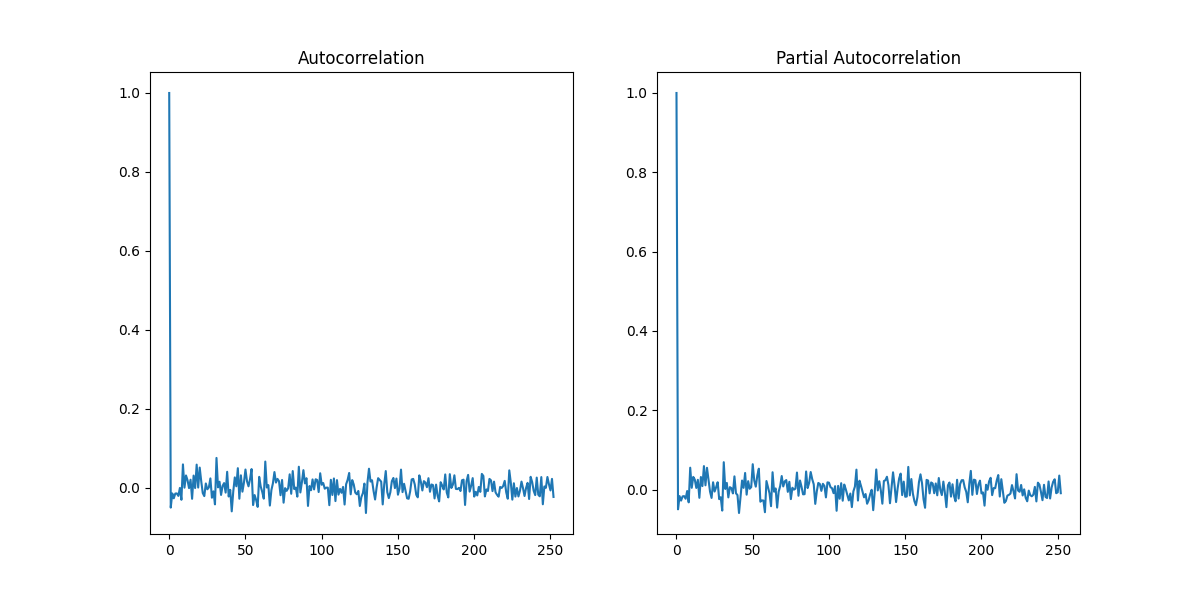
****

Figure : ACF and PACF plots of META after first-order differencing

The ACF and PACF plots confirmed stationarity, with the ACF showing a large initial spike due to differencing and subsequent values near zero and the PACF showing no significant autocorrelations beyond the first lag. No clear seasonal patterns were observed in the META data, and the lack of regular spikes or significant autocorrelation at specific lags further supports this. We therefore focused on trend detection using the original price data with a 210-day moving average (chosen based on RMSE and MAE metrics).

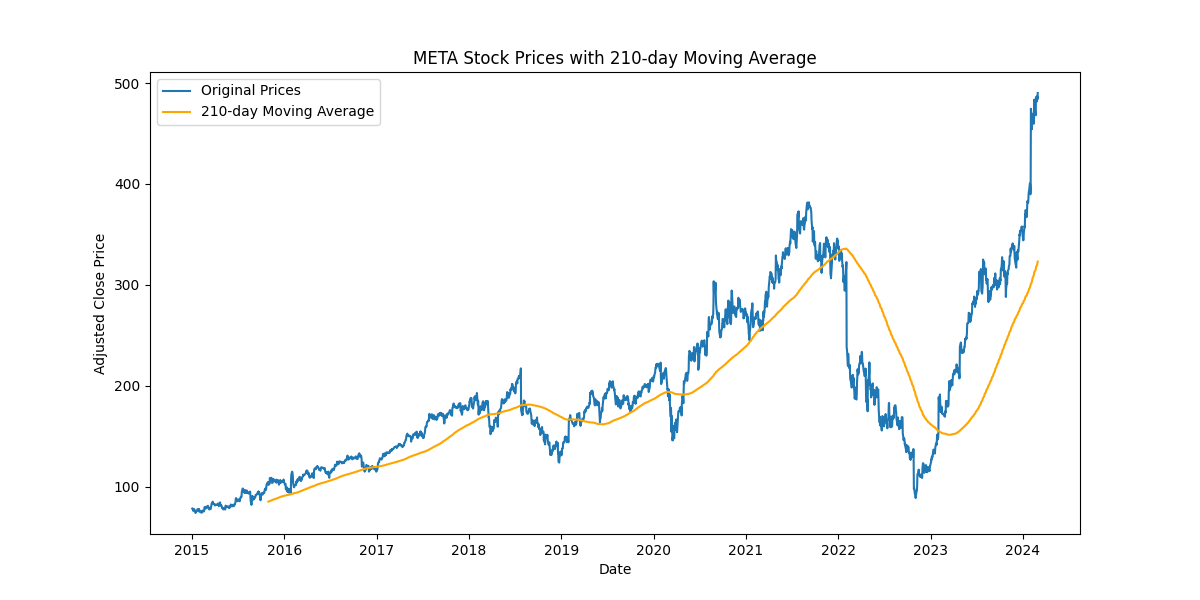
****

Figure : Alphabet original trends overlaid by 210-day simple moving average

Despite a sharp price drop in 2022, META's stock exhibits a clear upward trend from 2015 to 2024, confirmed by both the original price and the 210-day moving average. This long-term trend reflects a general increase over this period.

* + 1. **EDA for Microsoft**

Analysis of stationarity in the data suggests it's likely non-stationary. The ADF test statistic (1.1992) with a high p-value (0.9959) indicates this. Furthermore, the Ljung-Box test results show statistically significant autocorrelation (p-values of 0 across all lags), reinforcing the evidence for non-stationarity. Visual inspection of ACF and PACF plots can further support the evidence for non-stationarity.

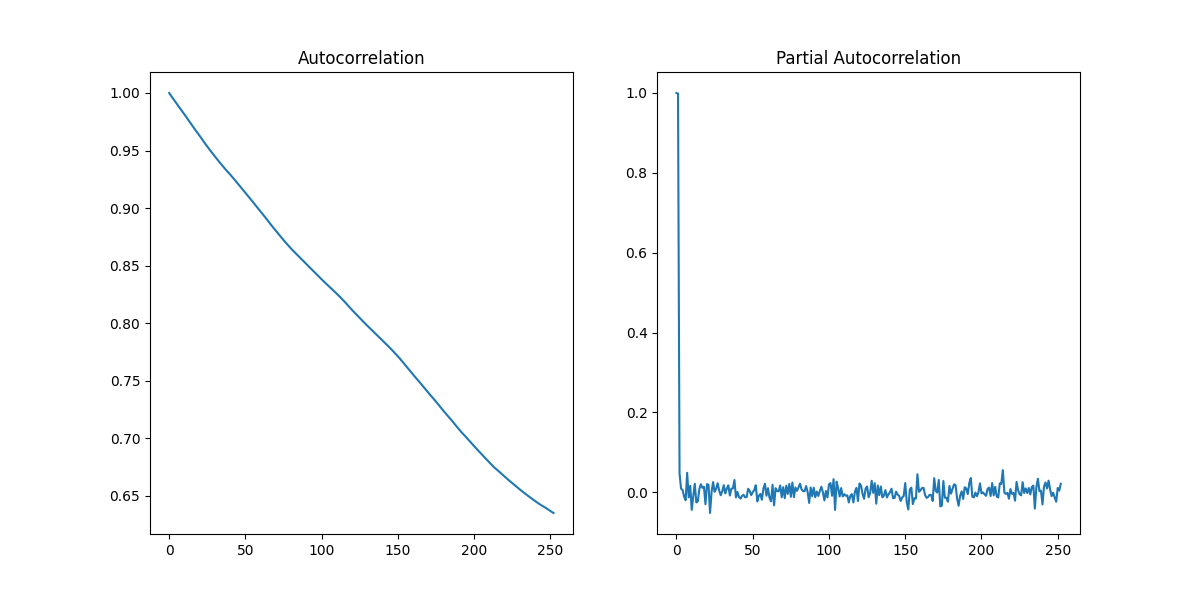
****

Figure : Original ACF and PACF plots of Microsoft

Examination of Microsoft's data reveals characteristics consistent with non-stationarity. The ACF plot exhibits strong autocorrelation across most lags, suggesting an absence of recurring patterns that would typically indicate seasonality. This observation is further supported by the PACF plot, which shows a pronounced spike at lag one but minimal autocorrelation at other lags, with a few exceptions.

The ADF test statistic of -10.39 and a highly significant p-value (1.9279e-18) strongly suggest the data is now stationary after first-order differencing. This is contradictory to the Ljung-Box test results which shows remaining autocorrelation (Appendix IV). To further investigate potential seasonality, we examined the ACF and PACF plots.

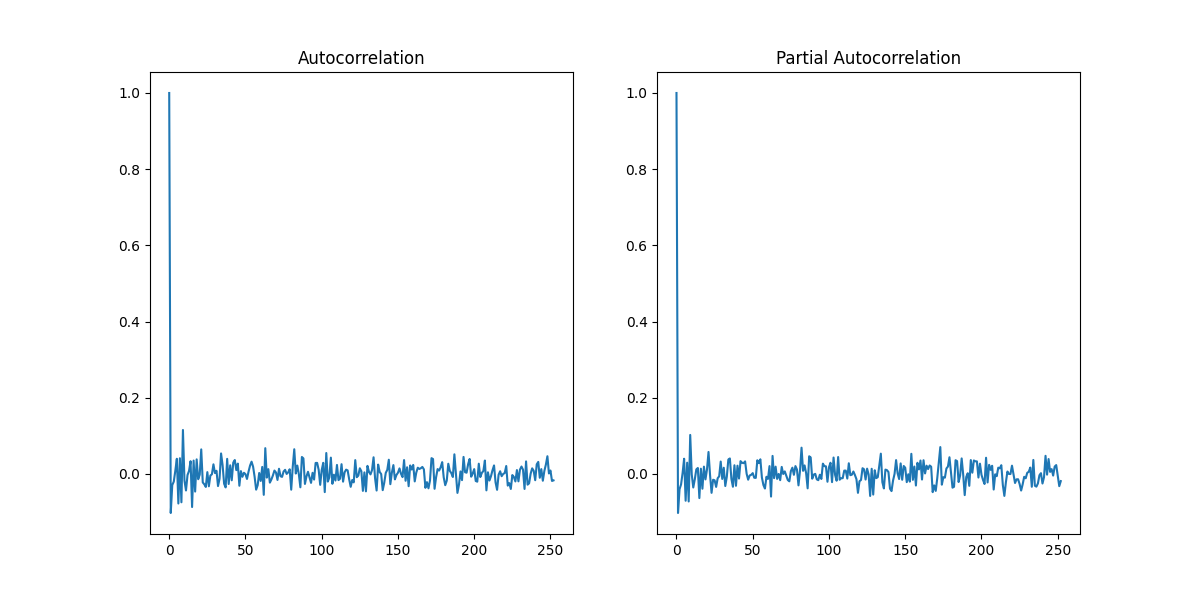


Figure : ACF and PACF plots of Microsoft after first-order differencing

The ACF and PACF plots suggest that the first-order differenced series for Microsoft is stationary, as indicated by the initial spike at lag 1 followed by autocorrelations near zero. This implies that the data no longer exhibits significant autocorrelation or clear seasonal patterns. Consequently, we focused on trend detection using the original price data with a 90-day moving average (selected based on RMSE and MAE metrics).

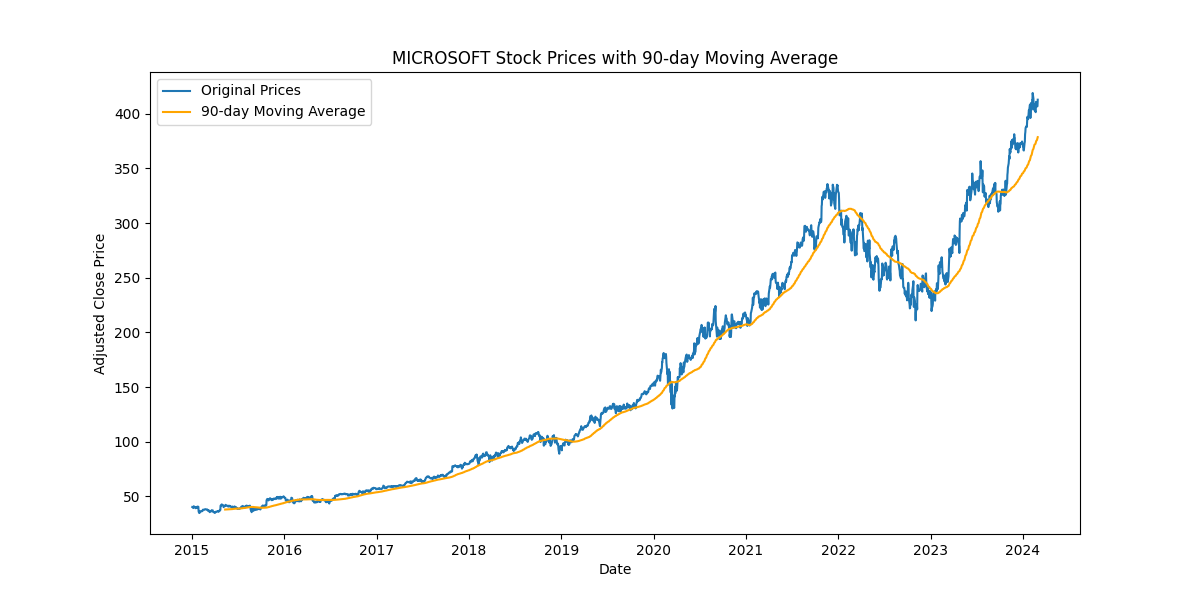
****

Figure : Microsoft original trends overlaid by 90-day simple moving average

Examining the period from 2015 to 2024, Microsoft's stock price reveals a clear upward trajectory, despite a price dip in 2022. This long-term trend is evident in both the original price data and the 90-day moving average, indicating a general increase over time.

* 1. **Data Splitting**

To train and evaluate the performance of our forecasting models, we adopted an 80/20 train-test split. This means that 80% of the data is utilised for training the models, and the remaining 20% is reserved for testing and evaluating their performance.

The training data comprises 1,844 daily closing prices for each stock, spanning the period from 1st January 2015 to 28th April 2022. The test data encompasses daily closing prices from 29th April 2022 to 29th February 2024.

* 1. **Model Implementation**

This stage involves fitting a series of models to the data to determine which one best captures the trends in stock prices. First, we fit two simple statistical models as baselines: Simple Moving Average (SMA) and Exponential Moving Average (EMA). These baselines are implemented using a naive approach. It means the last MA value from the training period is then used as a constant forecast for the entire test period. This naive approach assumes that the future stock prices will remain at the level of the last SMA or EMA value calculated from the training set.

Following the baseline models, we fit a range of more advanced models:

* Statistical Models:
* Exponential Smoothing
* ARIMA (Autoregressive Integrated Moving Average)
* ARIMA-GARCH (combines ARIMA with GARCH for volatility modelling)
* Machine Learning Models:
* Recurrent Neural Network (RNN)
* Long Short-Term Memory (LSTM)
* Gated Recurrent Unit (GRU)

**Model Selection Considerations**

Our decisions regarding model selection are informed by the findings of the exploratory data analysis (EDA) stage. Notably, the absence of significant seasonality allows us to set the seasonality parameter to "false" during model training. Conversely, the observed presence of volatility in the stock prices, evidenced by the results of distribution fitting and identified outliers, justifies the inclusion of models equipped to handle this phenomenon, such as ARIMA-GARCH.

**Model Tuning**

* Statistical Models

For these models, manual parameter tuning will be conducted across various scenarios to identify the configuration with the lowest Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). This process will involve trying different combinations of parameters and selecting the one that produces the best results.

* Machine Learning Models

To optimize these models, we will leverage an automated hyperparameter tuning function with 50 iterations. This function will automatically explore different combinations of hyperparameters and select the best set that minimizes the error metrics (MAE and RMSE).

**Python Libraries Used**

We will utilise the following Python libraries to implement and forecast the models:

|  |  |
| --- | --- |
| **Main Library** | **Imported Class** |
| ‘statsmodels.tsa.holtwinters’ | ‘ExponentialSmoothing’ |
| ‘statsmodels.tsa.arima.model’ | ‘ARIMA’ |
| ‘arch’ | ‘arch\_model’ |
| ‘tensorflow.keras.models’ | ‘Sequential’ |
| ‘tensorflow.keras.layers’ | ‘SimpleRNN’, ‘Dense’, ‘Dropout’, ‘LSTM’, ‘GRU’ |
| ‘sklearn.preprocessing’ | ‘MinMaxScaler’ |
| ‘sklearn.model\_selection’ | ‘ParameterSampler’ |
| ‘sklearn.metrics’ | ‘mean\_squared\_error’, ‘mean\_absolute\_error’ |
| ‘scipy.stats’ | ‘randint’, ‘uniform’ |

Table : Python Libraries utilised for model implementation

* + 1. **Alphabet Stock Price Forecast**

Before presenting the comparative results of the forecasting models, we first outline the hyperparameters obtained through the tuning process.

|  |  |
| --- | --- |
| **Forecasting Model** | **Tuned Parameters** |
| SMA | 50-day moving average |
| EMA | 50-day moving average |
| Exponential Smoothing | Seasonality = False |
| ARIMA | order (2, 2, 2) |
| ARIMA-GARCH | ARIMA(2,2,2)-GARCH(2,2) |
| RNN | {'batch\_size': 15, 'dropout': 0.25171866634056334, 'epochs': 48, 'learning\_rate': 0.0030474457706320727, 'look\_back': 1, 'return\_sequences': False, 'units': 88} |
| LSTM | {'batch\_size': 10, 'dropout': 0.1311217605210499, 'epochs': 35, 'learning\_rate': 0.009105956549136991, 'look\_back': 1, 'return\_sequences': False, 'units': 48} |
| GRU | {'batch\_size': 24, 'dropout': 0.0010595028648908156, 'epochs': 91, 'learning\_rate': 0.0065272063812476955, 'look\_back': 4, 'return\_sequences': False, 'units': 34} |

Table : Tuned Parameters of Forecasting Models for Alphabet Stock Price Prediction

The corresponding forecast charts are presented below:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Figure : Alphabet Baseline and Forecasting Models’ Charts

Based on the best-tuned hyperparameters, the following table presents a comparison of the forecasts for Alphabet stock prices:

|  |  |  |
| --- | --- | --- |
| **Models** | **RMSE** | **MAE** |
| SMA | 23.3930 | 19.3182 |
| EMA | 22.9918 | 19.0076 |
| Exponential Smoothing | 19.3459 | 14.8952 |
| ARIMA | 19.3427 | 14.8919 |
| ARIMA-GARCH | 17.7210 | 13.4031 |
| RNN | 2.6674 | 2.0425 |
| LSTM | 2.5260 | 1.9194 |
| GRU | 2.5221 | 1.9002 |

Table : Comparative Results of Forecasting Models and Performance Metrics for Alphabet

**Interpretation of the Results**

Among the statistical models, ARIMA-GARCH outperforms the SMA and EMA baseline models in forecasting Alphabet's stock price, as evidenced by its lower RMSE and MAE. The ARIMA-GARCH model also surpasses ARIMA due to its superior ability to capture volatility and temporal dynamics in the data.

However, advanced recurrent neural network models demonstrate substantially lower RMSE and MAE compared to statistical models, indicating higher accuracy in predicting the time series. This suggests that these models are better suited for capturing complex, non-linear relationships in the data.

Among these advanced models, the RNN achieved the lowest validation loss (5.9612 compared to 6.3194 for LSTM and 6.0293 for GRU). Nevertheless, the GRU model shows the most promising results in terms of RMSE and MAE, outperforming the baseline models and offering the best overall performance by effectively balancing complexity and predictive power.

* + 1. **META Stock Price Forecast**

we first outline the hyperparameters obtained through the tuning process.

|  |  |
| --- | --- |
| **Forecasting Model** | **Tuned Parameters** |
| SMA | 90-day moving average |
| EMA | 90-day moving average |
| Exponential Smoothing | Seasonality = False |
| ARIMA | order (1, 2, 2) |
| ARIMA-GARCH | ARIMA(1,2,2)-GARCH(1,1) |
| RNN | {'batch\_size': 16, 'dropout': 0.05467804803565551, 'epochs': 23, 'learning\_rate': 0.007659192886149137, 'look\_back': 3, 'return\_sequences': False, 'units': 38} |
| LSTM | {'batch\_size': 19, 'dropout': 0.08034928178417022, 'epochs': 14, 'learning\_rate': 0.00596315278232027, 'look\_back': 1, 'return\_sequences': False, 'units': 98} |
| GRU | {'batch\_size': 24, 'dropout': 0.3997474663039183, 'epochs': 46, 'learning\_rate': 0.010744806220539603, 'look\_back': 5, 'return\_sequences': False, 'units': 33} |

Table : Tuned Parameters of Forecasting Models for META Stock Price Prediction

The corresponding forecast charts are presented below:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Figure : META Baseline and Forecasting Models’ Charts

Based on the best-tuned hyperparameters, the following table presents a comparison of the forecasts for META stock prices:

|  |  |  |
| --- | --- | --- |
| **Models** | **RMSE** | **MAE** |
| SMA | 94.7119 | 82.5677 |
| EMA | 93.7591 | 81.3003 |
| Exponential Smoothing | 87.3022 | 71.7266 |
| ARIMA | 87.0224 | 71.6127 |
| ARIMA-GARCH | 86.2684 | 72.3196 |
| RNN | 7.3929 | 4.7165 |
| LSTM | 7.5807 | 5.0083 |
| GRU | 7.3961 | 4.9242 |

Table : Comparative Results of Forecasting Models and Performance Metrics for META

**Interpretation of the Results**

Among the statistical models, ARIMA-GARCH demonstrates the lowest RMSE and MAE compared to the baseline and other statistical models in forecasting META’s stock price. However, advanced recurrent neural network models exhibit substantially lower RMSE and MAE compared to statistical models.

Among these neural network models, the RNN presented the lowest validation loss (48.8468 compared to 48.9812 for LSTM and 51.6435 for GRU), as well as the lowest RMSE and MAE, followed closely by the GRU. This indicates that RNN is the most effective model for predicting META’s stock price, with GRU also performing well.

* + 1. **Microsoft Stock Price Forecast**

we first outline the hyperparameters obtained through the tuning process.

|  |  |
| --- | --- |
| **Forecasting Model** | **Tuned Parameters** |
| SMA | 90-day moving average |
| EMA | 90-day moving average |
| Exponential Smoothing | Seasonality = False |
| ARIMA | order (2, 2, 2) |
| ARIMA-GARCH | ARIMA(2,2,2)-GARCH(2,2) |
| RNN | {'batch\_size': 23, 'dropout': 0.044295524080964044, 'epochs': 73, 'learning\_rate': 0.010301134772215333, 'look\_back': 5, 'return\_sequences': False, 'units': 40} |
| LSTM | {'batch\_size': 5, 'dropout': 0.04211393060968949, 'epochs': 69, 'learning\_rate': 0.0016780074660399514, 'look\_back': 9, 'return\_sequences': False, 'units': 42} |
| GRU | {'batch\_size': 16, 'dropout': 0.27228816701190384, 'epochs': 23, 'learning\_rate': 0.001997670302453216, 'look\_back': 1, 'return\_sequences': False, 'units': 96} |

Table : Tuned Parameters of Forecasting Models for Microsoft Stock Price Prediction

The corresponding forecast charts are presented below:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Figure : Microsoft Baseline and Forecasting Models’ Charts

Based on the best-tuned hyperparameters, the following table presents a comparison of the forecasts for Microsoft stock prices:

|  |  |  |
| --- | --- | --- |
| **Models** | **RMSE** | **MAE** |
| SMA | 53.3485 | 46.7654 |
| EMA | 53.5230 | 46.5023 |
| Exponential Smoothing | 42.2610 | 35.2332 |
| ARIMA | 42.5827 | 35.5366 |
| ARIMA-GARCH | 40.8237 | 34.1141 |
| RNN | 14.1964 | 12.9562 |
| LSTM | 10.0935 | 8.8062 |
| GRU | 5.1217 | 3.9073 |

Table : Comparative Results of Forecasting Models and Performance Metrics for Microsoft

**Interpretation of the Results**

Among the statistical models, ARIMA-GARCH presented the lowest RMSE and MAE in forecasting Microsoft’s stock price. Unlike the other two stocks, this time Exponential Smoothing outperformed ARIMA. However, advanced recurrent neural network models exhibited substantially lower RMSE and MAE compared to statistical models.

Among these neural network models, the GRU exhibited the lowest validation loss (27.8640 compared to 28.0490 for RNN and 27.8863 for LSTM), as well as the lowest RMSE and MAE. This indicates that GRU is the most effective model for predicting Microsoft’s stock price.

* 1. **Forecasting Performance for Mid-Term Unseen Data**

To investigate discrepancies with findings from other studies, the final forecasts generated by the GRU model (representing deep learning models) are compared to those from the ARIMA-GARCH model (the best performer among statistical methods). This selection is based on the RMSE and MAE scores presented in the previous section. Notably, the GRU model's score for Meta was marginally lower than the RNN model, but the GRU was chosen due to the negligible difference.

The forecast horizon considered a mid-term timeframe, encompassing three calendar months – from the beginning of March to the end of May 2024 – which translates to 63 business days. For each model, the best-tuned parameters identified during the modelling phase were employed.

The visual representation of ARIMA-GARCH and GRU model forecasts for Alphabet, META, and Microsoft is as follows:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Figure : Mid-term Prediction Charts of Alphabet, META, and Microsoft

Appendix V provides the full prediction table, while the following tables summarize the key findings:

|  |  |  |  |
| --- | --- | --- | --- |
| **Statistics** | **Real Data** | **ARIMA-GARCH Forecast** | **GRU Forecast** |
| Mean | 158.8312698 | 118.5202297 | 159.4705051 |
| Standard Error | 1.677293441 | 0.110022838 | 1.029708699 |
| Median | 157.66 | 118.5305587 | 162.09752 |
| Mode | 151.94 | #N/A | #N/A |
| Standard Deviation | 13.31310396 | 0.873279204 | 8.173059424 |
| Sample Variance | 177.2387371 | 0.762616568 | 66.79890035 |
| Kurtosis | -0.890379401 | -1.19576964 | -0.595870826 |
| Skewness | -0.234767037 | -0.003255059 | -0.780712469 |
| Range | 46.98 | 2.98102 | 27.72407 |
| Minimum | 132.56 | 117.0109123 | 140.77133 |
| Maximum | 179.54 | 119.9919323 | 168.4954 |
| Sum | 10006.37 | 7466.77447 | 10046.64182 |
| Count | 63 | 63 | 63 |

Table : Summary of real and predicted values for Alphabet

|  |  |  |  |
| --- | --- | --- | --- |
| **Statistics** | **Real Data** | **ARIMA-GARCH Forecast** | **GRU Forecast** |
| Mean | 485.6965094 | 212.6107345 | 484.6987952 |
| Standard Error | 2.875910666 | 0.15982517 | 0.173658043 |
| Median | 491.350006 | 212.5984275 | 484.08276 |
| Mode | #N/A | #N/A | #N/A |
| Standard Deviation | 22.82683324 | 1.268572955 | 1.378367986 |
| Sample Variance | 521.0643159 | 1.609277343 | 1.899898306 |
| Kurtosis | -0.079848557 | -1.193884676 | 2.847212346 |
| Skewness | -0.567019961 | 0.001294391 | 1.737867769 |
| Range | 97.170014 | 4.3322994 | 6.26981 |
| Minimum | 430.170013 | 210.4653215 | 483.61697 |
| Maximum | 527.340027 | 214.7976209 | 489.88678 |
| Sum | 30598.88009 | 13394.47627 | 30536.0241 |
| Count | 63 | 63 | 63 |

Table : Summary of real and predicted values for META

|  |  |  |  |
| --- | --- | --- | --- |
| **Statistics** | **Real Data** | **ARIMA-GARCH Forecast** | **GRU Forecast** |
| Mean | 415.2309 | 281.6476058 | 411.2951716 |
| Standard Error | 1.279222 | 0.288803756 | 0.040978789 |
| Median | 415.6703 | 281.6731341 | 411.18304 |
| Mode | #N/A | #N/A | #N/A |
| Standard Deviation | 10.15351 | 2.292308751 | 0.325259052 |
| Sample Variance | 103.0938 | 5.254679412 | 0.105793451 |
| Kurtosis | -0.48916 | -1.159064321 | 6.673300909 |
| Skewness | -0.48231 | -0.025395912 | 2.271451127 |
| Range | 41.89099 | 8.1259588 | 1.59491 |
| Minimum | 388.629 | 277.3374435 | 411.01602 |
| Maximum | 430.52 | 285.4634023 | 412.61093 |
| Sum | 26159.55 | 17743.79917 | 25911.59581 |
| Count | 63 | 63 | 63 |

Table : Summary of real and predicted values for Microsoft

The following table presents a prediction comparison for all three stock prices:

|  |  |  |  |
| --- | --- | --- | --- |
| **Models** | **Alphabet**  **(RMSE, MAE)** | **META**  **(RMSE, MAE)** | **Microsoft**  **(RMSE, MAE)** |
| ARIMA-GARCH | (41.6175, 39.7798) | (274.6141, 273.5990) | (133.7888, 133.4012) |
| GRU | (5.9912, 5.2584) | (22.3640, 18.0397) | (10.7544, 9.1580) |

Table : Comparative Table for Mid-term Forecasting of Alphabet, META, and Microsoft

A visual inspection of the predicted values, alongside the RMSE and MAE results, clearly indicates that the GRU model significantly outperforms the ARIMA-GARCH model. This is evident from the lower values for both RMSE and MAE for GRU. This finding aligns well with the research of Siami-Namini et al. (2018), who highlighted the superiority of LSTM (another recurrent neural network) compared to ARIMA models. This is further supported by the detailed statistics in Summary Tables, which demonstrate the GRU model's ability to predict values closer to the actual closing prices.

* 1. **Summary**

In the EDA section, our examination of the stock price datasets for Alphabet, Meta, and Microsoft aimed to identify seasonality or trends. We utilised ADF and Ljung-Box tests, along with ACF and PACF plots. Additionally, a visual inspection of the original price data, overlaid with a simple moving average, helped to corroborate the presence of these patterns. None of the stock price series exhibited strong seasonality. However, all three companies displayed upward trends over the 2015-2024 period.

In the next step, to train and evaluate the performance of our forecasting models, we adopted an 80/20 train-test split. While the ARIMA-GARCH model provided a significant improvement over traditional statistical models, advanced recurrent neural networks, particularly GRU (with the exception of RNN for META), offered the highest accuracy for forecasting all three stock prices.

In the subsequent phase, the top performer from each category of models, i.e. ARIMA-GARCH and GRU was chosen to forecast a mid-term unseen data. A visual inspection of the predicted values, alongside the RMSE and MAE results, indicated that the GRU model significantly outperformed the ARIMA-GARCH model across all three stocks.

In the final chapter, "Conclusion, Limitations and Recommendations," we will discuss our research aims, objectives, and questions based on the findings of this chapter, and draw our final conclusions.

1. **Conclusion, Limitations, and Recommendations**
   1. **Introduction**

This concluding chapter revisits the original research aims, objectives, and questions outlined in the first chapter. It then synthesizes the key findings from the data analysis presented in Chapter 4 to draw comprehensive conclusions about our investigation into stock price forecasting models. Finally, the chapter addresses the limitations of the current study and proposes valuable recommendations for future research directions in this field.

* 1. **Research Conclusion**

**RQ1: Identifying Long-Term Trends**

This section investigates the effectiveness of time series models in identifying significant long-term trends within the historical stock prices of Alphabet, Meta, and Microsoft. The analysis utilised historical adjusted closing stock prices for each company, spanning the period from 1st January 2015 to 30th April 2024. Exploratory Data Analysis (EDA) revealed no strong evidence of seasonality within the data. However, upward trends were identified in all three stock price series.

Informed by the existing literature (Selvin et al., 2017; Siami-Namini et al., 2018; Hu et al. 2020; Hyndman et al., 2021; Kim, 2022; Kobiela et al., 2022), a variety of time series models were implemented to capture long-term trends. Traditional statistical models including ARIMA-GARCH, alongside deep learning models like LSTM. To ensure optimal performance, hyperparameter tuning was conducted for each model. Importantly, all models were compared to baseline models: Simple Moving Average (SMA) and Exponential Moving Average (EMA) to assess their effectiveness.

The chosen models successfully captured significant long-term trends within the historical stock prices. The best performing model varied for each company. Nevertheless, all models demonstrated the capacity to identify the overall direction in the stock price movements.

The absence of significant seasonal trends suggests that the stock prices of these AI/ML companies are primarily influenced by long-term factors. These factors could include industry growth, company performance, or macroeconomic conditions (Wu et al., 2021). This research aligns with previous findings highlighting the strengths and limitations of traditional and deep learning models in time series forecasting (Hu et al., 2020; Siami-Namini et al., 2018; Kobiela et al., 2022).

**RQ2: Comparative Suitability of Time Series Models for Trend Analysis in AI Stocks**

This research question investigates the relative effectiveness of different time series models in capturing underlying trends within the stock prices of leading AI/ML companies. The analysis revealed that, in general, machine learning models (RNNs, LSTMs, GRUs) demonstrated greater suitability compared to traditional statistical models (ARIMA, ARIMA-GARCH, Exponential Smoothing) for trend analysis. This can be attributed to the inherent limitations of statistical models.

* **Statistical Model Limitations**

These models often rely on assumptions about the data, such as stationarity. However, stock prices frequently exhibit trends and non-linear relationships, violating these assumptions (Hyndman et al., 2021). This can lead to suboptimal performance in capturing underlying trends. While ARIMA-GARCH addresses volatility, a common challenge in stock prices (Hu et al., 2020), its effectiveness can be limited, particularly when dealing with complex non-linearities. Exponential Smoothing can sometimes outperform ARIMA (Shahid et al., 2020), but its suitability depends heavily on the specific characteristics of the stock data.

* **Machine Learning Model Advantages**

In contrast, machine learning models, particularly RNNs and their variants (LSTMs, GRUs), excel at handling non-linear relationships and complex data structures (Selvin et al., 2017). This allows them to capture trends more effectively in financial data like stock prices, which often exhibit these characteristics (Beniwal et al., 2024).

Amongst statistical models, ARIMA-GARCH outperformed others due to its ability to handle volatility (Hu et al., 2020; Kim, 2022). Regarding deep learning models, while GRUs emerged as a strong performer across all companies, the most suitable advanced model differed for each company's stock data:

* Alphabet: GRU
* Meta: RNN (closely followed by GRU)
* Microsoft: GRU

In conclusion, the findings suggest that machine learning models, particularly RNNs and GRUs, are better at capturing underlying trends in the stock prices of leading AI/ML companies compared to traditional statistical models. This contradicts Kobiela et al. (2022) who found ARIMA to be superior, but aligns more closely with Siami-Namini et al. (2018). However, even among machine learning models, the optimal choice may vary depending on the specific company and the characteristics of its stock data. Unlike Beniwal et al. (2024), this study finds GRUs to be superior to LSTMs.

**RQ3: Similarities and Differences in Long-Term and Seasonal Trends of AI Stock Prices**

This research question investigates the similarities and differences in long-term and seasonal trends within the stock prices of Alphabet, Meta, and Microsoft.

* **Similarities**

A key finding from the initial Exploratory Data Analysis (EDA) was the lack of significant seasonal patterns in the price movements for all three companies. This suggests that factors influencing stock prices, such as industry growth, company performance, or macroeconomic conditions (Wu et al., 2021), have a more consistent, long-term effect rather than exhibiting predictable seasonal fluctuations.

Despite the absence of seasonality, all three companies exhibited a notable similarity: the presence of upward long-term trends in their historical stock prices. This finding suggests a sustained positive outlook from investors on the future prospects of these leading AI/ML players.

* **Differences**

While all companies displayed long-term growth, the specific model that best captured this trend differed for each:

* Alphabet: GRU emerged as the most effective model for capturing the long-term trend in Alphabet's stock price.
* Meta: RNN closely followed by GRU performed best in identifying the long-term trend for Meta's stock price.
* Microsoft: GRU again proved to be the most suitable model for capturing the long-term trend in Microsoft's stock price.

The AI/ML industry, unlike sectors with distinct seasonal patterns (e.g., retail with holiday shopping spikes) (Lili et al., 2023), experiences continuous development and adoption across various industries. This ongoing innovation likely contributes to a more consistent demand for AI-related services and products, reflected in the absence of significant seasonal variations in stock prices for all three companies.

In terms of variations in long-term trend model performance, even though all companies exhibited long-term growth, slight differences in their underlying data characteristics or market dynamics might have influenced the performance of specific time series models.

In conclusion, the analysis revealed a lack of significant seasonality in the stock prices, suggesting a focus on long-term factors influencing their value. While all companies displayed long-term upward trends, the optimal model for capturing these trends varied slightly between them. This underscores the importance of considering company-specific characteristics when selecting models for time series analysis in the AI/ML sector.

**RQ4: Accuracy of Time Series Models in Predicting AI Stock Price Trends**

This research question aimed to assess the effectiveness of various time series models in forecasting future trends within the stock prices of leading AI and ML companies. The analysis compared the performance of:

* **Statistical Models:** Represented by ARIMA-GARCH, which emerged as the most effective model amongst traditional statistical methods.
* **Machine Learning Models:** Represented by the GRU model, chosen due to its consistently strong performance.

The analysis focused on a mid-term forecast horizon, encompassing three months (March to May 2024). The analysis revealed that the GRU model significantly outperformed the ARIMA-GARCH model in predicting future trends for all three companies. This is evident from the consistently lower RMSE and MAE values for the GRU model compared to ARIMA-GARCH. This aligns with research highlighting the strengths of RNNs (similar to GRUs) in handling non-linear relationships within financial data (Siami-Namini et al., 2018). These findings support the notion that machine learning models, particularly RNN variants like GRUs, are better suited for capturing complex non-linear relationships and patterns within financial data like stock prices (Hyndman et al., 2021).

In conclusion, this analysis indicates that, for mid-term forecasting of stock prices in leading AI and ML companies, machine learning models, specifically the GRU model in this case, offer superior accuracy compared to traditional statistical models. This finding supports the potential of machine learning approaches like GRUs in financial forecasting, particularly when dealing with data exhibiting non-linear relationships. By showcasing the effectiveness of GRUs, this research mitigates the risks associated with suboptimal decisions arising from traditional models' limitations with complex data. The benefits extend beyond academia, empowering investors to identify opportunities, manage risk, and ultimately enhance returns.

* 1. **Limitations**

This study identified three key limitations associated with the employed methodology:

* **Limitations of Univariate Analysis**

This research utilised historical stock prices to identify long-term trends. However, due to the inherent volatility of financial markets, past performance is not necessarily indicative of future results. Additionally, the univariate analysis employed in this study does not account for external influences, such as broader economic fluctuations, technological advancements, and other factors. This limitation restricts the analysis from capturing the complete picture of stock price movements, potentially leading to inaccurate predictions. Furthermore, univariate analysis cannot explain the "why" behind price changes and the underlying causes influencing them. Understanding these causes is crucial for making informed investment decisions (Narayan et al., 2010).

* **Challenges of Univariate Models**

Univariate models, such as the one employed in this study which relies solely on stock price data, are susceptible to overfitting. This means they might perform well on historical data but fail to generalise and accurately predict future, unseen data. Introducing additional variables relevant to stock price movements can help mitigate overfitting and enhance the model's ability to adapt to changing market conditions (Zhang et al., 2005).

* **Deep Learning Model Uncertainty**

Deep learning models present another challenge when dealing with prediction uncertainty. Each time they are run, they might provide different results due to random initialisation, nonlinearity, and internal randomness (Gal et al., 2016). For research purposes or non-critical analysis, running the model once or utilising ensemble methods is sufficient (Lakshminarayanan et al., 2017). However, for high-risk decisions a more conservative approach should be considered.

* 1. **Recommendations for future research**

While univariate analysis provided valuable insights in this study, future research could explore a multivariate approach that considers additional features relevant to stock price movements. These features could include broader market and economic, customer sentiment data, and technological advancements. A multivariate approach has the potential to improve forecasting accuracy by capturing a more comprehensive picture of the factors influencing stock prices. Additionally, it could help mitigate the overfitting issue observed in univariate models (Tsai et al., 2010).

For high-risk decisions, it is recommended to consider a more conservative approach by using confidence intervals. This provides a range of possible future outcomes and quantifies the uncertainty associated with the predictions. Additionally, integrating the results of the forecasting model with expert knowledge and analysis can combine quantitative and qualitative insights, leading to more informed and robust decision-making (Hyndman et al., 2021).

* 1. **Conclusion**

This chapter summarizes the analysis of stock price forecasting models for leading AI companies (Alphabet, Meta, Microsoft).

**Key Findings:**

* Long-Term Trends: All three companies exhibited upward long-term trends in their stock prices, suggesting a positive outlook from investors on the future of AI/ML.
* Seasonal Trends: No significant seasonal patterns were identified, indicating that factors influencing stock prices have a more consistent, long-term effect.
* Model Comparison: Machine learning models (RNNs, LSTMs, GRUs) were generally more effective than traditional statistical models (ARIMA, ARIMA-GARCH, Exponential Smoothing) in capturing underlying trends.
* GRUs: Amongst machine learning models, GRUs emerged as a strong performer for capturing long-term trends.
* Forecasting Accuracy: The GRU model significantly outperformed the ARIMA-GARCH model in predicting mid-term trends (3 months) for all three companies.

**Limitations:**

* Focusing solely on historical prices might miss key influencing factors and lead to inaccurate predictions.
* The model might be susceptible to overfitting and produce unreliable forecasts for unseen data.
* Deep learning models can exhibit variations in predictions.

**Future Directions:**

* Explore techniques to mitigate overfitting and improve model generalizability.
* For high-risk decisions, use confidence intervals and expert analysis alongside model predictions.

# **Reference List**

Aggarwal, C. (2015) *Data mining: the textbook.* (vol. 1) New York: springer.

Aggarwal, C. (2017) *Outlier Analysis: the textbook.* (Second Edition) New York: springer.

Ariyo, A., Adewumi, O., Ayo, C. (2014) ‘Stock Price Prediction Using the ARIMA Model’ *UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, Cambridge, UK, 2014, pp. 106-112.* [Online] [Accessed on 12th May 2024] DOI: 10.1109/UKSim.2014.67

Arkhangelskaya, E., Nikolenko, S. (2023) ‘Deep Learning for Natural Language Processing: A Survey’*J Math Sci 273, 533–582.*  [Online] [Accessed on 22nd April 2024] DOI: <https://doi.org/10.1007/s10958-023-06519-6>

ArunKumar, A., Kalaga, V., CMS, K., Kawaji, M., Brenza, T. (2021) ‘Forecasting of COVID-19 using deep layer Recurrent Neural Networks (RNNs) with Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) cells.’ *Chaos, Solitons & Fractals, Volume 146, 110861.* [Online] [Accessed on 9th May 2024] DOI: <https://doi.org/10.1016/j.chaos.2021.110861>

Asokan, M., (2022) ‘A study of forecasts in financial time series using machine learning methods’ *Diva Portal.*  [Online] [Accessed on 25th April 2024] <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1671481&dswid=-5264>

Ayodele, E., Zaidi S., Zhang, Z., Scott, J., McLernon, D. (2021) ‘Chapter 9 - A review of deep learning approaches in glove-based gesture classification’*In Intelligent Data-Centric Systems,Machine Learning, Big Data, and IoT for Medical Informatics, Academic Press, Pages 143-164,.*  [Online] [Accessed on 7th May 2024] DOI: <https://doi.org/10.1016/B978-0-12-821777-1.00012-4>

Barclays (2024) *AI Revolution: Productivity boom and beyond*. [Online] [Accessed on 29th March 2024] <https://www.ib.barclays/our-insights/AI-productivity-boom.html?cid=paidsearch-textads_google_google_themes_ai-revolution_uk-we_ai_nonbrand_802852477106&gad_source=1&gclid=Cj0KCQjwzZmwBhD8ARIsAH4v1gXAy0SmYiK-fXDrA7XnR-D6VtLS-J89TRtvyt4mMSDQM1joyLzxqqsaAtPtEALw_wcB&gclsrc=aw.ds>

BBC (2015) *Facebook to open AI lab in Paris*. [Online] [Accessed on 7th June 2024] <https://www.bbc.co.uk/news/technology-32977242>

Bell, E., Bryman, A. and Harley, B. (2022) *Business research methods.* Oxford university press.

Beniwal, M., Singh, A., Kumar, N. (2024) ‘Forecasting multistep daily stock prices for long-term investment decisions: A study of deep learning models on global indices’ *Engineering Applications of Artificial Intelligence, Volume 129, 107617.* [Online] [Accessed on 9th May 2024] DOI: <https://doi.org/10.1016/j.engappai.2023.107617>

Bright, A. (2023) *What are the average returns of the FTSE 100?* IG. [Online] [Accessed on 25th April 2024] <https://www.ig.com/uk/trading-strategies/what-are-the-average-returns-of-the-ftse-100--230511>

Bollerslev, T. (1986) ‘Generalized autoregressive conditional Heteroskedasticity’.*Journal of econometrics, 31(3), pp.307-327.*  [Online] [Accessed on 7th May 2024] DOI: <https://doi.org/10.1016/0304-4076(86)90063-1>

Box, G., Jenkins, G (1976) *Time Series Analysis: Forecasting and Control.* Holden-Day

Box, G., Reinsel, G., Jenkins, G., Ljung, G., (2015) *Time Series Analysis: Forecasting and Control.* *5th Edition.* Wiley

Brown, R. (1959). *Statistical forecasting for inventory control*. McGraw/Hill.

Bughin, J., Seong, J., Manyika, J., Chui, M., Joshi, R. (2018) *Notes from the AI frontier: Modeling the impact of AI on the world economy* [Online] [Accessed on 16th July 2024] https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-modeling-the-impact-of-ai-on-the-world-economy

Burrell, N., Gross, C. (2017) ‘Quantitative Research, Purpose of’ *The SAGE Encyclopedia of Communication Research Methods.* [Online] [Accessed on 19th May 2024] DOI: <https://doi.org/10.4135/9781483381411>

Chatfield, C., Yar, M. (2018). *Time Series Forecasting (6th ed.)*. Chapman and Hall/CRC.

Chen, K., Zhou, Y. and Dai, F. (2015) ‘A LSTM-based method for stock returns prediction: A case study of China stock market’ *2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 2015, pp. 2823-2824.* [Online] [Accessed on 26th April 2024] DOI: 10.1109/BigData.2015.7364089

Chia-Hui, L. (2021) ‘The impact of artificial intelligence on economic growth and welfare’ *Journal of Macroeconomics*, 69: 103342. [Online] [Accessed on 28th March 2024] DOI: <https://doi.org/10.1016/j.jmacro.2021.103342>

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014) ‘Learning phrase representations using RNN encoder-decoder for statistical machine translation’ *arXiv preprint arXiv:1406.1078.* [Online] [Accessed on 9th May 2024] DOI: <https://doi.org/10.48550/arXiv.1406.1078>

Creswell, J.W., Creswell, J.D. (2022) *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. (6th ed.).* SAGE Publications.

Dash, S., (2020) *Smoothing Techniques for time series data.* Medium[Online] [Accessed on 9th May 2024] https://medium.com/@srv96/smoothing-techniques-for-time-series-data-91cccfd008a2

Dégerine, S., Lambert-Lacroix, S. (2003) ‘Characterization of the partial autocorrelation function of nonstationary time series’ *Journal of Multivariate Analysis,Volume 87, Issue 1, Pages 46-59,.* [Online] [Accessed on 4th June 2024] DOI: https://doi.org/10.1016/S0047-259X(03)00025-3

Devadoss, A. (2013) ‘Forecasting of Stock Prices Using Multi Layer Perceptron’ *Int J Comput Algorithm, vol. 2, pp. 440-449*. [Online] [Accessed on 25th April 2024] DOI: 10.20894/IJWT.104.002.002.006

Dickey, D., Fuller, W. (1979) ‘Distribution of the Estimators for Autoregressive Time Series With a Unit Root’ *Journal of the American Statistical Association June 197974(366).* [Online] [Accessed on 4th Jun 2024] DOI: 10.2307/2286348

Earnest, A., Chen, M., Ng, D., Leo, Y. (2005) ‘Using Autoregressive Integrated Moving Average (ARIMA) Models to Predict and Monitor the Number of Beds Occupied During a SARS Outbreak in a Tertiary Hospital in Singapore’ *in BMC Health Service Research, 5(36).* [Online] [Accessed on 25th April 2024] DOI: 10.1186/1472-6963-5-36

Engle, R. (1982) ‘Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation’ *Econometrica: Journal of the econometric society, pp.987-1007*. [Online] [Accessed on 7th May 2024] DOI: <https://doi.org/10.2307/1912773>

Engle, R. (2001) ‘Financial econometrics – A new discipline with new methods’ *Journal of Econometrics, Volume 100, Issue 1, pp 53-56.* [Online] [Accessed on 26th April 2024] DOI: <https://doi.org/10.1016/S0304-4076(00)00053-1>

Fang, Z., Ma, X., Pan, H., Yang, G., Arce, G. (2023) ‘Movement forecasting of financial time series based on adaptive LSTM-BN network’ *Expert Systems with Applications, Volume 213, Part C, 119207.* [Online] [Accessed on 8th May 2024] DOI: <https://doi.org/10.1016/j.eswa.2022.119207>

Fisher, C. and Buglear, J. (2010) *Researching and writing a dissertation: An essential guide for business students.* Pearson Education.

Gal, Y., & Ghahramani, Z. (2016) ‘Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.’ *In Proceedings of the 33rd International Conference on Machine Learning (pp. 1050-1059).* [Online] [Accessed on 28th June 2024] DOI: <https://doi.org/10.48550/arXiv.1506.02142>

Gao, T., Chai, Y., Liu, Y. (2017) ‘Applying long short term memory neural networks for predicting stock closing price’ *IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2017, pp. 575-578.* [Online] [Accessed on 8th May 2024] DOI: 10.1109/ICSESS.2017.8342981

Gardner, E., McKenzie, E. (1985) ‘Forecasting trends in time series’. *Management Science, 31(10), 1237–1246.* [Online] [Accessed on 4th May 2024] DOI: <https://doi.org/10.1287/mnsc.31.10.1237>

Goodfellow, I., Bengio, Y., Courville, A. (2016) *Deep learning.* MIT press.

Guo D., Zhang Y. (2012) ‘Novel recurrent neural network for time-varying problems solving’ *IEEE Computational Intelligence Magazine, 7 (4) (2012), pp. 61-65.* [Online] [Accessed on 7th May 2024] DOI: 10.1109/MCI.2012.2215139

Guresen, E., Kayakutlu, G. (2011) ‘Definition of artificial neural networks with comparison to other networks’ *Procedia Computer Science, Volume 3, Pages 426-433.* [Online] [Accessed on 9th May 2024] DOI: <https://doi.org/10.1016/j.procs.2010.12.071>

Holt, C. (1957) ‘Forecasting seasonals and trends by exponentially weighted averages’ *O.N.R. Memorandum No. 52. Carnegie Institute of Technology, Pittsburgh USA.* [Online] [Accessed on 4th May 2024] DOI: <https://doi.org/10.1016/j.ijforecast.2003.09.015>

Hewamalage, H., Bergmeir, C., Bandara, K. (2021) ‘Recurrent Neural Networks for Time Series Forecasting: Current status and future directions’ *International Journal of Forecasting, Volume 37, Issue 1, Pages 388-427.* [Online] [Accessed on 7th May 2024] DOI: <https://doi.org/10.1016/j.ijforecast.2020.06.008>

Hochreiter, S., Schmidhuber, J. (1997) ‘Long Short-Term Memory’ *in Neural Computation, vol. 9, no. 8, pp. 1735-1780.* [Online] [Accessed on 8th May 2024] DOI: 10.1162/neco.1997.9.8.1735

Hu, Y., Tao, Z., Xing, D., Pan, Z., Zhao, J., Chen, X. (2020) ‘Research on Stock Returns Forecast of the Four Major Banks Based on ARMA and GARCH Model’ *J. Phys.: Conf. Ser. 1616 012075.* [Online] [Accessed on 26th April 2024] DOI: 10.1088/1742-6596/1616/1/012075

Hyndman, R., Athanasopoulos, G (2021) *Forecasting: Principles and Practice.* OTexts

Idrees, S., Alam, M., Agarwal, P. (2019) ‘A Prediction Approach for Stock Market Volatility Based on Time Series Data’ *IEEE Access, vol. 7, pp. 17287-17298.* [Online] [Accessed on 9th May 2024] DOI: 10.1109/ACCESS.2019.2895252

Jain, C., Malehorn, J. (2005) *Practical Guide to Business Forecasting*. Institute of Business Forecasting

Jing, C., Xinyu, H., Hao, J., Xiren, M. (2021) ‘Low-Cost and Device-Free Human Activity Recognition Based on Hierarchical Learning Model’. *Sensors, 21. 2359.* [Online] [Accessed on 8th May 2024] DOI: 10.3390/s21072359

Johnston, M. P. (2014) *Secondary data analysis: A method of which the time has come.* Qualitative and Quantitative Methods in Libraries (QQML), 3, 619-626.

Kadam, P. (2012) ‘Trend Detection and Visualization and Custom Search’ *Research Gate.* [Online] [Accessed on 8th May 2024] https://www.researchgate.net/figure/Comparison-of-moving-averages-The-SMA-has-more-lag-than-the-EMA\_fig1\_235766539

Kai, Y., Xue, D., Xiaohui, Y. (2022) ‘Bayesian empirical likelihood inference and order shrinkage for autoregressive models’ *Statistical Papers. 63.* [Online] [Accessed on 8th May 2024] DOI: 10.1007/s00362-021-01231-6

Kaplan, J. (2016) *Artificial Intelligence. What Everyone Needs to Know*. Oxford University Press.

Kim, S. (2022) ‘ARMA–GARCH model with fractional generalized hyperbolic innovations’ *Financ Innov 8, 48*. [Online] [Accessed on 26th April 2024] DOI: https://doi.org/10.1186/s40854-022-00349-2

Kobiela, D., Krefta, D., Król, W., Weichbroth, P. (2022) ‘ARIMA vs LSTM on NASDAQ stock exchange data’ *Procedia Computer Science, Volume 207, Pages 3836-3845.* [Online] [Accessed on 9th May 2024] DOI: <https://doi.org/10.1016/j.procs.2022.09.445>

Lakshminarayanan, B., Pritzel, A., Blundell, C. (2017) ‘Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles’ *Advances in Neural Information Processing Systems, vol. 31.* [Online] [Accessed on 28th June 2024] DOI: <https://doi.org/10.48550/arXiv.1612.01474>

LeCun, Y., Bengio, Y. & Hinton, G. (2015) ‘Deep learning’ *Nature 521, 436–444.* [Online] [Accessed on 7th June 2024] DOI: <https://doi.org/10.1038/nature14539>

Lee, O. (2013) ‘The functional central limit theorem for ARMA–GARCH processes’ *Economics Letters, Volume 121, Issue 3.* [Online] [Accessed on 7th May 2024] DOI: <https://doi.org/10.1016/j.econlet.2013.09.018>

Lili Ye, Naiming Xie, John E. Boylan, Zhongju Shang (2023) ‘Forecasting seasonal demand for retail: A Fourier time-varying grey model’ *International Journal of Forecasting.* [Online] [Accessed on 23rd June 2024] DOI: <https://doi.org/10.1016/j.ijforecast.2023.12.006>

Liu, X., Du, H., Yu J. (2023) ‘A forecasting method for non-equal interval time series based on recurrent neural network’ *Neurocomputing, Volume 556, 126648.* [Online] [Accessed on 7th May 2024] DOI: <https://doi.org/10.1016/j.neucom.2023.126648>

Ljung, G., Box, G. (1978) ‘On a measure of lack of fit in time series models’ *Biometrika, Volume 65, Issue 2, August 1978, Pages 297–303.* [Online] [Accessed on 4th Jun 2024] DOI: <https://doi.org/10.1093/biomet/65.2.297>

Malki, A., Atlam, E., Hassanien, A., Ewis, A., Dagnew, G., Gad, I., (2022) ‘SARIMA model-based forecasting required number of COVID-19 vaccines globally and empirical analysis of peoples’ view towards the vaccines’ *Alexandria Engineering Journal,Volume 61, Issue 12, 2022, Pages 12091-12110.* [Online] [Accessed on 5th May 2024] DOI: <https://doi.org/10.1016/j.aej.2022.05.051>

Makridakis, S., Wheelwright, S., Hyndman, R. (2008) *FORECASTING METHODS AND APPLICATIONS, 3RD ED.* Wiley India Pvt. Limited

Montgomery, D., Jennings, C., Kulachi, M. (2008) *Introduction to Time Series Analysis and Forecasting*. John Wiley & Sons. Inc.

Montgomery, D., Jennings, C., Kulachi, M. (2015) *Introduction to Time Series Analysis and Forecasting (2nd edition)*. John Wiley & Sons. Inc.

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

Narayan, P., Narayan, S. (2010) ‘Modelling the impact of oil prices on Vietnam's stock prices.’ *Applied Energy, 87(1), 356-361.* [Online] [Accessed on 28th June 2024] DOI: <https://doi.org/10.1016/j.apenergy.2009.05.037>

Naroditsky, D. (2015) *Can A GM And Rybka Beat Stockfish?* [Online] [Accessed on 24th April 2024] <https://www.chess.com/article/view/how-rybka-and-i-tried-to-beat-the-strongest-chess-computer-in-the-world>

Nugus, S. (2009) *Financial Planning Using Excel:* *Forecasting Planning and Budgeting Techniques.* CIMA Publishing

Pil-Soo, K., Dong-Gyu, L., Seong-Whan, L. (2018) ‘Discriminative context learning with gated recurrent unit for group activity recognition’ *Pattern Recognition,Volume 76, Pages 149-161,.* [Online] [Accessed on 9th May 2024] DOI: <https://doi.org/10.1016/j.patcog.2017.10.037>

PWC (2022) *Sizing the prize, PWC’s Global Artificial Intelligence Study: Exploiting the AI Revolution*. [Online] [Accessed on 28th March 2024] <https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html>

Raghunathan, V. (2007) *Stock Exchanges, Investments and Derivatives: Straight Answers to 250 Nagging Questions*. Tata McGraw-Hill.

Rahman, A. (2023) ‘AI Revolution: Shaping Industries Through Artificial Intelligence and Machine Learning.’ *JOURNAL OF ENVIRONMENTAL SCIENCES AND TECHNOLOGY (JEST)*, 2(1) pp. 93-105

Rizwan, M. (2011) ‘Augmented Dickey Fuller Test’ *Available at SSRN* [Online] [Accessed on 4th June 2024] DOI: <http://dx.doi.org/10.2139/ssrn.1911068>

Rumbe, G., Hamasha, M., Mashaqbeh, S. (2024) ‘A comparison of Holts-Winter and Artificial Neural Network approach in forecasting: A case study for tent manufacturing industry’ *Results in Engineering, Volume 21,101899.* [Online] [Accessed on 4th May 2024] DOI: <https://doi.org/10.1016/j.rineng.2024.101899>

Santra, R., (2023) *Stationarity in Time Series.* Medium[Online] [Accessed on 9th May 2024] <https://medium.com/@ritusantra/stationarity-in-time-series-887eb42f62a9>

Saunders, M., Lewis, P., & Thornhill, A. (2019) *Research Methods for Business Students (8th ed.).* Pearson

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V., Oman, K. (2017) ‘Stock price prediction using LSTM, RNN and CNN-sliding window model’ *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13, pp. 1643{1647}*. [Online] [Accessed on 26th April 2024] DOI: 10.1109/ICACCI.2017.8126078

Shahid, S., Rahman, S. (2020) ‘Exponential Smoothing Methods for Detection of the Movement of Stock Prices’ *International Journal of Recent Technology and Engineering (IJRTE) 8(5):1420-1422*. [Online] [Accessed on 26th April 2024] DOI: 10.35940/ijrte.E6409.018520

Shumway, R., Stoffer, d. (2015). *Time Series Analysis and It’s Applications*. EZ Edition

Shanaev, S., Binam, G. (2020) ‘A Generalised Seasonality Test and Applications for Stock Market Seasonality’ *SSRN*. [Online] [Accessed on 25th April 2024] DOI: <https://ssrn.com/abstract=3722154> or [http://dx.doi.org/10.2139/ssrn.3722154](https://dx.doi.org/10.2139/ssrn.3722154)

Siami-Namini, S., Tavakoli, N., Siami Namin, A. (2018) ‘A Comparison of ARIMA and LSTM in Forecasting Time Series’ *17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, pp. 1394-1401.* [Online] [Accessed on 26th April 2024] DOI: 10.1109/ICMLA.2018.00227

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, L., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D. (2016) ‘Mastering the game of Go with deep neural networks and tree search.’ *Nature. 529. 484-489.* [Online] [Accessed on 22nd April 2024] DOI:[10.1038/nature16961](http://dx.doi.org/10.1038/nature16961)

Soori, M., Arezoo, B., Dastres, R. (2023) ‘Artificial intelligence, machine learning and deep learning in advanced robotics, a review’ *Cognitive Robotics*, Vol 3, pp. 54–70. [Online] [Accessed on 27th March 2024] DOI: <https://doi.org/10.1016/j.cogr.2023.04.001>

Srivatsavaya, P., (2023) *LSTM vs GRU.* Medium[Online] [Accessed on 9th May 2024] <https://medium.com/@prudhviraju.srivatsavaya/lstm-vs-gru-c1209b8ecb5a>

Sutskever, I., Vinyals, O. and Le, Q.V. (2014) ‘Sequence to sequence learning with neural networks’ *Advances in neural information processing systems, 27.* [Online] [Accessed on 26th April 2024] DOI: <https://doi.org/10.48550/arXiv.1409.3215>

Tsai, C. F., Hsiao, Y. C. (2010) ‘Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches.’ *Decision Support Systems, 50(1), 258-269.* [Online] [Accessed on 28th June 2024] DOI: <https://doi.org/10.1016/j.dss.2010.08.028>

Valse, A., (2020) *Everything about Components of Time Series: Part-1.* Medium[Online] [Accessed on 9th May 2024] <https://aishwaryagulve97.medium.com/everything-about-components-of-time-series-part-1-7476fb521477>

Wadi, S., Almasarweh, M., Alsaraireh, A. (2018) ‘Predicting Closed Price Time Series Data Using ARIMA Model’ *Modern Applied Science; Vol. 12, No. 11.* [Online] [Accessed on 11th May 2024] DOI: <https://doi.org/10.5539/mas.v12n11p181>

Wang, G., & Si, L. (2018) *Time Series Analysis and Forecasting with Applications.* John Wiley & Sons.

Wang J., Yan J., Li C., Gao R., Zhao R. (2019) ‘Deep heterogeneous GRU model for predictive analytics in smart manufacturing: Application to tool wear prediction.’ *Computers in Industry 111: 1–14.* [Online] [Accessed on 9th May 2024] DOI: <https://doi.org/10.1016/j.compind.2019.06.001>

Winters, P. (1960). ‘Forecasting sales by exponentially weighted moving averages.’ *Management Science, 6(3), 324–342*. [Online] [Accessed on 4th May 2024] DOI: <https://doi.org/10.1287/mnsc.6.3.324>

Wu, Z., Chong, T. (2021) ‘Does the macroeconomy matter to market volatility? Evidence from US industries.’ *Empir Econ 61, 2931–2962.* [Online] [Accessed on 23rd June 2024] DOI: <https://doi.org/10.1007/s00181-020-02001-3>

Yamak, P., Yujian, L. and Gadosey, P. (2019) ‘A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting’ *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence. Pages 49–55.* [Online] [Accessed on 26th April 2024] DOI: <https://doi.org/10.1145/3377713.3377722>

Yamak, P., Yujian, L. and Gadosey, P. (2019) ‘A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting’ *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence. Pages 49–55.* [Online] [Accessed on 26th April 2024] DOI: <https://doi.org/10.1145/3377713.3377722>

Yibin, N. (2019) ‘Forecasting Stock Prices using Exponential Smoothing’ *Towards Data Science.* [Online] [Accessed on 4th May 2024] <https://towardsdatascience.com/forecasting-stock-prices-using-exponential-smoothing-b37dfe54e8e9>

Yin, R. K. (2018) *Case Study Research and Applications: Design and Methods (6th ed.).* SAGE Publications.

Zhang, G., Qi, M. (2005) ‘Neural network forecasting for seasonal and trend time series’ *European Journal of Operational Research, Volume 160, Issue 2,* *Pages 501-514.* [Online] [Accessed on 28th June 2024] DOI: <https://doi.org/10.1016/j.ejor.2003.08.037>

Zhang, X., Zhong, C., Zhang, J., Wang, T., Ng, W. (2023) ‘Robust recurrent neural networks for time series forecasting’ *Neurocomputing, Volume 526, Pages 143-157.* [Online] [Accessed on 8th May 2024] DOI: <https://doi.org/10.1016/j.neucom.2023.01.037>

# **Appendices**

1. **Personal Reflection**

**Description:** Embarking on this dissertation was a chance to embark on a new learning journey and explore the world of statistics and machine learning. My role necessitated these skills, and I was tired of delegating or hiring consultants for tasks I couldn't handle myself. Therefore, I deliberately chose a topic that would push me, and I poured all my energy and time into it.

Initially, apprehension set in. I wasn't sure I could achieve everything I envisioned. Researching revealed no single platform that could handle all my tests, so I needed to improve my rudimentary Python knowledge. Additionally, grasping the methodologies and forecasting models required significant study. Before even starting the methodology chapter, I challenged myself to tackle it head-on. Weeks of dedicated effort followed, filled with both difficulty and frustration. However, I believed in myself and my ability to succeed, and ultimately, I achieved what initially seemed impossible and successfully implemented six models, half of them being deep learning models.

**Feelings:** Throughout the dissertation, a range of emotions coloured my experience. Apprehension at the beginning morphed into frustration as I wrestled with new skills and concepts. However, a sense of accomplishment grew alongside my deepening knowledge. There were moments of exhaustion, particularly when I became somewhat obsessed with dedicating every free minute to the work. Yet, the tangible results kept me motivated and propelled me forward. A sense of empowerment emerged as I pushed myself to a new level.

**Evaluation:** Looking back, the dissertation was a journey with both successes and challenges.

Positives:

* Self-Motivation: Despite limited external incentives, my inherent drive kept me focused.
* Time Management: While juggling work and family life, I discovered that time constraints could actually enhance focus and efficiency. Breaking down tasks into manageable pieces proved to be a successful strategy.
* Learning: I gained invaluable knowledge in analytical areas, both theoretical and practical, equipping me for future complex projects.

Challenges:

* Work-Life Balance: Finding a healthy balance between dissertation work and personal commitments proved demanding at times.
* Initial Uncertainty: Self-doubt regarding my ability to implement all the desired elements was a hurdle to overcome.

**Analysis:** The pressure of limited time became a catalyst for deeper concentration. Focusing on smaller, achievable tasks ensured steady progress while maintaining an overall perspective on the project's completion. My inherent self-motivation also played a key role in overcoming challenges and staying committed to the dissertation's goals.

**Conclusion:** This dissertation was a transformative experience that pushed me beyond my comfort zone. I honed new skills, developed effective time management strategies, and gained a wealth of valuable knowledge. Perhaps most importantly, I discovered a newfound sense of confidence in my ability to tackle complex projects of this type.

**Action Plan:** Looking ahead, I plan to leverage the recommendations outlined in the final chapter of my dissertation to elevate my work. I intend to identify influential factors and select the most appropriate multivariate analysis techniques, then utilise my newly acquired Python skills to implement them. Finally, I will compare the forecasting accuracy of the newly developed multivariate models with the univariate ones used in my dissertation.

1. **Ethos Application**



This form must be completed for all student projects.

**Before you proceed**

Some activities inherently involve increased risks or approval by external regulatory bodies, so a proportional ethics review is not

recommended and a full ethical review may be required.

These may include:

i.

Approval from an external regulatory body (including, but not limited to: NHS (HRA), HMPPS etc.);

ii.

Misleading participants;

iii.

Research without the participants' consent;

iv.

Clinical procedures with participants;

v.

The ingestion or administration of any substance to participants by any means of delivery;

vi.

The use of novel techniques, even where apparently non-invasive, whose safety may be open to question;

vii.

The use of ionising radiation or exposure to radioactive materials;

viii.

Engaging in, witnessing, or monitoring criminal activity;

ix.

Engaging with, or accessing terrorism related materials;

x.

A requirement for security clearance to access participants, data or materials;

xi.

Physical or psychological risk to the participants or researcher;

xii. The project activity takes place in a country outside of the UK for which there is currently an active travel warning issued by the

authorities (see info button);

xiii.

Animals, animal tissue, new or existing human tissue, or biological toxins and agents;

xiv. The sharing of participant personal data with a third party, regardless of the form under which the data is presented.

**If any of these activities are fundamental to your project, please contact your supervisor to determine if a full application is**

**required.**

This form must be completed for each research project which you undertake at the University.

It must be approved by your

supervisor (where relevant) PRIOR to the start of any data collection.

In completing this form, please consult the University's

[Research Ethics and Governance standard](https://www.mmu.ac.uk/research/research-integrity/ethics-and-governance)

[s](https://www.mmu.ac.uk/research/research-integrity/ethics-and-governance)

.

A1a

Please confirm that you will abide by the University's Research Ethics and Governance standards in relation to this project.

Yes

No

**START HERE - Basic Information**



A1b

Data Protection

The University is responsible for complying with the UK General Data Protection Regulation whenever personal data is

processed. Under the Data Protection Policy, all staff and students have a responsibility to comply with the regulation in their

[day-to-day activities. The first step you can take to understand these responsibilities is to review the](https://www.mmu.ac.uk/data-protection/data-protection-in-research)

[Data Protection i](https://www.mmu.ac.uk/data-protection/data-protection-in-research)

[Research guidance page](https://www.mmu.ac.uk/data-protection/data-protection-in-research)

[and complete the University’s Mandatory Data Protection Training. Student training is available](https://www.mmu.ac.uk/data-protection/data-protection-in-research)

through Moodle (in the ‘Skills Online’ section –

[please follow this lin](https://moodle.mmu.ac.uk/course/view.php?id=36&section=5)

[k](https://moodle.mmu.ac.uk/course/view.php?id=36&section=5)

. To make sure your knowledge is up to date, all staff and

)

students must complete the training every two years. If you have any issues in accessing the data protection training or have

any questions about the training, please contact

dataprotection@mmu.ac.uk

.

Have you reviewed the Data Protection guidance pages and completed the Data Protection Training in the last two years?

Yes

No

A2

Are you submitting this application as a learning experience, for a unit which already has ethical approval? (please confirm with

your supervisor)

Yes

No

A3

Student details

Title

First Name

Surname

Hamed

Soleimani

Email

HAMED.SOLEIMANI@stu.mmu.ac.uk

A3.1

Manchester Metropolitan University ID number

23699534

A4

Supervisor

Title

First Name

Surname

Dr

Maksym

Koghut

Faculty



Business and Law

Telephone

0161 247 2000

Email

m.koghut@mmu.ac.uk



A5

Which Faculty is responsible for the project?



Business and Law

A6

Course title

MSc Business Analytics

A7

Project title

Unveiling Trends in Leading AI and Machine Learning Companies' Stock Price: A Time Series Analysis

A8

What is the proposed start date of your project?

25/03/2024

A9

When do you expect to complete your project?

02/08/2024

A10

Please describe the overall aims of your project (3-4 sentences). Research questions should also be included here.

This project explores the potential of time series analysis to predict trends in the stock prices of leading AI/ML companies (Alphabet,

Meta, Microsoft). By analysing historical data for long-term and seasonal patterns, we aim to develop effective forecasting models.

This research can benefit both investors and companies within the AI/ML landscape by offering valuable insights for decision-making

and strategy optimization. research questions:

1

. How well can the chosen time series models identify significant long-term or seasonal trends within the historical stock prices of

leading Artificial Intelligence (AI) and Machine Learning (ML) companies (Alphabet, Meta, and Microsoft)?

2

. In comparison to each other, how suitable are the chosen time series models for uncovering underlying trends in the stock prices

of leading AI and ML companies?

3

. Are there any significant similarities or differences observed between the long-term and seasonal trends exhibited in the chosen

companies' stock prices?

4

. How accurately can the chosen time series models predict future trends in the stock prices of the leading AI and ML companies?



A11

Please describe the research activity

In this research we aim to forecast future trends in the stock prices of leading AI and ML companies: Alphabet (Google), Meta

Platforms (Facebook), and Microsoft. This research adopts a quantitative approach, focusing on analysing numerical data to

generate knowledge and understanding. Quantitative research is well-suited for examining historical stock price data and identifying

potential trends due to its emphasis on numerical precision and statistical analysis.

A deductive approach is employed, utilising established models like ARIMA and LSTM to validate their effectiveness in stock price

prediction. Additionally, a case study strategy is used, focusing on historical stock prices to provide in-depth insights.

The research relies on secondary data, using existing data collected by other sources. Historical stock price data will be obtained from

Yahoo Finance, a reputable source of financial information. Secondary research offers access to extensive datasets that might be

difficult or costly to collect independently, justified by the availability and comprehensiveness of the required historical stock price

data.

The dataset spans from 1st January 2015 to the end of May 2024, encompassing approximately nine years and five months, which

translates to 2,368 working days. For the purposes of model evaluation and analysis, data from the beginning of the period until the

end of February 2024 was utilised, resulting in 2,305 records. The remaining 63 working days, from 1st March 2024 to the end of May

2024

, were reserved for validating the forecasting models.

The primary variables extracted from the dataset are the date and the adjusted close price. The adjusted close price is chosen due to

its adjustment for corporate actions, providing a more accurate reflection of the stock's value over time.

The data collection process is straightforward: the historical stock prices will be downloaded directly from Yahoo Finance after ethos

confirmation is given, following the last business day of May. This ensures that the dataset is up-to-date.

The data collection period begins in 2015, a strategic choice reflecting the significant developments in AI and deep learning during

that time. Around 2015, Google’s advancements in deep learning began to gain substantial attention, marking a period of rapid

innovation in the field. Similarly, Facebook intensified its efforts in AI research, establishing the Facebook AI Research (FAIR) lab in

and achieving notable progress by 2015. This period is therefore pivotal for analysing the stock performance of companies

2013

heavily invested in AI and ML.

The collected data will be analysed using time series forecasting models, including Moving Average, Exponential Smoothing, ARIMA,

ARIMA-GARCH and RNN, GRU and LSTM. Finally, a mid-term future forecast will be made with top performers of statistical and deep

learning models to identify the best model and answer the research questions.

A12

Please provide details of the participants you intend to involve (please include information relating to the number involved and

their demographics; the inclusion and exclusion criteria)

As I intend to gather my data from a public platform, i.e. YAHOO Finance, my research does not involve any participants.

A13

Please upload your project protocol

Documents

**Type**

**Document Name**

**File Name**

**Version Date**

**Version**

**Size**

Project Protocol

Research protocol Final- Hamed Soleimani

Research protocol Final- Hamed Soleimani.docx

01/04/2024

2

25.6

KB



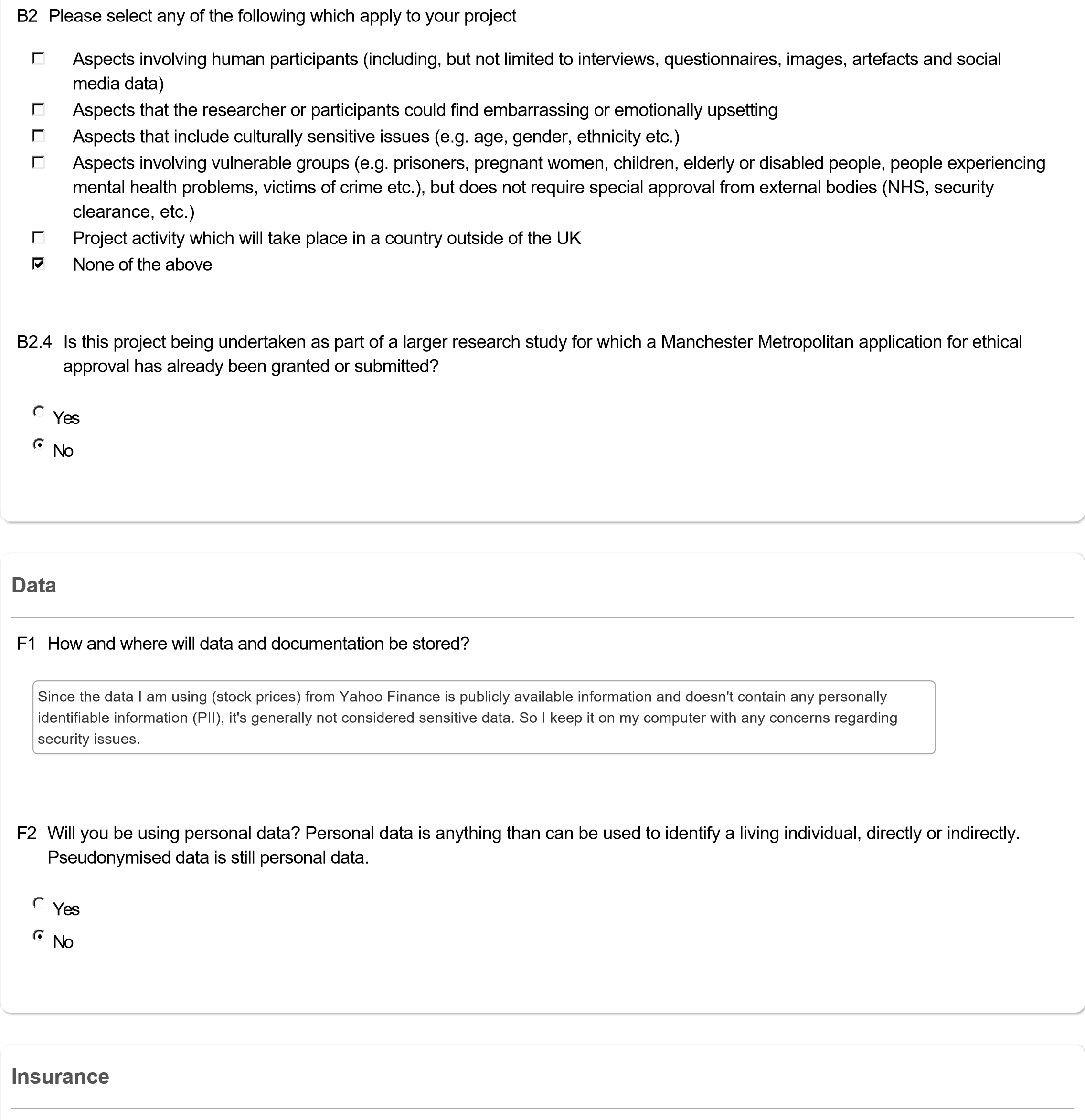
B1

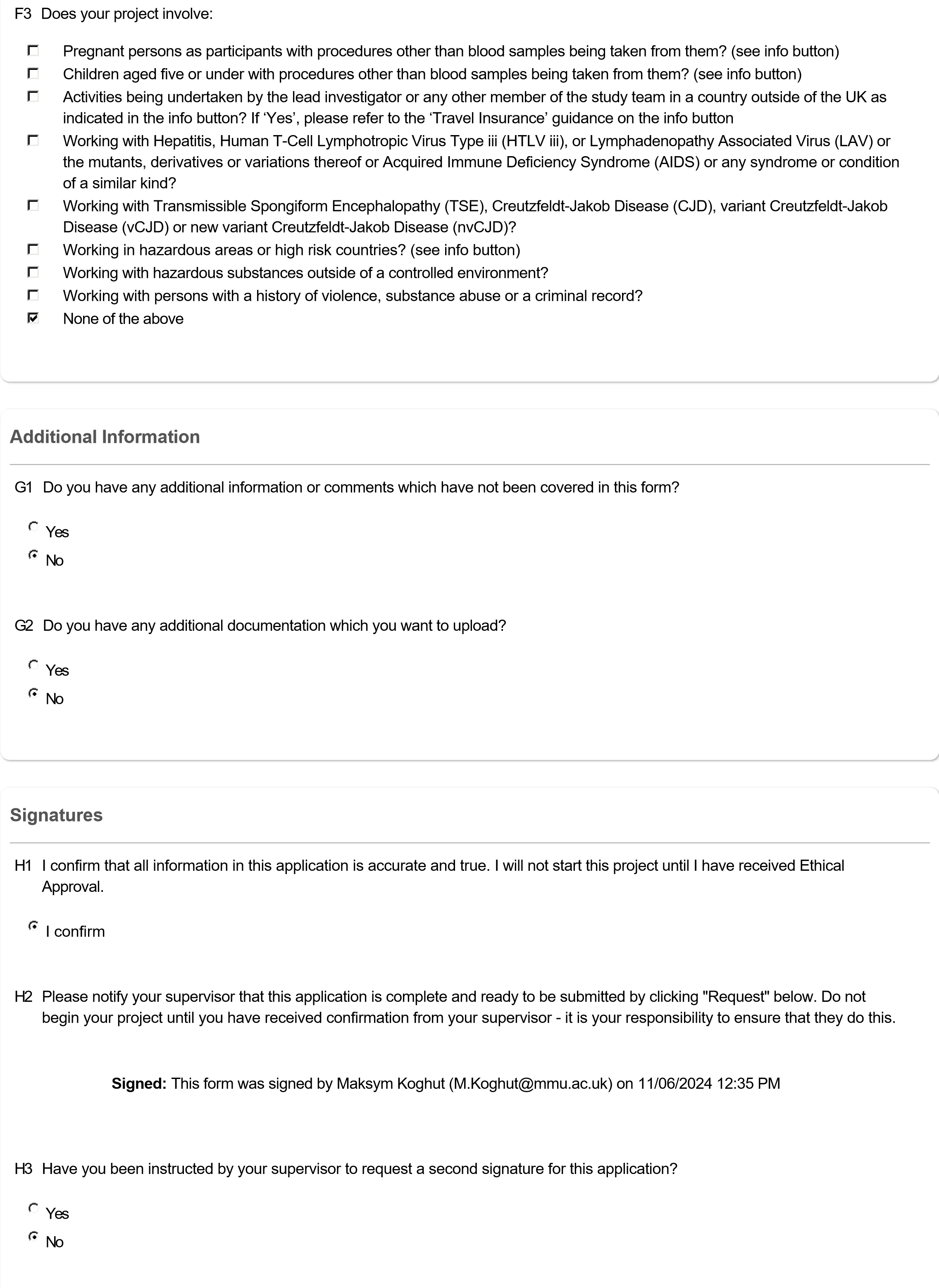
Are there any Health and Safety risks to the researcher and/or participants?

Yes

No

**Project Activity**







H4

By signing this application you are confirming that all details included in the form have been completed accurately and truthfully.

You are also confirming that you will comply with all relevant UK data protection laws, and that that research data generated by

the project will be securely archived in line with requirements specified by the University, unless specific legal, contractual,

ethical or regulatory requirements apply.

**Signed:**

This form was signed by Hamed Soleimani (HAMED.SOLEIMANI@stu.mmu.ac.uk)

on

10

/06/2024 3:08 PM

1. **Outliers Data**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Date | Adj. Close | Date | Adj. Close | Date | Adj. Close | Date | Adj. Close | Date | Adj. Close |
| 7/23/2021 | 137.82 | 10/18/2021 | 142.96 | 12/14/2021 | 144.97 | 9/11/2023 | 137.74 | 1/12/2024 | 144.24 |
| 7/26/2021 | 139.64 | 10/19/2021 | 143.82 | 12/15/2021 | 147.37 | 9/14/2023 | 138.99 | 1/16/2024 | 144.08 |
| 8/9/2021 | 138.00 | 10/20/2021 | 142.41 | 12/16/2021 | 144.84 | 9/15/2023 | 138.30 | 1/17/2024 | 142.89 |
| 8/10/2021 | 138.10 | 10/21/2021 | 142.78 | 12/17/2021 | 142.80 | 9/18/2023 | 138.96 | 1/18/2024 | 144.99 |
| 8/11/2021 | 137.69 | 10/22/2021 | 138.63 | 12/20/2021 | 142.40 | 9/19/2023 | 138.83 | 1/19/2024 | 147.97 |
| 8/12/2021 | 138.39 | 10/25/2021 | 138.77 | 12/21/2021 | 144.22 | 10/6/2023 | 138.73 | 1/22/2024 | 147.71 |
| 8/13/2021 | 138.41 | 10/26/2021 | 139.67 | 12/22/2021 | 146.95 | 10/9/2023 | 139.50 | 1/23/2024 | 148.68 |
| 8/16/2021 | 138.92 | 10/27/2021 | 146.43 | 12/23/2021 | 147.14 | 10/10/2023 | 139.20 | 1/24/2024 | 150.35 |
| 8/20/2021 | 138.44 | 10/28/2021 | 146.13 | 12/27/2021 | 148.06 | 10/11/2023 | 141.70 | 1/25/2024 | 153.64 |
| 8/23/2021 | 141.10 | 10/29/2021 | 148.27 | 12/28/2021 | 146.45 | 10/12/2023 | 140.29 | 1/26/2024 | 153.79 |
| 8/24/2021 | 142.40 | 11/1/2021 | 143.77 | 12/29/2021 | 146.50 | 10/13/2023 | 138.58 | 1/29/2024 | 154.84 |
| 8/25/2021 | 142.95 | 11/2/2021 | 145.86 | 12/30/2021 | 146.00 | 10/16/2023 | 140.49 | 1/30/2024 | 153.05 |
| 8/26/2021 | 142.12 | 11/3/2021 | 146.79 | 12/31/2021 | 144.68 | 10/17/2023 | 140.99 | 1/31/2024 | 141.80 |
| 8/27/2021 | 144.55 | 11/4/2021 | 148.68 | 1/3/2022 | 145.07 | 10/18/2023 | 139.28 | 2/1/2024 | 142.71 |
| 8/30/2021 | 145.47 | 11/5/2021 | 149.24 | 1/4/2022 | 144.42 | 10/19/2023 | 138.98 | 2/2/2024 | 143.54 |
| 8/31/2021 | 145.46 | 11/8/2021 | 149.35 | 1/5/2022 | 137.65 | 10/23/2023 | 137.90 | 2/5/2024 | 144.93 |
| 9/1/2021 | 145.84 | 11/9/2021 | 149.25 | 1/10/2022 | 138.57 | 10/24/2023 | 140.12 | 2/6/2024 | 145.41 |
| 9/2/2021 | 144.22 | 11/10/2021 | 146.63 | 1/11/2022 | 140.02 | 11/16/2023 | 138.70 | 2/7/2024 | 146.68 |
| 9/3/2021 | 144.77 | 11/11/2021 | 146.75 | 1/12/2022 | 141.65 | 11/20/2023 | 137.92 | 2/8/2024 | 147.22 |
| 9/7/2021 | 145.52 | 11/12/2021 | 149.65 | 1/13/2022 | 139.13 | 11/21/2023 | 138.62 | 2/9/2024 | 150.22 |
| 9/8/2021 | 144.88 | 11/15/2021 | 149.39 | 1/14/2022 | 139.79 | 11/22/2023 | 140.02 | 2/12/2024 | 148.73 |
| 9/9/2021 | 144.91 | 11/16/2021 | 149.08 | 2/1/2022 | 137.88 | 11/24/2023 | 138.22 | 2/13/2024 | 146.37 |
| 9/10/2021 | 141.92 | 11/17/2021 | 149.06 | 2/2/2022 | 148.04 | 11/27/2023 | 138.05 | 2/14/2024 | 147.14 |
| 9/13/2021 | 143.46 | 11/18/2021 | 150.71 | 2/3/2022 | 142.65 | 11/28/2023 | 138.62 | 2/15/2024 | 143.94 |
| 9/14/2021 | 143.41 | 11/19/2021 | 149.95 | 2/4/2022 | 143.02 | 12/7/2023 | 138.45 | 2/16/2024 | 141.76 |
| 9/15/2021 | 145.21 | 11/22/2021 | 147.08 | 2/7/2022 | 138.94 | 12/19/2023 | 138.10 | 2/20/2024 | 142.20 |
| 9/16/2021 | 144.37 | 11/23/2021 | 146.76 | 2/8/2022 | 139.21 | 12/20/2023 | 139.66 | 2/21/2024 | 143.84 |
| 9/17/2021 | 141.46 | 11/24/2021 | 146.72 | 2/9/2022 | 141.45 | 12/21/2023 | 141.80 | 2/22/2024 | 145.32 |
| 9/20/2021 | 139.02 | 11/26/2021 | 142.81 | 2/10/2022 | 138.60 | 12/22/2023 | 142.72 | 2/23/2024 | 145.29 |
| 9/21/2021 | 139.65 | 11/29/2021 | 146.11 | 3/22/2022 | 140.28 | 12/26/2023 | 142.82 | 2/26/2024 | 138.75 |
| 9/22/2021 | 140.94 | 11/30/2021 | 142.45 | 3/23/2022 | 138.50 | 12/27/2023 | 141.44 | 2/27/2024 | 140.10 |
| 9/23/2021 | 141.83 | 12/1/2021 | 141.62 | 3/24/2022 | 141.31 | 12/28/2023 | 141.28 | 2/29/2024 | 139.78 |
| 9/24/2021 | 142.63 | 12/2/2021 | 143.78 | 3/25/2022 | 141.52 | 12/29/2023 | 140.93 |  |  |
| 9/27/2021 | 141.50 | 12/3/2021 | 142.52 | 3/28/2022 | 141.95 | 1/2/2024 | 139.56 |  |  |
| 10/7/2021 | 139.19 | 12/6/2021 | 143.80 | 3/29/2022 | 143.25 | 1/3/2024 | 140.36 |  |  |
| 10/8/2021 | 140.06 | 12/7/2021 | 148.04 | 3/30/2022 | 142.64 | 1/4/2024 | 138.04 |  |  |
| 10/11/2021 | 138.85 | 12/8/2021 | 148.72 | 3/31/2022 | 139.65 | 1/8/2024 | 140.53 |  |  |
| 10/13/2021 | 137.90 | 12/9/2021 | 148.11 | 4/1/2022 | 140.70 | 1/9/2024 | 142.56 |  |  |
| 10/14/2021 | 141.41 | 12/10/2021 | 148.68 | 4/4/2022 | 143.64 | 1/10/2024 | 143.80 |  |  |
| 10/15/2021 | 141.68 | 12/13/2021 | 146.70 | 4/5/2022 | 141.06 | 1/11/2024 | 143.67 |  |  |

Table : Outliers of Alphabet Stock Prices

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Adj. Close | Date | Adj. Close |
| 6/28/2021 | 355.26 | 12/27/2023 | 357.45 |
| 7/23/2021 | 369.40 | 12/28/2023 | 357.94 |
| 7/26/2021 | 372.07 | 1/8/2024 | 358.28 |
| 7/27/2021 | 367.42 | 1/9/2024 | 357.05 |
| 7/28/2021 | 372.88 | 1/10/2024 | 370.08 |
| 7/29/2021 | 357.94 | 1/11/2024 | 369.28 |
| 7/30/2021 | 355.92 | 1/12/2024 | 374.09 |
| 8/4/2021 | 358.54 | 1/16/2024 | 367.07 |
| 8/5/2021 | 362.59 | 1/17/2024 | 367.98 |
| 8/6/2021 | 363.12 | 1/18/2024 | 375.73 |
| 8/9/2021 | 361.23 | 1/19/2024 | 383.04 |
| 8/10/2021 | 360.75 | 1/22/2024 | 381.38 |
| 8/11/2021 | 359.58 | 1/23/2024 | 384.79 |
| 8/12/2021 | 362.27 | 1/24/2024 | 390.29 |
| 8/13/2021 | 362.80 | 1/25/2024 | 392.76 |
| 8/16/2021 | 366.17 | 1/26/2024 | 393.72 |
| 8/17/2021 | 358.07 | 1/29/2024 | 400.59 |
| 8/18/2021 | 355.07 | 1/30/2024 | 399.64 |
| 8/20/2021 | 358.99 | 1/31/2024 | 389.73 |
| 8/23/2021 | 362.96 | 2/1/2024 | 394.36 |
| 8/24/2021 | 365.12 | 2/2/2024 | 474.49 |
| 8/25/2021 | 368.00 | 2/5/2024 | 458.92 |
| 8/26/2021 | 363.99 | 2/6/2024 | 454.24 |
| 8/27/2021 | 372.24 | 2/7/2024 | 469.09 |
| 8/30/2021 | 380.26 | 2/8/2024 | 469.50 |
| 8/31/2021 | 378.98 | 2/9/2024 | 467.61 |
| 9/1/2021 | 381.65 | 2/12/2024 | 468.40 |
| 9/2/2021 | 374.88 | 2/13/2024 | 459.63 |
| 9/3/2021 | 375.86 | 2/14/2024 | 472.78 |
| 9/7/2021 | 381.77 | 2/15/2024 | 483.52 |
| 9/8/2021 | 377.17 | 2/16/2024 | 472.82 |
| 9/9/2021 | 377.60 | 2/20/2024 | 471.25 |
| 9/10/2021 | 378.29 | 2/21/2024 | 468.03 |
| 9/13/2021 | 376.11 | 2/22/2024 | 486.13 |
| 9/14/2021 | 376.13 | 2/23/2024 | 484.03 |
| 9/15/2021 | 373.52 | 2/26/2024 | 481.74 |
| 9/16/2021 | 372.66 | 2/27/2024 | 487.05 |
| 9/17/2021 | 364.33 | 2/28/2024 | 484.02 |
| 9/20/2021 | 355.32 | 2/29/2024 | 490.13 |
| 9/21/2021 | 357.10 |  |  |

Table : Outliers of META Stock Prices

|  |  |
| --- | --- |
| Date | Adj. Close |
| 1/18/2024 | 392.44 |
| 1/19/2024 | 397.22 |
| 1/22/2024 | 395.07 |
| 1/23/2024 | 397.45 |
| 1/24/2024 | 401.09 |
| 1/25/2024 | 403.40 |
| 1/26/2024 | 402.46 |
| 1/29/2024 | 408.23 |
| 1/30/2024 | 407.10 |
| 1/31/2024 | 396.13 |
| 2/1/2024 | 402.31 |
| 2/2/2024 | 409.72 |
| 2/5/2024 | 404.17 |
| 2/6/2024 | 404.01 |
| 2/7/2024 | 412.54 |
| 2/8/2024 | 412.60 |
| 2/9/2024 | 419.02 |
| 2/12/2024 | 413.75 |
| 2/13/2024 | 404.84 |
| 2/14/2024 | 408.75 |
| 2/15/2024 | 405.83 |
| 2/16/2024 | 403.33 |
| 2/20/2024 | 402.06 |
| 2/21/2024 | 401.46 |
| 2/22/2024 | 410.91 |
| 2/23/2024 | 409.60 |
| 2/26/2024 | 406.81 |
| 2/27/2024 | 406.75 |
| 2/28/2024 | 406.99 |
| 2/29/2024 | 412.90 |

Table : Outliers of Microsoft Stock Prices

1. **Ljung\_Box Test Results After First-Order Differencing**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Lag | lb\_stat | lb\_pvalue | bp\_stat | bp\_pvalue | Lag | lb\_stat | lb\_pvalue | bp\_stat | bp\_pvalue |
| 1 | 2.352157 | 0.12511 | 2.349097 | 0.125356 | 127 | 271.9014 | 1.44E-12 | 264.5476 | 1.05E-11 |
| 2 | 2.499779 | 0.286536 | 2.496463 | 0.287012 | 128 | 274.8216 | 9.62E-13 | 267.3032 | 7.36E-12 |
| 3 | 7.736351 | 0.051787 | 7.72168 | 0.052128 | 129 | 281.0644 | 2.56E-13 | 273.1914 | 2.21E-12 |
| 4 | 8.532401 | 0.073911 | 8.515659 | 0.074414 | 130 | 281.1281 | 3.75E-13 | 273.2514 | 3.19E-12 |
| 5 | 11.01618 | 0.05106 | 10.9919 | 0.051541 | 131 | 281.2371 | 5.39E-13 | 273.3541 | 4.54E-12 |
| 6 | 16.27747 | 0.01234 | 16.23494 | 0.012547 | 132 | 281.3105 | 7.81E-13 | 273.4233 | 6.50E-12 |
| 7 | 16.4674 | 0.021173 | 16.42412 | 0.021512 | 133 | 281.755 | 1.02E-12 | 273.8418 | 8.45E-12 |
| 8 | 18.45294 | 0.018076 | 18.40105 | 0.018413 | 134 | 283.7746 | 8.68E-13 | 275.7422 | 7.42E-12 |
| 9 | 25.9644 | 0.002071 | 25.87669 | 0.00214 | 135 | 283.8363 | 1.25E-12 | 275.8003 | 1.06E-11 |
| 10 | 26.46274 | 0.003165 | 26.37243 | 0.00327 | 136 | 285.4153 | 1.20E-12 | 277.2848 | 1.03E-11 |
| 11 | 27.1535 | 0.004356 | 27.05929 | 0.004501 | 137 | 289.6771 | 5.59E-13 | 281.2897 | 5.23E-12 |
| 12 | 28.69035 | 0.004379 | 28.58681 | 0.004536 | 138 | 290.6919 | 6.24E-13 | 282.2429 | 5.89E-12 |
| 13 | 29.32865 | 0.005878 | 29.22097 | 0.006089 | 139 | 292.095 | 6.27E-13 | 283.5602 | 6.02E-12 |
| 14 | 29.64135 | 0.008548 | 29.5315 | 0.008849 | 140 | 293.845 | 5.74E-13 | 285.2024 | 5.65E-12 |
| 15 | 40.08339 | 0.000441 | 39.89656 | 0.00047 | 141 | 300.4314 | 1.43E-13 | 291.3804 | 1.61E-12 |
| 16 | 43.27244 | 0.000254 | 43.06071 | 0.000274 | 142 | 300.447 | 2.09E-13 | 291.395 | 2.33E-12 |
| 17 | 44.41461 | 0.000297 | 44.19347 | 0.000321 | 143 | 300.7596 | 2.82E-13 | 291.6879 | 3.11E-12 |
| 18 | 51.53756 | 4.40E-05 | 51.25465 | 4.87E-05 | 144 | 304.7019 | 1.43E-13 | 295.3806 | 1.71E-12 |
| 19 | 55.18947 | 2.17E-05 | 54.8733 | 2.43E-05 | 145 | 304.7054 | 2.09E-13 | 295.3839 | 2.47E-12 |
| 20 | 56.17335 | 2.74E-05 | 55.84779 | 3.06E-05 | 146 | 305.1676 | 2.70E-13 | 295.8164 | 3.17E-12 |
| 21 | 56.1927 | 4.73E-05 | 55.86695 | 5.28E-05 | 147 | 306.6359 | 2.67E-13 | 297.1898 | 3.19E-12 |
| 22 | 62.3481 | 9.98E-06 | 61.95829 | 1.14E-05 | 148 | 306.8668 | 3.65E-13 | 297.4058 | 4.33E-12 |
| 23 | 62.82708 | 1.48E-05 | 62.43207 | 1.69E-05 | 149 | 306.8944 | 5.26E-13 | 297.4316 | 6.14E-12 |
| 24 | 62.98737 | 2.39E-05 | 62.59055 | 2.73E-05 | 150 | 306.8989 | 7.60E-13 | 297.4358 | 8.74E-12 |
| 25 | 65.78445 | 1.59E-05 | 65.35488 | 1.84E-05 | 151 | 310.7977 | 3.95E-13 | 301.0759 | 4.92E-12 |
| 26 | 65.8199 | 2.64E-05 | 65.38991 | 3.04E-05 | 152 | 310.86 | 5.62E-13 | 301.134 | 6.90E-12 |
| 27 | 65.83954 | 4.31E-05 | 65.40929 | 4.94E-05 | 153 | 311.2574 | 7.30E-13 | 301.5047 | 8.91E-12 |
| 28 | 65.92943 | 6.75E-05 | 65.49802 | 7.72E-05 | 154 | 314.7486 | 4.23E-13 | 304.7597 | 5.55E-12 |
| 29 | 66.99055 | 7.74E-05 | 66.54487 | 8.87E-05 | 155 | 315.462 | 5.06E-13 | 305.4245 | 6.65E-12 |
| 30 | 75.81169 | 7.76E-06 | 75.2436 | 9.32E-06 | 156 | 315.8749 | 6.54E-13 | 305.8091 | 8.54E-12 |
| 31 | 80.42654 | 2.88E-06 | 79.79241 | 3.54E-06 | 157 | 321.3387 | 2.27E-13 | 310.8962 | 3.35E-12 |
| 32 | 80.42743 | 4.77E-06 | 79.79329 | 5.84E-06 | 158 | 321.4737 | 3.16E-13 | 311.0219 | 4.60E-12 |
| 33 | 81.50368 | 5.52E-06 | 80.85321 | 6.79E-06 | 159 | 321.7953 | 4.18E-13 | 311.321 | 6.04E-12 |
| 34 | 81.7651 | 8.22E-06 | 81.11054 | 1.01E-05 | 160 | 323.625 | 3.73E-13 | 313.0221 | 5.56E-12 |
| 35 | 89.21472 | 1.27E-06 | 88.44063 | 1.62E-06 | 161 | 323.6879 | 5.26E-13 | 313.0806 | 7.73E-12 |
| 36 | 89.45776 | 1.92E-06 | 88.67967 | 2.45E-06 | 162 | 324.5682 | 6.00E-13 | 313.8983 | 8.87E-12 |
| 37 | 89.46673 | 3.09E-06 | 88.68848 | 3.93E-06 | 163 | 324.7489 | 8.17E-13 | 314.0661 | 1.19E-11 |
| 38 | 89.46805 | 4.91E-06 | 88.68979 | 6.23E-06 | 164 | 324.7788 | 1.15E-12 | 314.0938 | 1.66E-11 |
| 39 | 89.95717 | 6.66E-06 | 89.17021 | 8.44E-06 | 165 | 328.3232 | 6.66E-13 | 317.3816 | 1.03E-11 |
| 40 | 93.89195 | 3.18E-06 | 93.03332 | 4.13E-06 | 166 | 328.812 | 8.37E-13 | 317.8347 | 1.29E-11 |
| 41 | 94.8177 | 3.78E-06 | 93.94181 | 4.92E-06 | 167 | 328.8198 | 1.18E-12 | 317.842 | 1.80E-11 |
| 42 | 103.7699 | 3.85E-07 | 102.7232 | 5.33E-07 | 168 | 328.8345 | 1.67E-12 | 317.8556 | 2.49E-11 |
| 43 | 106.7935 | 2.41E-07 | 105.6878 | 3.41E-07 | 169 | 329.0947 | 2.20E-12 | 318.0965 | 3.26E-11 |
| 44 | 106.8402 | 3.82E-07 | 105.7336 | 5.38E-07 | 170 | 330.8857 | 1.98E-12 | 319.7539 | 3.03E-11 |
| 45 | 111.1733 | 1.60E-07 | 109.9783 | 2.32E-07 | 171 | 331.1606 | 2.61E-12 | 320.0082 | 3.94E-11 |
| 46 | 111.3658 | 2.42E-07 | 110.1668 | 3.49E-07 | 172 | 333.9233 | 1.84E-12 | 322.5624 | 2.95E-11 |
| 47 | 112.2263 | 2.95E-07 | 111.0091 | 4.27E-07 | 173 | 336.4281 | 1.39E-12 | 324.8771 | 2.34E-11 |
| 48 | 117.2854 | 9.94E-08 | 115.9584 | 1.50E-07 | 174 | 339.286 | 9.61E-13 | 327.5169 | 1.72E-11 |
| 49 | 118.5332 | 1.08E-07 | 117.1787 | 1.63E-07 | 175 | 340.087 | 1.11E-12 | 328.2564 | 1.99E-11 |
| 50 | 118.724 | 1.62E-07 | 117.3652 | 2.44E-07 | 176 | 343.3888 | 6.85E-13 | 331.3034 | 1.33E-11 |
| 51 | 119.1934 | 2.20E-07 | 117.8237 | 3.32E-07 | 177 | 344.1771 | 7.93E-13 | 332.0305 | 1.54E-11 |
| 52 | 122.3001 | 1.36E-07 | 120.8577 | 2.09E-07 | 178 | 344.2516 | 1.09E-12 | 332.0991 | 2.09E-11 |
| 53 | 126.0059 | 6.98E-08 | 124.4751 | 1.11E-07 | 179 | 360.1365 | 2.82E-14 | 346.7373 | 8.29E-13 |
| 54 | 126.008 | 1.09E-07 | 124.4772 | 1.73E-07 | 180 | 360.2997 | 3.86E-14 | 346.8876 | 1.12E-12 |
| 55 | 126.2081 | 1.60E-07 | 124.6724 | 2.52E-07 | 181 | 363.4726 | 2.45E-14 | 349.8087 | 7.63E-13 |
| 56 | 126.2503 | 2.44E-07 | 124.7135 | 3.82E-07 | 182 | 364.8754 | 2.44E-14 | 351.0996 | 7.78E-13 |
| 57 | 127.6699 | 2.48E-07 | 126.0967 | 3.90E-07 | 183 | 365.7647 | 2.77E-14 | 351.9175 | 8.90E-13 |
| 58 | 134.1603 | 5.69E-08 | 132.4182 | 9.52E-08 | 184 | 370.7296 | 1.11E-14 | 356.4819 | 4.05E-13 |
| 59 | 134.2858 | 8.48E-08 | 132.5404 | 1.41E-07 | 185 | 371.0314 | 1.47E-14 | 356.7592 | 5.30E-13 |
| 60 | 134.3786 | 1.27E-07 | 132.6307 | 2.10E-07 | 186 | 371.0983 | 2.06E-14 | 356.8207 | 7.29E-13 |
| 61 | 134.6943 | 1.76E-07 | 132.9378 | 2.90E-07 | 187 | 373.9732 | 1.41E-14 | 359.46 | 5.34E-13 |
| 62 | 137.7525 | 1.11E-07 | 135.9112 | 1.88E-07 | 188 | 375.6007 | 1.33E-14 | 360.9533 | 5.17E-13 |
| 63 | 141.6581 | 5.50E-08 | 139.7067 | 9.65E-08 | 189 | 375.8907 | 1.75E-14 | 361.2193 | 6.76E-13 |
| 64 | 144.0365 | 4.22E-08 | 142.017 | 7.56E-08 | 190 | 377.0696 | 1.85E-14 | 362.3001 | 7.23E-13 |
| 65 | 144.2545 | 6.05E-08 | 142.2286 | 1.08E-07 | 191 | 378.059 | 2.04E-14 | 363.2067 | 8.08E-13 |
| 66 | 145.0576 | 7.28E-08 | 143.008 | 1.30E-07 | 192 | 381.7312 | 1.14E-14 | 366.5699 | 4.96E-13 |
| 67 | 146.2077 | 7.93E-08 | 144.1238 | 1.42E-07 | 193 | 381.8348 | 1.58E-14 | 366.6648 | 6.74E-13 |
| 68 | 149.7378 | 4.40E-08 | 147.5467 | 8.17E-08 | 194 | 382.4774 | 1.90E-14 | 367.2528 | 8.12E-13 |
| 69 | 150.5118 | 5.32E-08 | 148.2969 | 9.92E-08 | 195 | 387.4299 | 7.73E-15 | 371.7821 | 3.76E-13 |
| 70 | 150.6164 | 7.75E-08 | 148.3982 | 1.44E-07 | 196 | 390.3828 | 5.19E-15 | 374.4815 | 2.72E-13 |
| 71 | 158.0951 | 1.41E-08 | 155.6401 | 2.83E-08 | 197 | 391.1792 | 6.03E-15 | 375.2092 | 3.17E-13 |
| 72 | 158.7204 | 1.79E-08 | 156.2454 | 3.59E-08 | 198 | 391.5065 | 7.87E-15 | 375.5081 | 4.10E-13 |
| 73 | 161.9323 | 1.08E-08 | 159.3529 | 2.25E-08 | 199 | 392.4962 | 8.68E-15 | 376.4116 | 4.57E-13 |
| 74 | 162.4934 | 1.40E-08 | 159.8954 | 2.90E-08 | 200 | 392.8816 | 1.11E-14 | 376.7632 | 5.81E-13 |
| 75 | 162.4937 | 2.10E-08 | 159.8957 | 4.32E-08 | 201 | 392.8916 | 1.57E-14 | 376.7723 | 8.01E-13 |
| 76 | 165.1606 | 1.49E-08 | 162.4724 | 3.15E-08 | 202 | 393.2318 | 2.03E-14 | 377.0825 | 1.03E-12 |
| 77 | 166.3454 | 1.60E-08 | 163.6166 | 3.41E-08 | 203 | 398.428 | 7.82E-15 | 381.8167 | 4.57E-13 |
| 78 | 167.765 | 1.61E-08 | 164.987 | 3.46E-08 | 204 | 398.665 | 1.04E-14 | 382.0325 | 6.00E-13 |
| 79 | 168.2531 | 2.09E-08 | 165.458 | 4.49E-08 | 205 | 398.7083 | 1.45E-14 | 382.0719 | 8.19E-13 |
| 80 | 170.3504 | 1.74E-08 | 167.4806 | 3.81E-08 | 206 | 398.8974 | 1.94E-14 | 382.244 | 1.08E-12 |
| 81 | 171.7406 | 1.76E-08 | 168.8208 | 3.90E-08 | 207 | 399.902 | 2.12E-14 | 383.1575 | 1.20E-12 |
| 82 | 178.0735 | 4.52E-09 | 174.9231 | 1.08E-08 | 208 | 400.0266 | 2.87E-14 | 383.2708 | 1.59E-12 |
| 83 | 179.9951 | 3.96E-09 | 176.7738 | 9.66E-09 | 209 | 400.0431 | 3.99E-14 | 383.2858 | 2.17E-12 |
| 84 | 180.5563 | 5.05E-09 | 177.3141 | 1.23E-08 | 210 | 401.1026 | 4.30E-14 | 384.2479 | 2.37E-12 |
| 85 | 181.6229 | 5.59E-09 | 178.3404 | 1.37E-08 | 211 | 404.5635 | 2.58E-14 | 387.3891 | 1.56E-12 |
| 86 | 184.6217 | 3.63E-09 | 181.2248 | 9.23E-09 | 212 | 404.7401 | 3.44E-14 | 387.5493 | 2.04E-12 |
| 87 | 184.6877 | 5.29E-09 | 181.2883 | 1.33E-08 | 213 | 406.1787 | 3.37E-14 | 388.8538 | 2.06E-12 |
| 88 | 185.1456 | 6.89E-09 | 181.7283 | 1.73E-08 | 214 | 408.979 | 2.38E-14 | 391.3918 | 1.56E-12 |
| 89 | 185.8169 | 8.44E-09 | 182.3731 | 2.13E-08 | 215 | 409.2152 | 3.12E-14 | 391.6057 | 2.02E-12 |
| 90 | 187.2257 | 8.46E-09 | 183.7257 | 2.16E-08 | 216 | 410.1517 | 3.46E-14 | 392.4537 | 2.25E-12 |
| 91 | 187.341 | 1.20E-08 | 183.8363 | 3.03E-08 | 217 | 415.098 | 1.45E-14 | 396.9303 | 1.09E-12 |
| 92 | 187.6536 | 1.60E-08 | 184.1362 | 4.03E-08 | 218 | 415.1101 | 2.01E-14 | 396.9412 | 1.48E-12 |
| 93 | 206.4954 | 1.35E-10 | 202.2018 | 4.51E-10 | 219 | 415.1715 | 2.75E-14 | 396.9967 | 1.98E-12 |
| 94 | 208.8459 | 1.05E-10 | 204.4545 | 3.60E-10 | 220 | 418.1401 | 1.86E-14 | 399.6795 | 1.45E-12 |
| 95 | 209.0652 | 1.49E-10 | 204.6645 | 5.07E-10 | 221 | 418.2286 | 2.53E-14 | 399.7595 | 1.94E-12 |
| 96 | 210.5718 | 1.46E-10 | 206.107 | 5.06E-10 | 222 | 419.0338 | 2.88E-14 | 400.4865 | 2.22E-12 |
| 97 | 229.2672 | 1.02E-12 | 223.9998 | 4.78E-12 | 223 | 419.1327 | 3.89E-14 | 400.5757 | 2.94E-12 |
| 98 | 229.6854 | 1.40E-12 | 224.3999 | 6.54E-12 | 224 | 419.1933 | 5.29E-14 | 400.6304 | 3.92E-12 |
| 99 | 229.6891 | 2.16E-12 | 224.4035 | 9.99E-12 | 225 | 419.4807 | 6.80E-14 | 400.8894 | 4.98E-12 |
| 100 | 229.6951 | 3.32E-12 | 224.4092 | 1.52E-11 | 226 | 424.1708 | 3.07E-14 | 405.1158 | 2.58E-12 |
| 101 | 238.4035 | 4.01E-13 | 232.7286 | 2.11E-12 | 227 | 425.7393 | 2.92E-14 | 406.5286 | 2.53E-12 |
| 102 | 238.462 | 6.12E-13 | 232.7845 | 3.19E-12 | 228 | 426.838 | 3.10E-14 | 407.5177 | 2.73E-12 |
| 103 | 238.807 | 8.54E-13 | 233.1138 | 4.43E-12 | 229 | 427.399 | 3.73E-14 | 408.0225 | 3.28E-12 |
| 104 | 240.8867 | 7.18E-13 | 235.0979 | 3.82E-12 | 230 | 429.3645 | 3.22E-14 | 409.7902 | 2.96E-12 |
| 105 | 243.6511 | 4.94E-13 | 237.734 | 2.73E-12 | 231 | 429.3857 | 4.41E-14 | 409.8093 | 3.96E-12 |
| 106 | 244.1843 | 6.51E-13 | 238.2422 | 3.59E-12 | 232 | 429.5617 | 5.80E-14 | 409.9675 | 5.13E-12 |
| 107 | 249.4155 | 2.19E-13 | 243.2262 | 1.31E-12 | 233 | 431.2607 | 5.33E-14 | 411.4933 | 4.89E-12 |
| 108 | 249.805 | 3.01E-13 | 243.5971 | 1.80E-12 | 234 | 432.1607 | 5.91E-14 | 412.3012 | 5.47E-12 |
| 109 | 250.5471 | 3.73E-13 | 244.3034 | 2.23E-12 | 235 | 434.2737 | 4.93E-14 | 414.197 | 4.81E-12 |
| 110 | 251.6512 | 4.15E-13 | 245.3539 | 2.51E-12 | 236 | 435.2288 | 5.40E-14 | 415.0536 | 5.32E-12 |
| 111 | 251.838 | 6.00E-13 | 245.5316 | 3.60E-12 | 237 | 436.7524 | 5.17E-14 | 416.4193 | 5.25E-12 |
| 112 | 251.8663 | 9.05E-13 | 245.5584 | 5.36E-12 | 238 | 436.7862 | 7.00E-14 | 416.4496 | 6.96E-12 |
| 113 | 255.4552 | 4.94E-13 | 248.9684 | 3.10E-12 | 239 | 439.243 | 5.40E-14 | 418.6496 | 5.72E-12 |
| 114 | 255.5258 | 7.35E-13 | 249.0354 | 4.55E-12 | 240 | 442.4038 | 3.53E-14 | 421.4787 | 4.09E-12 |
| 115 | 258.5373 | 4.73E-13 | 251.8941 | 3.07E-12 | 241 | 443.654 | 3.60E-14 | 422.5972 | 4.27E-12 |
| 116 | 258.6207 | 6.99E-13 | 251.9733 | 4.49E-12 | 242 | 443.6619 | 4.91E-14 | 422.6042 | 5.68E-12 |
| 117 | 264.2061 | 2.17E-13 | 257.2705 | 1.54E-12 | 243 | 443.747 | 6.55E-14 | 422.6802 | 7.43E-12 |
| 118 | 264.6025 | 2.94E-13 | 257.6463 | 2.07E-12 | 244 | 444.5233 | 7.45E-14 | 423.3738 | 8.48E-12 |
| 119 | 264.8 | 4.20E-13 | 257.8334 | 2.93E-12 | 245 | 444.8162 | 9.45E-14 | 423.6353 | 1.06E-11 |
| 120 | 264.9762 | 6.01E-13 | 258.0002 | 4.16E-12 | 246 | 447.9446 | 6.25E-14 | 426.4272 | 7.69E-12 |
| 121 | 266.6981 | 5.57E-13 | 259.6303 | 3.94E-12 | 247 | 448.1794 | 8.04E-14 | 426.6367 | 9.74E-12 |
| 122 | 267.3571 | 6.95E-13 | 260.2538 | 4.92E-12 | 248 | 450.5775 | 6.29E-14 | 428.7748 | 8.13E-12 |
| 123 | 267.8272 | 9.11E-13 | 260.6985 | 6.43E-12 | 249 | 450.5976 | 8.49E-14 | 428.7927 | 1.07E-11 |
| 124 | 267.8854 | 1.33E-12 | 260.7535 | 9.30E-12 | 250 | 461.3085 | 9.73E-15 | 438.3331 | 1.79E-12 |
| 125 | 270.2518 | 1.03E-12 | 262.9896 | 7.47E-12 | 251 | 466.621 | 3.83E-15 | 443.0627 | 8.39E-13 |
| 126 | 271.9002 | 9.75E-13 | 264.5465 | 7.20E-12 | 252 | 470.9896 | 1.88E-15 | 446.9502 | 4.74E-13 |

Table : Ljung\_Box Test Results of Alphabet

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Lag | lb\_stat | lb\_pvalue | bp\_stat | bp\_pvalue | Lag | lb\_stat | lb\_pvalue | bp\_stat | bp\_pvalue |
| 1 | 5.707707 | 0.016891 | 5.700281 | 0.016962 | 127 | 216.7782 | 1.20E-06 | 211.1518 | 3.99E-06 |
| 2 | 6.147515 | 0.046247 | 6.139327 | 0.046437 | 128 | 217.0748 | 1.49E-06 | 211.4317 | 4.92E-06 |
| 3 | 7.732063 | 0.051886 | 7.720439 | 0.052157 | 129 | 226.8914 | 2.27E-07 | 220.6906 | 8.99E-07 |
| 4 | 8.190353 | 0.084849 | 8.177536 | 0.085287 | 130 | 227.3517 | 2.75E-07 | 221.1245 | 1.08E-06 |
| 5 | 8.638018 | 0.124403 | 8.623843 | 0.125042 | 131 | 233.1498 | 9.99E-08 | 226.5883 | 4.36E-07 |
| 6 | 9.573394 | 0.143805 | 9.555974 | 0.14464 | 132 | 233.7969 | 1.16E-07 | 227.1978 | 5.08E-07 |
| 7 | 9.573394 | 0.214065 | 9.555974 | 0.215163 | 133 | 234.7202 | 1.27E-07 | 228.067 | 5.58E-07 |
| 8 | 11.64076 | 0.167966 | 11.61437 | 0.169257 | 134 | 234.8636 | 1.65E-07 | 228.2019 | 7.17E-07 |
| 9 | 19.88264 | 0.018651 | 19.81694 | 0.019076 | 135 | 236.9502 | 1.39E-07 | 230.1646 | 6.21E-07 |
| 10 | 19.88386 | 0.030371 | 19.81815 | 0.03102 | 136 | 236.9535 | 1.86E-07 | 230.1677 | 8.19E-07 |
| 11 | 22.21191 | 0.022782 | 22.13308 | 0.023362 | 137 | 238.4595 | 1.78E-07 | 231.5829 | 7.97E-07 |
| 12 | 23.12425 | 0.026689 | 23.03988 | 0.027389 | 138 | 239.4639 | 1.90E-07 | 232.5264 | 8.58E-07 |
| 13 | 23.12474 | 0.040203 | 23.04037 | 0.041194 | 139 | 240.0917 | 2.21E-07 | 233.1157 | 9.95E-07 |
| 14 | 24.12907 | 0.044204 | 24.03773 | 0.045344 | 140 | 244.3522 | 1.15E-07 | 237.1139 | 5.58E-07 |
| 15 | 25.8788 | 0.039316 | 25.77456 | 0.04046 | 141 | 244.5797 | 1.46E-07 | 237.3273 | 7.02E-07 |
| 16 | 28.16517 | 0.030213 | 28.04309 | 0.031247 | 142 | 249.081 | 7.17E-08 | 241.5475 | 3.75E-07 |
| 17 | 28.16529 | 0.043044 | 28.0432 | 0.044436 | 143 | 249.3615 | 9.01E-08 | 241.8104 | 4.67E-07 |
| 18 | 36.27189 | 0.006511 | 36.07949 | 0.006892 | 144 | 250.9819 | 8.41E-08 | 243.3282 | 4.45E-07 |
| 19 | 36.27435 | 0.009766 | 36.08193 | 0.010314 | 145 | 251.2719 | 1.05E-07 | 243.5997 | 5.52E-07 |
| 20 | 42.46617 | 0.002403 | 42.21468 | 0.002593 | 146 | 251.9991 | 1.19E-07 | 244.2802 | 6.27E-07 |
| 21 | 43.11694 | 0.003037 | 42.85896 | 0.003276 | 147 | 253.5729 | 1.12E-07 | 245.7523 | 6.03E-07 |
| 22 | 43.50763 | 0.004084 | 43.24558 | 0.004401 | 148 | 253.5731 | 1.49E-07 | 245.7525 | 7.88E-07 |
| 23 | 44.49934 | 0.004575 | 44.22654 | 0.004937 | 149 | 254.9698 | 1.46E-07 | 247.0577 | 7.83E-07 |
| 24 | 44.80449 | 0.006143 | 44.52825 | 0.00662 | 150 | 255.6852 | 1.65E-07 | 247.726 | 8.88E-07 |
| 25 | 44.83098 | 0.008741 | 44.55443 | 0.009395 | 151 | 255.6903 | 2.18E-07 | 247.7307 | 1.15E-06 |
| 26 | 44.83541 | 0.01227 | 44.5588 | 0.013152 | 152 | 261.0204 | 9.13E-08 | 252.7049 | 5.37E-07 |
| 27 | 46.2011 | 0.012103 | 45.90732 | 0.013018 | 153 | 261.2928 | 1.14E-07 | 252.959 | 6.64E-07 |
| 28 | 47.62032 | 0.01177 | 47.30808 | 0.012707 | 154 | 261.5709 | 1.42E-07 | 253.2183 | 8.18E-07 |
| 29 | 47.79237 | 0.015443 | 47.47781 | 0.016639 | 155 | 261.6997 | 1.82E-07 | 253.3383 | 1.04E-06 |
| 30 | 51.81969 | 0.007965 | 51.44925 | 0.008726 | 156 | 263.3951 | 1.67E-07 | 254.9175 | 9.72E-07 |
| 31 | 65.33814 | 0.000303 | 64.77424 | 0.000356 | 157 | 265.2599 | 1.47E-07 | 256.6537 | 8.83E-07 |
| 32 | 65.34353 | 0.000453 | 64.77955 | 0.000531 | 158 | 265.4373 | 1.86E-07 | 256.8189 | 1.10E-06 |
| 33 | 65.93063 | 0.00057 | 65.35774 | 0.000667 | 159 | 266.587 | 1.92E-07 | 257.8883 | 1.15E-06 |
| 34 | 66.65956 | 0.000684 | 66.07529 | 0.000801 | 160 | 267.8971 | 1.91E-07 | 259.1063 | 1.16E-06 |
| 35 | 66.72693 | 0.000972 | 66.14158 | 0.001134 | 161 | 268.104 | 2.39E-07 | 259.2987 | 1.44E-06 |
| 36 | 67.08683 | 0.001264 | 66.49555 | 0.001473 | 162 | 268.8455 | 2.67E-07 | 259.9873 | 1.61E-06 |
| 37 | 67.40924 | 0.001644 | 66.81251 | 0.001912 | 163 | 270.2487 | 2.60E-07 | 261.2901 | 1.59E-06 |
| 38 | 71.33745 | 0.000849 | 70.67257 | 0.001008 | 164 | 272.7724 | 2.01E-07 | 263.6322 | 1.29E-06 |
| 39 | 72.4333 | 0.000908 | 71.74895 | 0.001081 | 165 | 273.411 | 2.29E-07 | 264.2245 | 1.47E-06 |
| 40 | 72.48909 | 0.001259 | 71.80372 | 0.001493 | 166 | 273.5098 | 2.92E-07 | 264.3161 | 1.84E-06 |
| 41 | 80.71907 | 0.000212 | 79.88023 | 0.000265 | 167 | 274.254 | 3.25E-07 | 265.0058 | 2.05E-06 |
| 42 | 80.90276 | 0.000291 | 80.06042 | 0.000363 | 168 | 274.7093 | 3.84E-07 | 265.4275 | 2.41E-06 |
| 43 | 82.58284 | 0.000268 | 81.70771 | 0.000337 | 169 | 274.7184 | 4.95E-07 | 265.436 | 3.06E-06 |
| 44 | 82.62533 | 0.000378 | 81.74936 | 0.000473 | 170 | 276.2419 | 4.70E-07 | 266.8459 | 2.97E-06 |
| 45 | 88.5231 | 0.000116 | 87.52692 | 0.000151 | 171 | 276.4723 | 5.78E-07 | 267.059 | 3.61E-06 |
| 46 | 90.26483 | 0.000105 | 89.2324 | 0.000138 | 172 | 276.6674 | 7.14E-07 | 267.2393 | 4.41E-06 |
| 47 | 92.67547 | 8.02E-05 | 91.59181 | 0.000107 | 173 | 276.7933 | 8.93E-07 | 267.3557 | 5.43E-06 |
| 48 | 92.88258 | 0.000109 | 91.79443 | 0.000145 | 174 | 278.5469 | 8.09E-07 | 268.9755 | 5.06E-06 |
| 49 | 93.30982 | 0.00014 | 92.21222 | 0.000185 | 175 | 278.7258 | 9.99E-07 | 269.1406 | 6.17E-06 |
| 50 | 98.44486 | 5.23E-05 | 97.23147 | 7.20E-05 | 176 | 279.6824 | 1.06E-06 | 270.0233 | 6.58E-06 |
| 51 | 99.12545 | 6.29E-05 | 97.89642 | 8.66E-05 | 177 | 282.5843 | 7.67E-07 | 272.7 | 5.04E-06 |
| 52 | 99.16657 | 8.88E-05 | 97.93657 | 0.000122 | 178 | 283.1121 | 8.84E-07 | 273.1866 | 5.78E-06 |
| 53 | 100.3571 | 9.28E-05 | 99.0987 | 0.000128 | 179 | 283.2073 | 1.11E-06 | 273.2743 | 7.13E-06 |
| 54 | 105.7673 | 3.27E-05 | 104.3776 | 4.69E-05 | 180 | 283.2435 | 1.40E-06 | 273.3077 | 8.85E-06 |
| 55 | 110.1958 | 1.46E-05 | 108.6966 | 2.17E-05 | 181 | 286.1768 | 1.01E-06 | 276.0082 | 6.78E-06 |
| 56 | 110.9606 | 1.73E-05 | 109.4421 | 2.57E-05 | 182 | 286.5217 | 1.20E-06 | 276.3256 | 7.99E-06 |
| 57 | 113.1271 | 1.40E-05 | 111.5532 | 2.12E-05 | 183 | 287.9671 | 1.16E-06 | 277.6551 | 7.84E-06 |
| 58 | 118.5632 | 4.77E-06 | 116.8478 | 7.57E-06 | 184 | 291.0258 | 8.17E-07 | 280.467 | 5.89E-06 |
| 59 | 120.4369 | 4.20E-06 | 118.672 | 6.75E-06 | 185 | 291.0258 | 1.04E-06 | 280.467 | 7.35E-06 |
| 60 | 120.4694 | 6.07E-06 | 118.7036 | 9.68E-06 | 186 | 291.1895 | 1.27E-06 | 280.6173 | 8.90E-06 |
| 61 | 120.7059 | 8.24E-06 | 118.9336 | 1.31E-05 | 187 | 293.7628 | 9.90E-07 | 282.9798 | 7.26E-06 |
| 62 | 122.4345 | 7.53E-06 | 120.6143 | 1.21E-05 | 188 | 293.7775 | 1.25E-06 | 282.9932 | 9.01E-06 |
| 63 | 133.0299 | 6.33E-07 | 130.9111 | 1.13E-06 | 189 | 293.7995 | 1.57E-06 | 283.0135 | 1.11E-05 |
| 64 | 133.0327 | 9.35E-07 | 130.9137 | 1.66E-06 | 190 | 293.7998 | 1.98E-06 | 283.0137 | 1.38E-05 |
| 65 | 133.1583 | 1.33E-06 | 131.0356 | 2.34E-06 | 191 | 293.9505 | 2.42E-06 | 283.1518 | 1.66E-05 |
| 66 | 137.8715 | 5.41E-07 | 135.6099 | 1.00E-06 | 192 | 294.9086 | 2.54E-06 | 284.0293 | 1.76E-05 |
| 67 | 137.9454 | 7.80E-07 | 135.6816 | 1.43E-06 | 193 | 296.0166 | 2.60E-06 | 285.0436 | 1.82E-05 |
| 68 | 138.2744 | 1.04E-06 | 136.0006 | 1.91E-06 | 194 | 300.6846 | 1.38E-06 | 289.3149 | 1.07E-05 |
| 69 | 142.1301 | 5.40E-07 | 139.7375 | 1.02E-06 | 195 | 301.4462 | 1.51E-06 | 290.0114 | 1.17E-05 |
| 70 | 142.5706 | 7.00E-07 | 140.1644 | 1.33E-06 | 196 | 304.2013 | 1.14E-06 | 292.5299 | 9.35E-06 |
| 71 | 143.8262 | 7.28E-07 | 141.3802 | 1.39E-06 | 197 | 304.4167 | 1.37E-06 | 292.7267 | 1.12E-05 |
| 72 | 144.5247 | 8.76E-07 | 142.0563 | 1.67E-06 | 198 | 304.5028 | 1.70E-06 | 292.8053 | 1.36E-05 |
| 73 | 145.3304 | 1.02E-06 | 142.8358 | 1.95E-06 | 199 | 306.0671 | 1.60E-06 | 294.2333 | 1.31E-05 |
| 74 | 146.3016 | 1.14E-06 | 143.7749 | 2.19E-06 | 200 | 307.279 | 1.60E-06 | 295.339 | 1.33E-05 |
| 75 | 149.6092 | 6.90E-07 | 146.9721 | 1.37E-06 | 201 | 307.5187 | 1.92E-06 | 295.5577 | 1.57E-05 |
| 76 | 149.6096 | 9.90E-07 | 146.9725 | 1.95E-06 | 202 | 308.4238 | 2.04E-06 | 296.3827 | 1.68E-05 |
| 77 | 149.7393 | 1.36E-06 | 147.0978 | 2.67E-06 | 203 | 308.4408 | 2.54E-06 | 296.3982 | 2.05E-05 |
| 78 | 149.7407 | 1.93E-06 | 147.0991 | 3.74E-06 | 204 | 308.6908 | 3.02E-06 | 296.6258 | 2.41E-05 |
| 79 | 152.5855 | 1.33E-06 | 149.844 | 2.65E-06 | 205 | 311.897 | 2.12E-06 | 299.5443 | 1.81E-05 |
| 80 | 153.1039 | 1.65E-06 | 150.3439 | 3.28E-06 | 206 | 314.2303 | 1.73E-06 | 301.667 | 1.55E-05 |
| 81 | 157.4838 | 7.69E-07 | 154.5662 | 1.61E-06 | 207 | 315.3976 | 1.75E-06 | 302.7286 | 1.59E-05 |
| 82 | 157.5034 | 1.08E-06 | 154.5851 | 2.25E-06 | 208 | 315.4326 | 2.17E-06 | 302.7604 | 1.93E-05 |
| 83 | 157.5056 | 1.52E-06 | 154.5873 | 3.13E-06 | 209 | 315.5814 | 2.63E-06 | 302.8956 | 2.31E-05 |
| 84 | 158.6622 | 1.60E-06 | 155.7006 | 3.31E-06 | 210 | 316.9143 | 2.58E-06 | 304.106 | 2.30E-05 |
| 85 | 165.5828 | 3.97E-07 | 162.3602 | 8.95E-07 | 211 | 317.857 | 2.71E-06 | 304.9616 | 2.43E-05 |
| 86 | 165.8937 | 5.19E-07 | 162.6592 | 1.17E-06 | 212 | 318.019 | 3.26E-06 | 305.1085 | 2.89E-05 |
| 87 | 166.3824 | 6.47E-07 | 163.129 | 1.45E-06 | 213 | 318.6659 | 3.60E-06 | 305.6951 | 3.19E-05 |
| 88 | 171.2521 | 2.68E-07 | 167.8087 | 6.37E-07 | 214 | 318.9871 | 4.21E-06 | 305.9862 | 3.68E-05 |
| 89 | 171.572 | 3.49E-07 | 168.1159 | 8.27E-07 | 215 | 319.7696 | 4.54E-06 | 306.6951 | 3.98E-05 |
| 90 | 173.0427 | 3.40E-07 | 169.528 | 8.16E-07 | 216 | 321.01 | 4.52E-06 | 307.8182 | 4.02E-05 |
| 91 | 178.0121 | 1.37E-07 | 174.297 | 3.50E-07 | 217 | 321.0259 | 5.54E-06 | 307.8326 | 4.84E-05 |
| 92 | 178.0696 | 1.91E-07 | 174.3522 | 4.85E-07 | 218 | 321.0259 | 6.81E-06 | 307.8326 | 5.83E-05 |
| 93 | 178.1494 | 2.64E-07 | 174.4286 | 6.64E-07 | 219 | 321.0886 | 8.26E-06 | 307.8893 | 6.94E-05 |
| 94 | 179.2681 | 2.81E-07 | 175.5008 | 7.11E-07 | 220 | 321.9098 | 8.81E-06 | 308.6314 | 7.43E-05 |
| 95 | 179.3007 | 3.90E-07 | 175.532 | 9.77E-07 | 221 | 322.2348 | 1.02E-05 | 308.925 | 8.51E-05 |
| 96 | 180.4673 | 4.09E-07 | 176.649 | 1.03E-06 | 222 | 324.1239 | 9.10E-06 | 310.6307 | 7.85E-05 |
| 97 | 181.4541 | 4.47E-07 | 177.5935 | 1.13E-06 | 223 | 329.1936 | 4.75E-06 | 315.2056 | 4.65E-05 |
| 98 | 181.7086 | 5.82E-07 | 177.837 | 1.47E-06 | 224 | 329.6434 | 5.40E-06 | 315.6114 | 5.25E-05 |
| 99 | 185.0429 | 3.60E-07 | 181.0251 | 9.46E-07 | 225 | 331.8788 | 4.54E-06 | 317.6267 | 4.61E-05 |
| 100 | 185.2026 | 4.80E-07 | 181.1778 | 1.25E-06 | 226 | 332.24 | 5.24E-06 | 317.9522 | 5.27E-05 |
| 101 | 185.5739 | 6.05E-07 | 181.5326 | 1.57E-06 | 227 | 333.454 | 5.23E-06 | 319.0456 | 5.34E-05 |
| 102 | 185.5818 | 8.30E-07 | 181.5401 | 2.13E-06 | 228 | 333.454 | 6.40E-06 | 319.0456 | 6.40E-05 |
| 103 | 185.5832 | 1.13E-06 | 181.5414 | 2.88E-06 | 229 | 334.6395 | 6.42E-06 | 320.1124 | 6.51E-05 |
| 104 | 185.5849 | 1.54E-06 | 181.543 | 3.88E-06 | 230 | 334.7694 | 7.67E-06 | 320.2292 | 7.64E-05 |
| 105 | 190.1794 | 7.20E-07 | 185.9244 | 1.93E-06 | 231 | 335.4352 | 8.38E-06 | 320.8277 | 8.34E-05 |
| 106 | 191.2097 | 7.72E-07 | 186.9064 | 2.09E-06 | 232 | 335.4358 | 1.02E-05 | 320.8283 | 9.93E-05 |
| 107 | 191.9756 | 8.80E-07 | 187.6361 | 2.38E-06 | 233 | 336.5016 | 1.04E-05 | 321.7855 | 0.000103 |
| 108 | 193.3746 | 8.65E-07 | 188.9684 | 2.37E-06 | 234 | 336.5017 | 1.26E-05 | 321.7855 | 0.000122 |
| 109 | 196.0462 | 6.33E-07 | 191.5113 | 1.79E-06 | 235 | 336.9298 | 1.43E-05 | 322.1697 | 0.000137 |
| 110 | 197.004 | 6.89E-07 | 192.4227 | 1.96E-06 | 236 | 338.933 | 1.25E-05 | 323.9661 | 0.000125 |
| 111 | 197.6992 | 7.96E-07 | 193.0838 | 2.27E-06 | 237 | 340.9834 | 1.09E-05 | 325.804 | 0.000113 |
| 112 | 197.7122 | 1.07E-06 | 193.0962 | 3.03E-06 | 238 | 341.554 | 1.21E-05 | 326.3153 | 0.000124 |
| 113 | 198.0756 | 1.33E-06 | 193.4415 | 3.73E-06 | 239 | 341.6866 | 1.43E-05 | 326.434 | 0.000145 |
| 114 | 198.0932 | 1.78E-06 | 193.4581 | 4.93E-06 | 240 | 342.5333 | 1.51E-05 | 327.1918 | 0.000154 |
| 115 | 202.3632 | 9.13E-07 | 197.5115 | 2.70E-06 | 241 | 344.3038 | 1.38E-05 | 328.7757 | 0.000145 |
| 116 | 202.5771 | 1.17E-06 | 197.7145 | 3.43E-06 | 242 | 345.0589 | 1.48E-05 | 329.4509 | 0.000155 |
| 117 | 203.6111 | 1.24E-06 | 198.695 | 3.67E-06 | 243 | 346.2113 | 1.49E-05 | 330.4809 | 0.000158 |
| 118 | 207.109 | 7.61E-07 | 202.011 | 2.36E-06 | 244 | 348.1702 | 1.32E-05 | 332.2308 | 0.000145 |
| 119 | 207.735 | 8.87E-07 | 202.6041 | 2.75E-06 | 245 | 352.5729 | 7.88E-06 | 336.162 | 9.76E-05 |
| 120 | 208.7029 | 9.57E-07 | 203.5208 | 2.99E-06 | 246 | 352.6108 | 9.48E-06 | 336.1958 | 0.000115 |
| 121 | 208.894 | 1.22E-06 | 203.7017 | 3.79E-06 | 247 | 352.6112 | 1.15E-05 | 336.1962 | 0.000136 |
| 122 | 209.2311 | 1.51E-06 | 204.0207 | 4.66E-06 | 248 | 354.6125 | 1.01E-05 | 337.9805 | 0.000124 |
| 123 | 209.8462 | 1.76E-06 | 204.6025 | 5.40E-06 | 249 | 354.8431 | 1.17E-05 | 338.186 | 0.000142 |
| 124 | 209.9864 | 2.25E-06 | 204.735 | 6.86E-06 | 250 | 354.9461 | 1.39E-05 | 338.2777 | 0.000166 |
| 125 | 215.0592 | 9.90E-07 | 209.5284 | 3.27E-06 | 251 | 356.3102 | 1.35E-05 | 339.4921 | 0.000164 |
| 126 | 216.3641 | 9.89E-07 | 210.7608 | 3.31E-06 | 252 | 357.6496 | 1.32E-05 | 340.684 | 0.000164 |

Table : Ljung\_Box Test Results of META

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Lag | lb\_stat | lb\_pvalue | bp\_stat | bp\_pvalue | Lag | lb\_stat | lb\_pvalue | bp\_stat | bp\_pvalue |
| 1 | 24.19585 | 8.70E-07 | 24.16437 | 8.85E-07 | 127 | 296.9057 | 1.23E-15 | 291.0217 | 6.76E-15 |
| 2 | 26.14945 | 2.10E-06 | 26.11459 | 2.13E-06 | 128 | 296.9481 | 1.87E-15 | 291.0617 | 1.02E-14 |
| 3 | 27.06392 | 5.71E-06 | 27.02707 | 5.81E-06 | 129 | 302.0215 | 6.57E-16 | 295.8469 | 3.94E-15 |
| 4 | 27.22828 | 1.79E-05 | 27.191 | 1.82E-05 | 130 | 303.1276 | 7.35E-16 | 296.8897 | 4.46E-15 |
| 5 | 30.90355 | 9.79E-06 | 30.85512 | 1.00E-05 | 131 | 303.2008 | 1.11E-15 | 296.9586 | 6.66E-15 |
| 6 | 44.77961 | 5.18E-08 | 44.68304 | 5.41E-08 | 132 | 303.2037 | 1.70E-15 | 296.9613 | 1.01E-14 |
| 7 | 48.73272 | 2.56E-08 | 48.62072 | 2.69E-08 | 133 | 303.4642 | 2.40E-15 | 297.2066 | 1.42E-14 |
| 8 | 61.49423 | 2.37E-10 | 61.32688 | 2.56E-10 | 134 | 308.1647 | 9.51E-16 | 301.6299 | 6.16E-15 |
| 9 | 92.37844 | 5.42E-16 | 92.06377 | 6.27E-16 | 135 | 308.4982 | 1.32E-15 | 301.9436 | 8.52E-15 |
| 10 | 93.0907 | 1.30E-15 | 92.77233 | 1.51E-15 | 136 | 313.1161 | 5.35E-16 | 306.2851 | 3.77E-15 |
| 11 | 97.42624 | 5.77E-16 | 97.08342 | 6.74E-16 | 137 | 314.6017 | 5.33E-16 | 307.6812 | 3.84E-15 |
| 12 | 97.44384 | 1.76E-15 | 97.10092 | 2.05E-15 | 138 | 314.674 | 7.97E-16 | 307.7491 | 5.69E-15 |
| 13 | 97.5954 | 4.84E-15 | 97.25149 | 5.64E-15 | 139 | 314.6741 | 1.21E-15 | 307.7492 | 8.57E-15 |
| 14 | 100.2056 | 4.33E-15 | 99.8436 | 5.08E-15 | 140 | 319.09 | 5.23E-16 | 311.8931 | 4.03E-15 |
| 15 | 117.6739 | 5.34E-18 | 117.1831 | 6.65E-18 | 141 | 320.3882 | 5.50E-16 | 313.1109 | 4.31E-15 |
| 16 | 120.5532 | 4.30E-18 | 120.0399 | 5.40E-18 | 142 | 320.4102 | 8.29E-16 | 313.1315 | 6.43E-15 |
| 17 | 125.5023 | 1.37E-18 | 124.9482 | 1.75E-18 | 143 | 320.7012 | 1.16E-15 | 313.4042 | 8.92E-15 |
| 18 | 128.9216 | 8.52E-19 | 128.3379 | 1.10E-18 | 144 | 324.1422 | 6.60E-16 | 316.6273 | 5.44E-15 |
| 19 | 129.363 | 1.92E-18 | 128.7753 | 2.49E-18 | 145 | 325.9043 | 6.06E-16 | 318.2771 | 5.15E-15 |
| 20 | 129.37 | 5.12E-18 | 128.7822 | 6.59E-18 | 146 | 325.9347 | 9.08E-16 | 318.3056 | 7.62E-15 |
| 21 | 139.0683 | 2.05E-19 | 138.3837 | 2.75E-19 | 147 | 327.2722 | 9.39E-16 | 319.5566 | 8.03E-15 |
| 22 | 140.2105 | 3.29E-19 | 139.5141 | 4.44E-19 | 148 | 327.769 | 1.23E-15 | 320.0211 | 1.05E-14 |
| 23 | 141.9077 | 4.10E-19 | 141.1929 | 5.57E-19 | 149 | 327.7862 | 1.83E-15 | 320.0372 | 1.55E-14 |
| 24 | 144.605 | 3.31E-19 | 143.8598 | 4.54E-19 | 150 | 327.8131 | 2.72E-15 | 320.0624 | 2.27E-14 |
| 25 | 144.6605 | 8.08E-19 | 143.9146 | 1.11E-18 | 151 | 328.3443 | 3.49E-15 | 320.5582 | 2.92E-14 |
| 26 | 147.0902 | 7.15E-19 | 146.3149 | 9.89E-19 | 152 | 328.3451 | 5.19E-15 | 320.559 | 4.29E-14 |
| 27 | 147.1287 | 1.70E-18 | 146.3529 | 2.35E-18 | 153 | 328.5121 | 7.33E-15 | 320.7148 | 6.02E-14 |
| 28 | 147.13 | 4.04E-18 | 146.3542 | 5.57E-18 | 154 | 331.8091 | 4.41E-15 | 323.7888 | 3.87E-14 |
| 29 | 148.6254 | 5.10E-18 | 147.8295 | 7.07E-18 | 155 | 332.5918 | 5.27E-15 | 324.5182 | 4.65E-14 |
| 30 | 148.6387 | 1.17E-17 | 147.8425 | 1.62E-17 | 156 | 333.3561 | 6.32E-15 | 325.2301 | 5.61E-14 |
| 31 | 148.8059 | 2.47E-17 | 148.0074 | 3.41E-17 | 157 | 335.9028 | 4.67E-15 | 327.6012 | 4.35E-14 |
| 32 | 151.152 | 2.14E-17 | 150.3189 | 3.00E-17 | 158 | 337.1529 | 4.90E-15 | 328.7646 | 4.65E-14 |
| 33 | 151.5303 | 4.07E-17 | 150.6914 | 5.69E-17 | 159 | 337.4895 | 6.58E-15 | 329.0777 | 6.22E-14 |
| 34 | 158.3323 | 5.90E-18 | 157.3872 | 8.61E-18 | 160 | 338.8939 | 6.62E-15 | 330.3834 | 6.39E-14 |
| 35 | 159.5992 | 7.81E-18 | 158.6338 | 1.15E-17 | 161 | 339.833 | 7.54E-15 | 331.2561 | 7.36E-14 |
| 36 | 161.1248 | 9.26E-18 | 160.1343 | 1.37E-17 | 162 | 339.8644 | 1.09E-14 | 331.2852 | 1.06E-13 |
| 37 | 164.0431 | 6.30E-18 | 163.0032 | 9.49E-18 | 163 | 340.5344 | 1.34E-14 | 331.9073 | 1.29E-13 |
| 38 | 167.7455 | 3.14E-18 | 166.6413 | 4.85E-18 | 164 | 340.968 | 1.73E-14 | 332.3097 | 1.67E-13 |
| 39 | 169.3684 | 3.54E-18 | 168.2354 | 5.52E-18 | 165 | 341.531 | 2.17E-14 | 332.832 | 2.10E-13 |
| 40 | 170.5871 | 4.65E-18 | 169.4319 | 7.30E-18 | 166 | 342.3853 | 2.51E-14 | 333.624 | 2.45E-13 |
| 41 | 171.2306 | 7.59E-18 | 170.0634 | 1.19E-17 | 167 | 342.7499 | 3.30E-14 | 333.9619 | 3.21E-13 |
| 42 | 173.3428 | 6.96E-18 | 172.1353 | 1.11E-17 | 168 | 346.0442 | 2.01E-14 | 337.0133 | 2.10E-13 |
| 43 | 176.5012 | 4.25E-18 | 175.2321 | 6.93E-18 | 169 | 347.1833 | 2.16E-14 | 338.068 | 2.28E-13 |
| 44 | 176.7391 | 7.97E-18 | 175.4653 | 1.30E-17 | 170 | 350.7078 | 1.24E-14 | 341.3296 | 1.41E-13 |
| 45 | 178.4741 | 8.36E-18 | 177.1649 | 1.38E-17 | 171 | 351.4464 | 1.47E-14 | 342.0128 | 1.70E-13 |
| 46 | 180.7169 | 7.20E-18 | 179.361 | 1.21E-17 | 172 | 355.7482 | 6.89E-15 | 345.9899 | 8.74E-14 |
| 47 | 180.8512 | 1.37E-17 | 179.4925 | 2.29E-17 | 173 | 359.5732 | 3.64E-15 | 349.5247 | 5.03E-14 |
| 48 | 180.9803 | 2.60E-17 | 179.6187 | 4.33E-17 | 174 | 363.3421 | 1.95E-15 | 353.006 | 2.94E-14 |
| 49 | 181.0034 | 5.07E-17 | 179.6413 | 8.41E-17 | 175 | 363.5222 | 2.71E-15 | 353.1722 | 4.03E-14 |
| 50 | 181.0039 | 9.87E-17 | 179.6418 | 1.63E-16 | 176 | 363.9357 | 3.52E-15 | 353.5538 | 5.23E-14 |
| 51 | 181.4032 | 1.64E-16 | 180.0319 | 2.71E-16 | 177 | 364.1402 | 4.83E-15 | 353.7424 | 7.12E-14 |
| 52 | 181.4843 | 3.04E-16 | 180.1112 | 5.01E-16 | 178 | 364.9289 | 5.67E-15 | 354.4695 | 8.42E-14 |
| 53 | 182.6186 | 3.82E-16 | 181.2184 | 6.34E-16 | 179 | 367.3491 | 4.35E-15 | 356.6998 | 6.79E-14 |
| 54 | 185.0843 | 2.95E-16 | 183.6242 | 4.99E-16 | 180 | 367.5458 | 5.96E-15 | 356.881 | 9.22E-14 |
| 55 | 185.9584 | 4.03E-16 | 184.4767 | 6.87E-16 | 181 | 369.7067 | 4.89E-15 | 358.8704 | 7.91E-14 |
| 56 | 186.0854 | 7.17E-16 | 184.6006 | 1.22E-15 | 182 | 370.4761 | 5.76E-15 | 359.5784 | 9.38E-14 |
| 57 | 190.0934 | 3.18E-16 | 188.506 | 5.60E-16 | 183 | 372.2841 | 5.18E-15 | 361.2413 | 8.74E-14 |
| 58 | 192.1515 | 2.82E-16 | 190.5105 | 5.06E-16 | 184 | 372.4489 | 7.13E-15 | 361.3929 | 1.19E-13 |
| 59 | 192.167 | 5.17E-16 | 190.5256 | 9.24E-16 | 185 | 372.4611 | 1.02E-14 | 361.4041 | 1.67E-13 |
| 60 | 192.9514 | 7.17E-16 | 191.2889 | 1.29E-15 | 186 | 372.6188 | 1.40E-14 | 361.5489 | 2.27E-13 |
| 61 | 193.7638 | 9.79E-16 | 192.0791 | 1.76E-15 | 187 | 379.3023 | 3.58E-15 | 367.6846 | 6.92E-14 |
| 62 | 200.8647 | 1.47E-16 | 198.9829 | 2.85E-16 | 188 | 379.3027 | 5.14E-15 | 367.685 | 9.77E-14 |
| 63 | 211.7464 | 5.67E-18 | 209.5579 | 1.24E-17 | 189 | 385.4998 | 1.49E-15 | 373.3688 | 3.32E-14 |
| 64 | 211.9618 | 9.73E-18 | 209.7672 | 2.12E-17 | 190 | 387.4915 | 1.27E-15 | 375.1947 | 2.96E-14 |
| 65 | 212.3444 | 1.56E-17 | 210.1387 | 3.40E-17 | 191 | 387.6025 | 1.78E-15 | 375.2964 | 4.09E-14 |
| 66 | 213.5695 | 1.86E-17 | 211.3276 | 4.08E-17 | 192 | 388.2538 | 2.16E-15 | 375.8929 | 4.97E-14 |
| 67 | 214.0612 | 2.84E-17 | 211.8046 | 6.25E-17 | 193 | 393.2998 | 8.39E-16 | 380.5122 | 2.20E-14 |
| 68 | 214.0942 | 5.07E-17 | 211.8366 | 1.11E-16 | 194 | 393.3546 | 1.19E-15 | 380.5624 | 3.08E-14 |
| 69 | 214.2873 | 8.50E-17 | 212.0237 | 1.86E-16 | 195 | 393.3821 | 1.69E-15 | 380.5875 | 4.31E-14 |
| 70 | 214.3243 | 1.49E-16 | 212.0596 | 3.25E-16 | 196 | 395.043 | 1.58E-15 | 382.1058 | 4.16E-14 |
| 71 | 214.9044 | 2.17E-16 | 212.6214 | 4.72E-16 | 197 | 398.9672 | 8.23E-16 | 385.6913 | 2.40E-14 |
| 72 | 215.3483 | 3.28E-16 | 213.051 | 7.13E-16 | 198 | 399.0962 | 1.14E-15 | 385.8091 | 3.28E-14 |
| 73 | 215.432 | 5.56E-16 | 213.132 | 1.21E-15 | 199 | 399.099 | 1.63E-15 | 385.8117 | 4.60E-14 |
| 74 | 215.571 | 9.22E-16 | 213.2664 | 1.99E-15 | 200 | 399.514 | 2.09E-15 | 386.1904 | 5.87E-14 |
| 75 | 215.6716 | 1.54E-15 | 213.3637 | 3.31E-15 | 201 | 400.368 | 2.39E-15 | 386.9692 | 6.78E-14 |
| 76 | 215.9565 | 2.40E-15 | 213.6389 | 5.15E-15 | 202 | 401.3945 | 2.62E-15 | 387.9048 | 7.54E-14 |
| 77 | 215.9566 | 4.08E-15 | 213.639 | 8.72E-15 | 203 | 403.2355 | 2.33E-15 | 389.5822 | 6.98E-14 |
| 78 | 215.9997 | 6.81E-15 | 213.6806 | 1.45E-14 | 204 | 403.399 | 3.17E-15 | 389.7311 | 9.38E-14 |
| 79 | 216.3657 | 1.02E-14 | 214.0338 | 2.16E-14 | 205 | 403.4192 | 4.47E-15 | 389.7495 | 1.30E-13 |
| 80 | 220.4221 | 4.57E-15 | 217.9459 | 1.02E-14 | 206 | 403.5777 | 6.06E-15 | 389.8937 | 1.74E-13 |
| 81 | 221.2454 | 5.87E-15 | 218.7395 | 1.32E-14 | 207 | 406.7311 | 3.89E-15 | 392.7613 | 1.21E-13 |
| 82 | 231.3315 | 3.67E-16 | 228.4583 | 9.43E-16 | 208 | 411.4081 | 1.70E-15 | 397.0124 | 6.00E-14 |
| 83 | 231.3352 | 6.22E-16 | 228.4618 | 1.59E-15 | 209 | 411.4573 | 2.37E-15 | 397.0571 | 8.25E-14 |
| 84 | 232.5137 | 7.14E-16 | 229.5964 | 1.84E-15 | 210 | 412.2116 | 2.77E-15 | 397.742 | 9.70E-14 |
| 85 | 232.5139 | 1.20E-15 | 229.5966 | 3.08E-15 | 211 | 412.2962 | 3.83E-15 | 397.8188 | 1.32E-13 |
| 86 | 235.4376 | 7.79E-16 | 232.4087 | 2.07E-15 | 212 | 412.5475 | 5.06E-15 | 398.0468 | 1.73E-13 |
| 87 | 240.2104 | 2.78E-16 | 236.9973 | 7.87E-16 | 213 | 413.833 | 5.16E-15 | 399.2125 | 1.80E-13 |
| 88 | 244.2202 | 1.27E-16 | 240.8506 | 3.79E-16 | 214 | 415.0797 | 5.32E-15 | 400.3423 | 1.90E-13 |
| 89 | 245.8558 | 1.25E-16 | 242.4217 | 3.81E-16 | 215 | 419.4705 | 2.52E-15 | 404.32 | 1.01E-13 |
| 90 | 245.9585 | 2.03E-16 | 242.5202 | 6.15E-16 | 216 | 419.4798 | 3.53E-15 | 404.3284 | 1.40E-13 |
| 91 | 246.0388 | 3.30E-16 | 242.5973 | 9.95E-16 | 217 | 419.6089 | 4.80E-15 | 404.4452 | 1.87E-13 |
| 92 | 246.2143 | 5.18E-16 | 242.7657 | 1.55E-15 | 218 | 419.6953 | 6.57E-15 | 404.5234 | 2.52E-13 |
| 93 | 247.5271 | 5.64E-16 | 244.0244 | 1.71E-15 | 219 | 419.6954 | 9.17E-15 | 404.5235 | 3.46E-13 |
| 94 | 247.5507 | 9.21E-16 | 244.047 | 2.78E-15 | 220 | 419.7106 | 1.27E-14 | 404.5372 | 4.72E-13 |
| 95 | 248.0667 | 1.28E-15 | 244.5413 | 3.87E-15 | 221 | 420.825 | 1.35E-14 | 405.5439 | 5.09E-13 |
| 96 | 250.0793 | 1.11E-15 | 246.4684 | 3.44E-15 | 222 | 423.2022 | 1.05E-14 | 407.6901 | 4.22E-13 |
| 97 | 252.1174 | 9.56E-16 | 248.419 | 3.04E-15 | 223 | 424.6761 | 1.03E-14 | 409.0202 | 4.22E-13 |
| 98 | 252.4182 | 1.42E-15 | 248.7067 | 4.49E-15 | 224 | 428.6336 | 5.46E-15 | 412.5899 | 2.51E-13 |
| 99 | 254.3782 | 1.25E-15 | 250.5809 | 4.05E-15 | 225 | 428.6537 | 7.56E-15 | 412.608 | 3.41E-13 |
| 100 | 254.6338 | 1.86E-15 | 250.8252 | 6.03E-15 | 226 | 428.8087 | 1.01E-14 | 412.7476 | 4.50E-13 |
| 101 | 256.6998 | 1.58E-15 | 252.7989 | 5.27E-15 | 227 | 429.7927 | 1.11E-14 | 413.6339 | 4.99E-13 |
| 102 | 262.1651 | 4.69E-16 | 258.0177 | 1.70E-15 | 228 | 430.0615 | 1.43E-14 | 413.8759 | 6.41E-13 |
| 103 | 269.4822 | 7.72E-17 | 265.0016 | 3.14E-16 | 229 | 431.0142 | 1.58E-14 | 414.7332 | 7.15E-13 |
| 104 | 270.445 | 9.31E-17 | 265.9201 | 3.82E-16 | 230 | 431.353 | 2.01E-14 | 415.038 | 9.03E-13 |
| 105 | 270.6392 | 1.43E-16 | 266.1054 | 5.82E-16 | 231 | 432.3343 | 2.19E-14 | 415.9201 | 9.99E-13 |
| 106 | 275.0923 | 5.76E-17 | 270.3499 | 2.52E-16 | 232 | 432.662 | 2.79E-14 | 416.2145 | 1.26E-12 |
| 107 | 276.6928 | 5.68E-17 | 271.8748 | 2.54E-16 | 233 | 436.6174 | 1.50E-14 | 419.7668 | 7.60E-13 |
| 108 | 276.7011 | 9.18E-17 | 271.8826 | 4.07E-16 | 234 | 439.4193 | 1.06E-14 | 422.282 | 5.80E-13 |
| 109 | 277.4216 | 1.19E-16 | 272.5685 | 5.28E-16 | 235 | 441.4425 | 9.06E-15 | 424.0973 | 5.19E-13 |
| 110 | 278.7858 | 1.25E-16 | 273.8664 | 5.67E-16 | 236 | 442.993 | 8.63E-15 | 425.4877 | 5.11E-13 |
| 111 | 279.4126 | 1.66E-16 | 274.4625 | 7.52E-16 | 237 | 442.9944 | 1.19E-14 | 425.489 | 6.90E-13 |
| 112 | 279.7938 | 2.36E-16 | 274.8249 | 1.07E-15 | 238 | 443.5578 | 1.43E-14 | 425.9938 | 8.30E-13 |
| 113 | 281.3951 | 2.31E-16 | 276.3463 | 1.07E-15 | 239 | 443.7371 | 1.88E-14 | 426.1543 | 1.08E-12 |
| 114 | 282.3785 | 2.73E-16 | 277.2803 | 1.27E-15 | 240 | 444.4238 | 2.20E-14 | 426.769 | 1.26E-12 |
| 115 | 282.422 | 4.27E-16 | 277.3216 | 1.98E-15 | 241 | 446.0807 | 2.04E-14 | 428.2512 | 1.22E-12 |
| 116 | 282.7376 | 6.13E-16 | 277.621 | 2.83E-15 | 242 | 448.6711 | 1.52E-14 | 430.5675 | 9.73E-13 |
| 117 | 282.9539 | 9.04E-16 | 277.8261 | 4.15E-15 | 243 | 448.9236 | 1.97E-14 | 430.7932 | 1.24E-12 |
| 118 | 283.4194 | 1.23E-15 | 278.2674 | 5.67E-15 | 244 | 449.3451 | 2.44E-14 | 431.1698 | 1.53E-12 |
| 119 | 286.1861 | 8.48E-16 | 280.8889 | 4.07E-15 | 245 | 450.1526 | 2.76E-14 | 431.8907 | 1.74E-12 |
| 120 | 286.8005 | 1.10E-15 | 281.4708 | 5.31E-15 | 246 | 450.3237 | 3.61E-14 | 432.0434 | 2.25E-12 |
| 121 | 287.9838 | 1.21E-15 | 282.591 | 5.92E-15 | 247 | 451.8955 | 3.42E-14 | 433.4455 | 2.21E-12 |
| 122 | 291.2099 | 7.28E-16 | 285.6436 | 3.75E-15 | 248 | 457.502 | 1.27E-14 | 438.4442 | 9.76E-13 |
| 123 | 291.3851 | 1.08E-15 | 285.8093 | 5.51E-15 | 249 | 457.5065 | 1.73E-14 | 438.4482 | 1.30E-12 |
| 124 | 291.4301 | 1.65E-15 | 285.8519 | 8.36E-15 | 250 | 457.7225 | 2.24E-14 | 438.6406 | 1.67E-12 |
| 125 | 291.9784 | 2.17E-15 | 286.37 | 1.10E-14 | 251 | 458.5912 | 2.49E-14 | 439.414 | 1.88E-12 |
| 126 | 292.103 | 3.22E-15 | 286.4876 | 1.63E-14 | 252 | 459.3211 | 2.87E-14 | 440.0635 | 2.17E-12 |

Table : Ljung\_Box Test Results of Microsoft

1. **Mid-Term Stock Price Forecasts (Real vs. Predicted)**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Real Adj. Close | ARIMA-GARCH Forecast | GRU Forecast |
| 3/1/2024 | 138.08 | 117.0109123 | 140.77133 |
| 3/4/2024 | 134.20 | 117.1148563 | 141.83636 |
| 3/5/2024 | 133.78 | 117.1190027 | 142.74832 |
| 3/6/2024 | 132.56 | 117.2080723 | 143.68272 |
| 3/7/2024 | 135.24 | 117.2153377 | 144.5996 |
| 3/8/2024 | 136.29 | 117.3022552 | 145.50374 |
| 3/11/2024 | 138.94 | 117.3115488 | 146.38942 |
| 3/12/2024 | 139.62 | 117.3964951 | 147.25711 |
| 3/13/2024 | 140.77 | 117.4077094 | 148.10616 |
| 3/14/2024 | 144.34 | 117.4907838 | 148.93617 |
| 3/15/2024 | 142.17 | 117.5038225 | 149.74673 |
| 3/18/2024 | 148.48 | 117.5851188 | 150.53743 |
| 3/19/2024 | 147.92 | 117.5998905 | 151.30797 |
| 3/20/2024 | 149.68 | 117.6794978 | 152.05807 |
| 3/21/2024 | 148.74 | 117.6959156 | 152.78758 |
| 3/22/2024 | 151.77 | 117.7739186 | 153.4963 |
| 3/25/2024 | 151.15 | 117.7918999 | 154.18423 |
| 3/26/2024 | 151.70 | 117.868379 | 154.8513 |
| 3/27/2024 | 151.94 | 117.8878456 | 155.49756 |
| 3/28/2024 | 152.26 | 117.9628772 | 156.12308 |
| 4/1/2024 | 156.50 | 117.9837545 | 156.72797 |
| 4/2/2024 | 155.87 | 118.0574112 | 157.31245 |
| 4/3/2024 | 156.37 | 118.0796285 | 157.87672 |
| 4/4/2024 | 151.94 | 118.1519792 | 158.42104 |
| 4/5/2024 | 153.94 | 118.1754693 | 158.94566 |
| 4/8/2024 | 156.14 | 118.2465796 | 159.45093 |
| 4/9/2024 | 158.14 | 118.2712787 | 159.9372 |
| 4/10/2024 | 157.66 | 118.3412106 | 160.40485 |
| 4/11/2024 | 160.79 | 118.3670581 | 160.85428 |
| 4/12/2024 | 159.19 | 118.4358708 | 161.2859 |
| 4/15/2024 | 156.33 | 118.4628091 | 161.70016 |
| 4/16/2024 | 156.00 | 118.5305587 | 162.09752 |
| 4/17/2024 | 156.88 | 118.5585331 | 162.4784 |
| 4/18/2024 | 157.46 | 118.6252729 | 162.8433 |
| 4/19/2024 | 155.72 | 118.6542315 | 163.19272 |
| 4/22/2024 | 157.95 | 118.720012 | 163.52707 |
| 4/23/2024 | 159.92 | 118.7499055 | 163.8469 |
| 4/24/2024 | 161.10 | 118.8147749 | 164.15265 |
| 4/25/2024 | 157.95 | 118.8455564 | 164.44481 |
| 4/26/2024 | 173.69 | 118.9095604 | 164.72385 |
| 4/29/2024 | 167.90 | 118.9411854 | 164.99023 |
| 4/30/2024 | 164.64 | 119.0043673 | 165.24446 |
| 5/1/2024 | 165.57 | 119.0367934 | 165.48695 |
| 5/2/2024 | 168.46 | 119.0991945 | 165.7182 |
| 5/3/2024 | 168.99 | 119.1323817 | 165.93864 |
| 5/6/2024 | 169.83 | 119.194041 | 166.14868 |
| 5/7/2024 | 172.98 | 119.2279511 | 166.34874 |
| 5/8/2024 | 171.16 | 119.2889059 | 166.53928 |
| 5/9/2024 | 171.58 | 119.3235026 | 166.72064 |
| 5/10/2024 | 170.29 | 119.3837882 | 166.89323 |
| 5/13/2024 | 170.90 | 119.4190371 | 167.05743 |
| 5/14/2024 | 171.93 | 119.4786871 | 167.21362 |
| 5/15/2024 | 173.88 | 119.5145555 | 167.36217 |
| 5/16/2024 | 175.43 | 119.5736017 | 167.50336 |
| 5/17/2024 | 177.29 | 119.6100585 | 167.63756 |
| 5/20/2024 | 178.46 | 119.6685312 | 167.7651 |
| 5/21/2024 | 179.54 | 119.705547 | 167.88628 |
| 5/22/2024 | 178.00 | 119.763475 | 168.00139 |
| 5/23/2024 | 175.06 | 119.8010217 | 168.11072 |
| 5/24/2024 | 176.33 | 119.8584322 | 168.21452 |
| 5/28/2024 | 178.02 | 119.8964832 | 168.31306 |
| 5/29/2024 | 177.40 | 119.9534022 | 168.4066 |
| 5/30/2024 | 173.56 | 119.9919323 | 168.4954 |

Table : Mid-Term Stock Price Forecasts (Real vs. Predicted) for Alphabet

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Real Adj. Close | ARIMA-GARCH Forecast | GRU Forecast |
| 3/1/2024 | 502.299988 | 210.5180617 | 489.88678 |
| 3/4/2024 | 498.190002 | 210.4653215 | 488.42148 |
| 3/5/2024 | 490.220001 | 210.6770481 | 487.8196 |
| 3/6/2024 | 496.089996 | 210.6036823 | 487.45547 |
| 3/7/2024 | 512.190002 | 210.8142265 | 487.2603 |
| 3/8/2024 | 505.950012 | 210.7430348 | 487.05295 |
| 3/11/2024 | 483.589996 | 210.9513757 | 486.80322 |
| 3/12/2024 | 499.75 | 210.8823722 | 486.5621 |
| 3/13/2024 | 495.570007 | 211.0885419 | 486.34396 |
| 3/14/2024 | 491.829987 | 211.0216926 | 486.14682 |
| 3/15/2024 | 484.100006 | 211.225725 | 485.96536 |
| 3/18/2024 | 496.980011 | 211.1609963 | 485.79602 |
| 3/19/2024 | 496.23999 | 211.3629247 | 485.6382 |
| 3/20/2024 | 505.519989 | 211.3002835 | 485.49158 |
| 3/21/2024 | 507.76001 | 211.5001407 | 485.35544 |
| 3/22/2024 | 509.579987 | 211.4395544 | 485.22913 |
| 3/25/2024 | 503.019989 | 211.6373728 | 485.11182 |
| 3/26/2024 | 495.890015 | 211.5788094 | 485.00278 |
| 3/27/2024 | 493.859985 | 211.7746208 | 484.90146 |
| 3/28/2024 | 485.579987 | 211.7180487 | 484.8073 |
| 4/1/2024 | 491.350006 | 211.9118843 | 484.71988 |
| 4/2/2024 | 497.369995 | 211.8572726 | 484.63858 |
| 4/3/2024 | 506.73999 | 212.0491632 | 484.56302 |
| 4/4/2024 | 510.920013 | 211.9964812 | 484.49283 |
| 4/5/2024 | 527.340027 | 212.1864572 | 484.42758 |
| 4/8/2024 | 519.25 | 212.1356748 | 484.3669 |
| 4/9/2024 | 516.900024 | 212.3237661 | 484.31046 |
| 4/10/2024 | 519.830017 | 212.2748537 | 484.2581 |
| 4/11/2024 | 523.159973 | 212.4610896 | 484.20932 |
| 4/12/2024 | 511.899994 | 212.414018 | 484.164 |
| 4/15/2024 | 500.230011 | 212.5984275 | 484.1219 |
| 4/16/2024 | 499.76001 | 212.5531681 | 484.08276 |
| 4/17/2024 | 494.170013 | 212.7357796 | 484.04633 |
| 4/18/2024 | 501.799988 | 212.692304 | 484.01242 |
| 4/19/2024 | 481.070007 | 212.8731457 | 483.98096 |
| 4/22/2024 | 481.730011 | 212.8314261 | 483.95163 |
| 4/23/2024 | 496.100006 | 213.0105255 | 483.9244 |
| 4/24/2024 | 493.5 | 212.9705346 | 483.89905 |
| 4/25/2024 | 441.380005 | 213.1479188 | 483.87552 |
| 4/26/2024 | 443.290009 | 213.1096297 | 483.8536 |
| 4/29/2024 | 432.619995 | 213.2853255 | 483.83322 |
| 4/30/2024 | 430.170013 | 213.2487115 | 483.8142 |
| 5/1/2024 | 439.190002 | 213.4227453 | 483.7966 |
| 5/2/2024 | 441.679993 | 213.3877803 | 483.78018 |
| 5/3/2024 | 451.959991 | 213.560178 | 483.765 |
| 5/6/2024 | 465.679993 | 213.5268363 | 483.75082 |
| 5/7/2024 | 468.23999 | 213.6976233 | 483.7376 |
| 5/8/2024 | 472.600006 | 213.6658797 | 483.72537 |
| 5/9/2024 | 475.420013 | 213.8350813 | 483.71396 |
| 5/10/2024 | 476.200012 | 213.8049107 | 483.70328 |
| 5/13/2024 | 468.01001 | 213.9725515 | 483.69342 |
| 5/14/2024 | 471.850006 | 213.9439295 | 483.68423 |
| 5/15/2024 | 481.540009 | 214.1100338 | 483.6757 |
| 5/16/2024 | 473.230011 | 214.0829362 | 483.66776 |
| 5/17/2024 | 471.910004 | 214.2475281 | 483.66034 |
| 5/20/2024 | 468.839996 | 214.2219311 | 483.65347 |
| 5/21/2024 | 464.630005 | 214.3850341 | 483.64713 |
| 5/22/2024 | 467.779999 | 214.3609144 | 483.6411 |
| 5/23/2024 | 465.779999 | 214.5225517 | 483.63553 |
| 5/24/2024 | 478.220001 | 214.4998861 | 483.63037 |
| 5/28/2024 | 479.920013 | 214.6600807 | 483.62564 |
| 5/29/2024 | 474.359985 | 214.6388466 | 483.6212 |
| 5/30/2024 | 467.049988 | 214.7976209 | 483.61697 |

Table : Mid-Term Stock Price Forecasts (Real vs. Predicted) for META

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Real Adj. Close | ARIMA-GARCH Forecast | GRU Forecast |
| 3/1/2024 | 414.751892 | 277.3374435 | 412.61093 |
| 3/4/2024 | 414.172974 | 278.2081936 | 412.5961 |
| 3/5/2024 | 401.925018 | 277.798426 | 411.83636 |
| 3/6/2024 | 401.366028 | 278.4038513 | 411.70245 |
| 3/7/2024 | 408.403381 | 278.068078 | 411.66315 |
| 3/8/2024 | 405.488617 | 278.6384933 | 411.70847 |
| 3/11/2024 | 403.791656 | 278.3293514 | 411.69836 |
| 3/12/2024 | 414.532288 | 278.8753534 | 411.6664 |
| 3/13/2024 | 414.352631 | 278.5896583 | 411.6245 |
| 3/14/2024 | 424.454407 | 279.1129143 | 411.58896 |
| 3/15/2024 | 415.670258 | 278.8493315 | 411.55884 |
| 3/18/2024 | 416.568634 | 279.3510821 | 411.53198 |
| 3/19/2024 | 420.651276 | 279.1084164 | 411.5061 |
| 3/20/2024 | 424.464386 | 279.5898216 | 411.48096 |
| 3/21/2024 | 428.596924 | 279.3669453 | 411.45663 |
| 3/22/2024 | 427.968048 | 279.8291019 | 411.43356 |
| 3/25/2024 | 422.098633 | 279.6249481 | 411.4116 |
| 3/26/2024 | 420.890808 | 280.068894 | 411.3907 |
| 3/27/2024 | 420.671204 | 279.8824532 | 411.3708 |
| 3/28/2024 | 419.962494 | 280.3091702 | 411.35178 |
| 4/1/2024 | 423.805573 | 280.1394874 | 411.33368 |
| 4/2/2024 | 420.681213 | 280.5499044 | 411.31647 |
| 4/3/2024 | 419.692993 | 280.3960761 | 411.30002 |
| 4/4/2024 | 417.127625 | 280.7910719 | 411.2844 |
| 4/5/2024 | 424.753845 | 280.6522434 | 411.26947 |
| 4/8/2024 | 423.825531 | 281.0326493 | 411.2553 |
| 4/9/2024 | 425.512482 | 280.9080118 | 411.24176 |
| 4/10/2024 | 422.497925 | 281.2746146 | 411.22885 |
| 4/11/2024 | 427.159515 | 281.1634031 | 411.21655 |
| 4/12/2024 | 421.140381 | 281.5169469 | 411.2048 |
| 4/15/2024 | 412.895264 | 281.4184374 | 411.19363 |
| 4/16/2024 | 413.833557 | 281.7596263 | 411.18304 |
| 4/17/2024 | 411.09848 | 281.6731341 | 411.1729 |
| 4/18/2024 | 403.542114 | 282.0026341 | 411.16327 |
| 4/19/2024 | 398.401398 | 281.9275113 | 411.15405 |
| 4/22/2024 | 400.238068 | 282.2459527 | 411.14526 |
| 4/23/2024 | 406.836182 | 282.1815862 | 411.13693 |
| 4/24/2024 | 408.323486 | 282.4895653 | 411.129 |
| 4/25/2024 | 398.321533 | 282.4353753 | 411.1214 |
| 4/26/2024 | 405.58844 | 282.7334559 | 411.11417 |
| 4/29/2024 | 401.525757 | 282.6888938 | 411.1072 |
| 4/30/2024 | 388.628998 | 282.9776098 | 411.10062 |
| 5/1/2024 | 394.228912 | 282.9421563 | 411.09436 |
| 5/2/2024 | 397.123688 | 283.2220125 | 411.0884 |
| 5/3/2024 | 405.927826 | 283.1951767 | 411.08273 |
| 5/6/2024 | 412.795441 | 283.4666508 | 411.07724 |
| 5/7/2024 | 408.602997 | 283.447968 | 411.0721 |
| 5/8/2024 | 409.800842 | 283.7115119 | 411.0672 |
| 5/9/2024 | 411.577637 | 283.7005426 | 411.0625 |
| 5/10/2024 | 413.993256 | 283.9565839 | 411.05804 |
| 5/13/2024 | 412.975098 | 283.9529121 | 411.0538 |
| 5/14/2024 | 415.809998 | 284.2018553 | 411.0497 |
| 5/15/2024 | 423.079987 | 284.2050876 | 411.04587 |
| 5/16/2024 | 420.98999 | 284.4473153 | 411.0422 |
| 5/17/2024 | 420.209991 | 284.4570796 | 411.03867 |
| 5/20/2024 | 425.339996 | 284.6929539 | 411.03534 |
| 5/21/2024 | 429.040009 | 284.708898 | 411.03217 |
| 5/22/2024 | 430.519989 | 284.9387614 | 411.02914 |
| 5/23/2024 | 427 | 284.960552 | 411.02628 |
| 5/24/2024 | 430.160004 | 285.1847287 | 411.02344 |
| 5/28/2024 | 430.320007 | 285.2120507 | 411.02087 |
| 5/29/2024 | 429.170013 | 285.4308472 | 411.01834 |
| 5/30/2024 | 414.670013 | 285.4634023 | 411.01602 |

Table : Mid-Term Stock Price Forecasts (Real vs. Predicted) for Microsoft

1. **Python Code for Time Series Analysis and Forecasting**

1 **import** pandas **as** pd

2 **import** numpy **as** np

3 **import** matplotlib**.***pyplot* **as** plt

4 **import** scipy**.***stats* **as** stats

5 **from** statsmodels**.***tsa***.***stattools* **import** acf**,** pacf**,** adfuller

6 **from** statsmodels**.***stats***.***diagnostic* **import** acorr\_ljungbox

7 **import** os

8 **from** statsmodels**.***tsa***.***stattools* **import** adfuller

9 **from** statsmodels**.***tsa***.***arima***.***model* **import** ARIMA

10 **from** sklearn**.***metrics* **import** mean\_squared\_error**,** mean\_absolute\_error**,** r2\_score

11 **from** fitter **import** Fitter**,** get\_common\_distributions**,** get\_distributions

12 **from** scipy**.***stats* **import** norm**,** expon**,** lognorm**,** gamma**,** beta**,** weibull\_min**,** chi2**,** pareto**,** uniform**,** t**,** gumbel\_r**,** burr**,** invgauss**,** triang**,** laplace**,** logistic

13 **from** statsmodels**.***tsa***.***holtwinters* **import** ExponentialSmoothing

14 **from** arch **import** arch\_model

15 **from** pmdarima **import** auto\_arima

16 **import** tensorflow **as** tf

17 **from** tensorflow**.***keras***.***models* **import** Sequential

18 **from** tensorflow**.***keras***.***layers* **import** SimpleRNN**,** Dense**,** Dropout**,** LSTM**,** GRU

19 **from** sklearn**.***preprocessing* **import** MinMaxScaler

20 **from** sklearn**.***model\_selection* **import** ParameterSampler

21 **from** scipy**.***stats* **import** randint**,** uniform

22 **from** keras**.***models* **import** Sequential

23 **from** keras**.***layers* **import** GRU**,** Dense**,** Dropout

24 **from** pandas**.***tseries***.***offsets* **import** CustomBusinessDay

25

26

27 # Different distributions list

28 distributions **=** **{**

29 'Normal'**:** norm**,**

30 'Exponential'**:** expon**,**

31 'Log-Normal'**:** lognorm**,**

32 'Gamma'**:** gamma**,**

33 'Beta'**:** beta**,**

34 'Weibull\_Min'**:** weibull\_min**,**

35 'Chi-Squared'**:** chi2**,**

36 'Pareto'**:** pareto**,**

37 'Uniform'**:** uniform**,**

38 'T-Distribution'**:** t**,**

39 'Gumbel\_R'**:** gumbel\_r**,**

40 'Burr'**:** burr**,**

41 'Inverse Gaussian'**:** invgauss**,**

42 'Triangular'**:** triang**,**

43 'Laplace'**:** laplace**,**

44 'Logistic'**:** logistic

45

46 **}**

47

48

49 **def** read\_csv\_file**(**filename**,** field\_name**=None,** index\_col**=None,** parse\_dates**=True):**

50 **try:**

51 # Read the CSV file using pandas

52 df **=** pd**.***read\_csv***(**filename**,** index\_col**=**index\_col**,**

53 parse\_dates**=**parse\_dates**)**

54

55 # Check if the specified field exists

56 **if** field\_name **is** **not** **None:**

57 **if** field\_name **not** **in** df**.***columns***:**

58 **raise** **ValueError(**

59 f"Field '{field\_name}' not found in the CSV file."**)**

60 # Return the data from the specified field as a pandas Series

61 **return** df**[**field\_name**]**

62

63 # Return the whole DataFrame if no specific field is requested

64 **return** df

65

66 **except** **FileNotFoundError:**

67 **print(**f"Error: File '{filename}' not found."**)**

68 **except** **ValueError** **as** e**:**

69 **print(**e**)**

70

71

72 # When we want to use all of dataset for trend analysis

73 # stock\_name = 'Historical Alphabet'

74 # stock\_name = 'Historical META'

75 # stock\_name = 'Historical Microsoft'

76

77 # When we don't want to bring the part of data we later want to forecast

78 # stock\_name = 'ALPHABET'

79 # stock\_name = 'META'

80 stock\_name **=** 'MICROSOFT'

81 analyzed\_field **=** 'Adj Close'

82

83 # Load data

84 close\_prices **=** read\_csv\_file**(**"C:/Stock Price lists/" **+** stock\_name **+** ".csv"**,**

85 field\_name**=**analyzed\_field**,** index\_col**=**"Date"**,** parse\_dates**=True)**

86

87 close\_prices**.***index* **=** pd**.***to\_datetime***(**close\_prices**.***index***)**

88

89 # Load the actual next prices data

90 actual\_next\_prices\_filename **=** "C:/Stock Price lists/" **+** \

91 stock\_name **+** "\_actual\_next\_prices.csv"

92 actual\_next\_prices **=** read\_csv\_file**(**

93 actual\_next\_prices\_filename**,**

94 field\_name**=**analyzed\_field**,**

95 index\_col**=**"Date"**,**

96 parse\_dates**=True**

97 **)**

98

99 **if** actual\_next\_prices **is** **not** **None:**

100 actual\_next\_prices**.***index* **=** pd**.***to\_datetime***(**actual\_next\_prices**.***index***)**

101

102 # Saves a pandas DataFrame to a file.

103

104

105 **def** save\_dataframe\_to\_file**(**df**,** filename**,** overwrite**,** index**):**

106

107 **try:**

108 **if** os**.***path***.***exists***(**filename**)** **and** **not** overwrite**:**

109 **print(**

110 f"File {filename} already exists. Set overwrite=True to overwrite it."**)**

111 **return**

112

113 df**.***to\_csv***(**filename**,** index**=**index**)**

114 **print(**f"Data saved to {filename} successfully."**)**

115 **except** **Exception** **as** e**:**

116 **print(**f"Error occurred while saving data: {e}"**)**

117

118

119 **def** mean\_absolute\_percentage\_error**(**y\_true**,** y\_pred**):**

120 **return** np**.***mean***(**np**.abs((**y\_true **-** y\_pred**)** **/** y\_true**))** **\*** 100

121

122

123 **def** do\_descriptive\_analysis**(**filtered\_data**):**

124 **print(**stock\_name **+** ' descriptive analysis:'**)**

125 **print(**filtered\_data**.***describe***())**

126

127

128 # Fit and find the best probability distribution using fitter

129 **def** fit\_best\_distribution**(**filtered\_data**,** show\_best\_fit**,** show\_plot**):**

130 distributions\_to\_fit **=** **[**

131 'norm'**,** 'expon'**,** 'lognorm'**,** 'gamma'**,** 'beta'**,** 'weibull\_min'**,** 'chi2'**,** 'pareto'**,** 'uniform'**,** 't'**,** 'gumbel\_r'**,** 'burr'**,**

132 'invgauss'**,** 'triang'**,** 'laplace'**,** 'logistic'

133 **]**

134

135 f **=** Fitter**(**filtered\_data**,** distributions**=**distributions\_to\_fit**,** timeout**=**600**)**

136 f**.***fit***()**

137

138 # Plot the best fitting distribution

139 f**.***summary***()**

140 # Print the best distribution

141 best\_fit **=** f**.***get\_best***()**

142 best\_dist\_name **=** **list(**best\_fit**.***keys***())[**0**]**

143 **if** show\_best\_fit**:**

144 **print(**"Best fitting distribution:"**,** best\_fit**)**

145

146 **if** show\_plot**:**

147 # Plot the histogram of the data

148 plt**.***figure***(**figsize**=(**12**,** 6**))**

149 plt**.***hist***(**filtered\_data**,** bins**=**30**,** density**=True,**

150 alpha**=**0.6**,** color**=**'g'**,** label**=**'Data'**)**

151

152 # Plot the best fitting distribution

153 x **=** np**.***linspace***(min(**filtered\_data**),** **max(**filtered\_data**),** 1000**)**

154 dist **=** **getattr(**stats**,** best\_dist\_name**)**

155 param **=** best\_fit**[**best\_dist\_name**]**

156

157 **if** best\_dist\_name **==** 'uniform'**:**

158 pdf\_fitted **=** dist**.***pdf***(**x**,** loc**=**param**[**'loc'**],** scale**=**param**[**'scale'**])**

159 **elif** best\_dist\_name **==** 'beta'**:**

160 x\_min**,** x\_max **=** **min(**filtered\_data**),** **max(**filtered\_data**)**

161 x **=** np**.***linspace***(**x\_min**,** x\_max**,** 1000**)**

162 a**,** b**,** loc**,** scale **=** param**[**'a'**],** param**[**'b'**],** param**[**'loc'**],** param**[**'scale'**]**

163 x\_scaled **=** **(**x **-** loc**)** **/** scale

164 pdf\_fitted **=** dist**.***pdf***(**x\_scaled**,** a**,** b**)** **/** scale

165 **else:**

166 pdf\_fitted **=** dist**.***pdf***(**x**,** **\***param**.***values***())**

167

168 plt**.***plot***(**x**,** pdf\_fitted**,** '--'**,** label**=**f'Best fit: {best\_dist\_name}'**)**

169 plt**.***title***(**stock\_name **+** ' Best Fitting Distribution'**)**

170 plt**.***xlabel***(**'Data'**)**

171 plt**.***ylabel***(**'Density'**)**

172 plt**.***legend***()**

173 plt**.***show***()**

174

175 **return** best\_fit

176

177

178 # detect outliers based on the passed distribution

179 **def** detect\_outliers**(**distribution\_name**,** filtered\_data**,** show\_outliers**,** just\_upperbound**,** save\_to\_file**):**

180 distribution **=** distribution\_name

181 match distribution**:**

182 case "norm"**:**

183 # Identify outliers using the fitted Normal distribution

184 normal\_params **=** distributions**[**'Normal'**].***fit***(**filtered\_data**)**

185 mean **=** normal\_params**[**0**]**

186 std\_dev **=** normal\_params**[**1**]**

187

188 # Define outliers as points more than 3 standard deviations from the mean

189 outlier\_threshold **=** 1

190 lower\_bound **=** mean **-** outlier\_threshold **\*** std\_dev

191 upper\_bound **=** mean **+** outlier\_threshold **\*** std\_dev

192

193 # Identify outliers

194 outliers **=** filtered\_data**[(**filtered\_data **<** lower\_bound**)** **|** **(**

195 filtered\_data **>** upper\_bound**)]**

196

197 case "expon"**:**

198 # Identify outliers using the fitted Exponential distribution

199 expon\_params **=** distributions**[**'Exponential'**].***fit***(**filtered\_data**)**

200 loc **=** expon\_params**[**0**]**

201 # scale is 1/lambda for Exponential distribution

202 scale **=** expon\_params**[**1**]**

203

204 # Define outliers as points beyond the 95th percentile

205 upper\_bound **=** expon**.***ppf***(**0.95**,** loc**=**loc**,** scale**=**scale**)**

206

207 # Identify outliers

208 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

209

210 case "lognorm"**:**

211 # Identify outliers using the fitted Log-Normal distribution

212 lognormal\_params **=** distributions**[**'Log-Normal'**].***fit***(**filtered\_data**)**

213 shape **=** lognormal\_params**[**0**]**

214 loc **=** lognormal\_params**[**1**]**

215 scale **=** lognormal\_params**[**2**]**

216

217 # Transform the data to the logarithmic scale

218 log\_transformed\_data **=** np**.***log***(**filtered\_data**)**

219

220 # Fit the Normal distribution to the log-transformed data

221 lognorm\_mean **=** np**.***mean***(**log\_transformed\_data**)**

222 lognorm\_std **=** np**.***std***(**log\_transformed\_data**)**

223

224 # Define outliers as points more than 3 standard deviations from the mean

225 outlier\_threshold **=** 3

226 lower\_bound **=** lognorm\_mean **-** outlier\_threshold **\*** lognorm\_std

227 upper\_bound **=** lognorm\_mean **+** outlier\_threshold **\*** lognorm\_std

228

229 # Transform the bounds back to the original scale

230 lower\_bound\_exp **=** np**.***exp***(**lower\_bound**)**

231 upper\_bound\_exp **=** np**.***exp***(**upper\_bound**)**

232

233 # Identify outliers in the original data scale

234 outliers **=** filtered\_data**[(**filtered\_data **<** lower\_bound\_exp**)** **|** **(**

235 filtered\_data **>** upper\_bound\_exp**)]**

236

237 case "gamma"**:**

238 # Identify outliers using the fitted Gamma distribution

239 gamma\_params **=** distributions**[**'Gamma'**].***fit***(**filtered\_data**)**

240 shape **=** gamma\_params**[**0**]**

241 loc **=** gamma\_params**[**1**]**

242 scale **=** gamma\_params**[**2**]**

243

244 # Define outliers as points beyond the 95th percentile (upper bound)

245 upper\_bound **=** gamma**.***ppf***(**0.95**,** shape**,** loc**=**loc**,** scale**=**scale**)**

246

247 # Identify outliers in the original data scale

248 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

249

250 case "beta"**:**

251 # Identify outliers using the fitted Beta distribution

252 beta\_params **=** distributions**[**'Beta'**].***fit***(**filtered\_data**)**

253 a **=** beta\_params**[**0**]**

254 b **=** beta\_params**[**1**]**

255 loc **=** beta\_params**[**2**]**

256 scale **=** beta\_params**[**3**]**

257

258 # Define outliers as points beyond the 95th percentile (upper bound)

259 upper\_bound **=** beta**.***ppf***(**0.95**,** a**,** b**,** loc**=**loc**,** scale**=**scale**)**

260

261 # Identify outliers in the original data scale

262 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

263

264 case "weibull\_min"**:**

265 # Identify outliers using the fitted Weibull Min distribution

266 weibull\_min\_params **=** distributions**[**'Weibull\_Min'**].***fit***(**

267 filtered\_data**)**

268 c **=** weibull\_min\_params**[**0**]**

269 loc **=** weibull\_min\_params**[**1**]**

270 scale **=** weibull\_min\_params**[**2**]**

271

272 # Define outliers as points beyond the 95th percentile (upper bound)

273 upper\_bound **=** weibull\_min**.***ppf***(**0.95**,** c**,** loc**=**loc**,** scale**=**scale**)**

274

275 # Identify outliers in the original data scale

276 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

277

278 case "chi2"**:**

279 # Identify outliers using the fitted Chi-Squared distribution

280 chi2\_params **=** distributions**[**'Chi-Squared'**].***fit***(**filtered\_data**)**

281 **if** **len(**chi2\_params**)** **==** 3**:**

282 dfr **=** chi2\_params**[**0**]**

283 loc **=** chi2\_params**[**1**]**

284 scale **=** chi2\_params**[**2**]**

285 **else:**

286 dfr **=** chi2\_params**[**0**]**

287 loc **=** 0

288 scale **=** 1

289

290 # Define outliers as points beyond the 95th percentile (upper bound)

291 upper\_bound **=** chi2**.***ppf***(**0.95**,** dfr**,** loc**=**loc**,** scale**=**scale**)**

292

293 # Identify outliers in the original data scale

294 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

295

296 case "pareto"**:**

297 # Identify outliers using the fitted Pareto distribution

298 pareto\_params **=** distributions**[**'Pareto'**].***fit***(**filtered\_data**)**

299 b **=** pareto\_params**[**0**]**

300 loc **=** pareto\_params**[**1**]**

301 scale **=** pareto\_params**[**2**]**

302

303 # Define outliers as points beyond the 95th percentile (upper bound)

304 upper\_bound **=** pareto**.***ppf***(**0.95**,** b**,** loc**=**loc**,** scale**=**scale**)**

305

306 # Identify outliers in the original data scale

307 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

308

309 case "uniform"**:**

310 # Identify outliers using the fitted Uniform distribution

311 uniform\_params **=** distributions**[**'Uniform'**].***fit***(**filtered\_data**)**

312 loc **=** uniform\_params**[**0**]**

313 scale **=** uniform\_params**[**1**]**

314

315 # For a uniform distribution, every point within the range [loc, loc+scale] is considered non-outlier.

316 # We'll define outliers as points outside this range.

317

318 # Calculate lower and upper bounds of the uniform distribution

319 lower\_bound **=** loc

320 upper\_bound **=** loc **+** scale

321

322 # Identify outliers in the original data scale

323 outliers **=** filtered\_data**[(**filtered\_data **<** lower\_bound**)** **|** **(**

324 filtered\_data **>** upper\_bound**)]**

325

326 case "t"**:**

327 # Identify outliers using the fitted T-Distribution

328 t\_params **=** distributions**[**'T-Distribution'**].***fit***(**filtered\_data**)**

329 dfr **=** t\_params**[**0**]** # Degrees of freedom

330 # Location parameter (mean)

331 loc **=** t\_params**[**1**]** **if** **len(**t\_params**)** **>** 1 **else** 0

332 # Scale parameter (standard deviation)

333 scale **=** t\_params**[**2**]** **if** **len(**t\_params**)** **>** 2 **else** 1

334

335 # Define outliers as points beyond the 95th percentile (upper bound)

336 upper\_bound **=** t**.***ppf***(**0.95**,** dfr**,** loc**=**loc**,** scale**=**scale**)**

337

338 # Identify outliers in the original data scale

339 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

340

341 case "gumbel\_r"**:**

342 # Identify outliers using the fitted Gumbel\_R distribution

343 gumbel\_r\_params **=** distributions**[**'Gumbel\_R'**].***fit***(**filtered\_data**)**

344 loc **=** gumbel\_r\_params**[**0**]** # Location parameter

345 scale **=** gumbel\_r\_params**[**1**]** # Scale parameter

346

347 # Define outliers as points beyond the 95th percentile (upper bound)

348 upper\_bound **=** gumbel\_r**.***ppf***(**0.95**,** loc**=**loc**,** scale**=**scale**)**

349

350 # Identify outliers in the original data scale

351 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

352

353 case "triang"**:**

354 # Identify outliers using the fitted Triangular distribution

355 triang\_params **=** distributions**[**'Triangular'**].***fit***(**filtered\_data**)**

356 c **=** triang\_params**[**0**]** # Lower bound

357 loc **=** triang\_params**[**1**]** # Mode

358 scale **=** triang\_params**[**2**]** **-** c # Width

359

360 # Define outliers as points beyond the 95th percentile (upper bound)

361 upper\_bound **=** triang**.***ppf***(**0.95**,** c**,** loc**=**loc**,** scale**=**scale**)**

362

363 # Identify outliers in the original data scale

364 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

365

366 case "laplace"**:**

367 # Identify outliers using the fitted Laplace distribution

368 laplace\_params **=** distributions**[**'Laplace'**].***fit***(**filtered\_data**)**

369 loc **=** laplace\_params**[**0**]** # Location parameter

370 scale **=** laplace\_params**[**1**]** # Scale parameter

371

372 # Define outliers as points beyond the 95th percentile (upper bound)

373 upper\_bound **=** laplace**.***ppf***(**0.95**,** loc**=**loc**,** scale**=**scale**)**

374

375 # Identify outliers in the original data scale

376 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

377

378 case "logistic"**:**

379 # Identify outliers using the fitted Logistic distribution

380 logistic\_params **=** distributions**[**'Logistic'**].***fit***(**filtered\_data**)**

381 loc **=** logistic\_params**[**0**]** # Location parameter

382 scale **=** logistic\_params**[**1**]** # Scale parameter

383

384 # Define outliers as points beyond the 95th percentile (upper bound)

385 upper\_bound **=** logistic**.***ppf***(**0.95**,** loc**=**loc**,** scale**=**scale**)**

386

387 # Identify outliers in the original data scale

388 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

389

390 case "burr"**:**

391 # Identify outliers using the fitted Burr distribution

392 burr\_params **=** distributions**[**'Burr'**].***fit***(**filtered\_data**)**

393 alpha **=** burr\_params**[**0**]** # shape parameter

394 betaa **=** burr\_params**[**1**]** # shape parameter

395 loc **=** burr\_params**[**2**]** # location parameter

396 scale **=** burr\_params**[**3**]** # scale parameter

397

398 # Define outliers as points beyond the 95th percentile (upper bound)

399 upper\_bound **=** burr**.***ppf***(**0.95**,** alpha**,** betaa**,** loc**=**loc**,** scale**=**scale**)**

400

401 # Identify outliers in the original data scale

402 outliers **=** filtered\_data**[**filtered\_data **>** upper\_bound**]**

403

404 case "IQR"**:**

405

406 # Identify outliers using IQR (Interquartile range) method

407 Q1 **=** filtered\_data**.***quantile***(**0.25**)**

408 Q3 **=** filtered\_data**.***quantile***(**0.75**)**

409 IQR **=** Q3 **-** Q1

410

411 # Identify outliers based on 1.5 IQR rule

412 lower\_bound **=** Q1 **-** **(**1.5 **\*** IQR**)**

413 upper\_bound **=** Q3 **+** **(**1.5 **\*** IQR**)**

414

415 # Identify outliers in the filtered DataFrame using original indices

416 **if** **not** just\_upperbound**:**

417 outliers **=** filtered\_data**[(**filtered\_data **<** lower\_bound**)** **|** **(**

418 filtered\_data **>** upper\_bound**)]**

419 **else:**

420 outliers **=** filtered\_data**[(**filtered\_data **>** upper\_bound**)]**

421 case \_**:**

422 **print(**"Unknown Distribution"**)**

423

424 outliers\_df **=** close\_prices**.***loc***[**outliers**.***index***]**

425

426 # Print outliers information

427 **if** show\_outliers**:**

428

429 **print(**"Number of outliers:"**,** **len(**outliers\_df**))**

430

431 **if** **len(**outliers\_df**)** **>** 0**:**

432 **print(**"Outliers:"**)**

433 # Return the entire DataFrame containing outliers

434 **print(**outliers\_df**)**

435

436 **if** save\_to\_file**:**

437 save\_dataframe\_to\_file**(**

438 outliers\_df**,** "C:/Stock Price lists/" **+** stock\_name **+** "\_outliers.csv"**,** overwrite**=True,** index**=True)**

439

440 **return** outliers\_df # Return the DataFrame containing all outliers

441

442

443 # perform adfuller test

444 **def** perform\_adfuller**(**prices**):**

445 result **=** adfuller**(**prices**)**

446 **print(**'ADF Statistic:'**,** result**[**0**])**

447 **print(**'p-value:'**,** result**[**1**])**

448

449 **return** result

450

451

452 **def** perform\_differencing**(**prices**):**

453 prices **=** prices**.***diff***().***dropna***()**

454 **return** prices

455

456

457 **def** Perform\_Ljung\_Box\_test**(**prices**,** lags**=**252**,** return\_df**=True):**

458

459 # Test for seasonality (252 lags for yearly seasonality in daily data)

460 lb\_stats **=** acorr\_ljungbox**(**prices**,** lags**,** return\_df**)**

461

462 # Print test statistic and p-value for each lag

463 **print(**"Ljung-Box Test Statistics and p-values:"**)**

464 **print(**lb\_stats**)**

465

466 save\_dataframe\_to\_file**(**

467 lb\_stats**,** "C:/Stock Price lists/" **+** stock\_name **+** "\_lb\_stats.csv"**,** overwrite**=True,** index**=False)**

468

469

470 **def** Calculate\_correlogram\_acf\_pacf**(**prices**,** nlags**=**252**):**

471 # Calculate lags up to 252 days (one trading year)

472 plt**.***tight\_layout***()**

473 plt**.***show***()**

474 plt**.***figure***(**figsize**=(**12**,** 6**))**

475 plt**.***subplot***(**121**)**

476 plt**.***plot***(**acf**(**prices**,** nlags**=**252**))**

477 plt**.***title***(**'Autocorrelation'**)**

478 plt**.***subplot***(**122**)**

479 plt**.***plot***(**pacf**(**prices**,** nlags**=**252**))**

480 plt**.***title***(**'Partial Autocorrelation'**)**

481 plt**.***show***()**

482

483

484 **def** draw\_Original\_Trend**(**prices**,** chart\_type\_title**):**

485 # Plot the original series

486 plt**.***figure***(**figsize**=(**12**,** 6**))**

487 plt**.***plot***(**prices**)**

488 plt**.***title***(**chart\_type\_title **+** ' ' **+** stock\_name **+** ' Stock Prices'**)**

489 plt**.***xlabel***(**'Date'**)**

490 plt**.***ylabel***(**'Adjusted Close Price'**)**

491 plt**.***show***()**

492

493

494 **def** fit\_sma**(**prices**,** window\_size**,** show\_plot**=False,** print\_result**=False,** save\_to\_file**=False):**

495 # Calculate the window\_size-day (for example 100-day) moving average

496 sma **=** prices**.***rolling***(**window**=**window\_size**).***mean***()**

497

498 **if** show\_plot**:**

499 # Plot the original prices and the moving average

500 plt**.***figure***(**figsize**=(**12**,** 6**))**

501 plt**.***plot***(**prices**,** label**=**'Original Prices'**)**

502 plt**.***plot***(**sma**,** label**=str(**window\_size**)** **+**

503 '-day Moving Average'**,** color**=**'orange'**)**

504 plt**.***title***(**stock\_name **+** ' Stock Prices with ' **+**

505 **str(**window\_size**)** **+** '-day Moving Average'**)**

506 plt**.***xlabel***(**'Date'**)**

507 plt**.***ylabel***(**'Adjusted Close Price'**)**

508 plt**.***legend***()**

509 plt**.***show***()**

510

511 **if** save\_to\_file**:**

512 save\_dataframe\_to\_file**(**

513 sma**,** "C:/Stock Price lists/" **+** stock\_name **+** "\_sma.csv"**,** overwrite**=True,** index**=True)**

514

515 **return** sma

516

517

518 **def** evaluate\_vs\_baseline\_sma**(**prices**,** stock\_name**,** window\_size**,** test\_size**=**0.2**):**

519 # Ensure the prices data has a DateTime index

520 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

521 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

522

523 # Split the data into training and test sets

524 split\_index **=** **int(len(**prices**)** **\*** **(**1 **-** test\_size**))**

525 train**,** test **=** prices**[:**split\_index**],** prices**[**split\_index**:]**

526 window\_size **=** window\_size

527 sma **=** fit\_sma**(**train**,** window\_size**,** show\_plot**=False,**

528 print\_result**=False,** save\_to\_file**=False)**

529

530 # Use the last SMA value from the training period to forecast the test period

531 last\_sma\_value **=** sma**.***iloc***[-**1**]**

532 sma\_baseline **=** pd**.***Series***(**last\_sma\_value**,** index**=**test**.***index***)**

533

534 # Calculate MSE, RMSE, MAE, MAPE, and R² for the SMA baseline

535 mse **=** mean\_squared\_error**(**test**,** sma\_baseline**)**

536 rmse **=** np**.***sqrt***(**mse**)**

537 mae **=** mean\_absolute\_error**(**test**,** sma\_baseline**)**

538 mape **=** mean\_absolute\_percentage\_error**(**test**,** sma\_baseline**)**

539 r2 **=** r2\_score**(**test**,** sma\_baseline**)**

540

541 **print(**f"Mean Squared Error: {mse}"**)**

542 **print(**f"Root Mean Squared Error: {rmse}"**)**

543 **print(**f"Mean Absolute Error: {mae}"**)**

544 **print(**f"Mean Absolute Percentage Error: {mape}%"**)**

545 **print(**f"R-squared: {r2}"**)**

546

547 # Plot the actual prices and the SMA baseline

548 plt**.***figure***(**figsize**=(**14**,** 7**))**

549 plt**.***plot***(**train**.***index***,** train**,** label**=**'Training Data'**)**

550 plt**.***plot***(**test**.***index***,** test**,** label**=**'Actual Prices'**)**

551 plt**.***plot***(**test**.***index***,** sma\_baseline**,** label**=**'SMA Baseline'**,** linestyle**=**'--'**)**

552

553 plt**.***title***(**f'{stock\_name} ' **+**

554 **str(**window\_size**)** **+** '-day SMA Baseline'**)**

555 plt**.***xlabel***(**'Date'**)**

556 plt**.***ylabel***(**'Price'**)**

557 plt**.***legend***()**

558 plt**.***show***()**

559

560

561 **def** fit\_ema**(**prices**,** window\_size**,** show\_plot**=False,** print\_result**=False,** save\_to\_file**=False):**

562 # Calculate the window\_size-day (for example 100-day) exponential moving average

563 ema **=** prices**.***ewm***(**span**=**window\_size**,** adjust**=False).***mean***()**

564

565 **if** show\_plot**:**

566 # Plot the original prices and the moving average

567 plt**.***figure***(**figsize**=(**12**,** 6**))**

568 plt**.***plot***(**prices**,** label**=**'Original Prices'**)**

569 plt**.***plot***(**

570 ema**,** label**=**f'{window\_size}-day Exponential Moving Average'**,** color**=**'orange'**)**

571 plt**.***title***(**f'{stock\_name} ' **+**

572 **str(**window\_size**)** **+** '-day Exponential Moving Average'**)**

573 plt**.***xlabel***(**'Date'**)**

574 plt**.***ylabel***(**'Adjusted Close Price'**)**

575 plt**.***legend***()**

576 plt**.***show***()**

577

578 **if** print\_result**:**

579 **print(**f"Exponential Moving Average (window size {window\_size}): {ema}"**)**

580

581 **if** save\_to\_file**:**

582 ema**.***to\_csv***(**

583 f"C:/Stock Price lists/{stock\_name}\_ema.csv"**,** index**=True)**

584

585 **return** ema

586

587

588 **def** evaluate\_vs\_baseline\_ema**(**prices**,** stock\_name**,** window\_size**,** test\_size**=**0.2**):**

589 # Ensure the prices data has a DateTime index

590 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

591 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

592

593 # Split the data into training and test sets

594 split\_index **=** **int(len(**prices**)** **\*** **(**1 **-** test\_size**))**

595 train**,** test **=** prices**[:**split\_index**],** prices**[**split\_index**:]**

596 ema **=** fit\_ema**(**train**,** window\_size**,** show\_plot**=False,**

597 print\_result**=False,** save\_to\_file**=False)**

598

599 # Use the last EMA value from the training period to forecast the test period

600 last\_ema\_value **=** ema**.***iloc***[-**1**]**

601 ema\_baseline **=** pd**.***Series***(**last\_ema\_value**,** index**=**test**.***index***)**

602

603 # Calculate MSE and RMSE for the EMA baseline

604 mse **=** mean\_squared\_error**(**test**,** ema\_baseline**)**

605 rmse **=** np**.***sqrt***(**mse**)**

606 mae **=** mean\_absolute\_error**(**test**,** ema\_baseline**)**

607 mape **=** mean\_absolute\_percentage\_error**(**test**,** ema\_baseline**)**

608 r2 **=** r2\_score**(**test**,** ema\_baseline**)**

609

610 **print(**f"Mean Squared Error: {mse}"**)**

611 **print(**f"Root Mean Squared Error: {rmse}"**)**

612 **print(**f"Mean Absolute Error: {mae}"**)**

613 **print(**f"Mean Absolute Percentage Error: {mape}%"**)**

614 **print(**f"R-squared: {r2}"**)**

615

616 # Plot the actual prices and the EMA baseline

617 plt**.***figure***(**figsize**=(**14**,** 7**))**

618 plt**.***plot***(**train**.***index***,** train**,** label**=**'Training Data'**)**

619 plt**.***plot***(**test**.***index***,** test**,** label**=**'Actual Prices'**)**

620 plt**.***plot***(**test**.***index***,** ema\_baseline**,** label**=**'EMA Baseline'**,** linestyle**=**'--'**)**

621

622 plt**.***title***(**f'{stock\_name} {window\_size}-day EMA Baseline'**)**

623 plt**.***xlabel***(**'Date'**)**

624 plt**.***ylabel***(**'Price'**)**

625 plt**.***legend***()**

626 plt**.***show***()**

627

628

629 **class** **ExponentialSmoothingTimeSeriesPredictor:**

630 **def** \_\_init\_\_**(**self**,** prices**,** stock\_name**,** test\_size**=**0.2**,** seasonal\_periods**=None,** confidence\_level**=**0.95**):**

631 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

632 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

633

634 self**.***prices* **=** prices

635 self**.***stock\_name* **=** stock\_name

636 self**.***test\_size* **=** test\_size

637 self**.***seasonal\_periods* **=** seasonal\_periods

638 self**.***confidence\_level* **=** confidence\_level

639

640 self**.***split\_index* **=** **int(len(**prices**)** **\*** **(**1 **-** test\_size**))**

641 self**.***train* **=** prices**[:**self**.***split\_index***]**

642 self**.***test* **=** prices**[**self**.***split\_index***:]**

643 self**.***model\_fit* **=** **None**

644

645 **def** fit\_model**(**self**):**

646 **try:**

647 model **=** ExponentialSmoothing**(**

648 self**.***train***,** trend**=**'add'**,** seasonal**=None,** seasonal\_periods**=**self**.***seasonal\_periods***)**

649 self**.***model\_fit* **=** model**.***fit***()**

650 **print(**self**.***model\_fit***.***summary***())**

651 **except** **Exception** **as** e**:**

652 **print(**"Error in fitting Exponential Smoothing model:"**,** e**)**

653

654 **def** forecast**(**self**,** steps**):**

655 **if** self**.***model\_fit* **is** **None:**

656 **raise** **ValueError(**

657 "Model is not fitted yet. Call fit\_model() first."**)**

658

659 **try:**

660 forecast\_values **=** self**.***model\_fit***.***forecast***(**steps**)**

661 **except** **Exception** **as** e**:**

662 **print(**"Error in forecasting:"**,** e**)**

663 **return** **None**

664

665 **return** forecast\_values

666

667 **def** save\_dataframe\_to\_file**(**self**,** df**,** filename**,** overwrite**=True,** index**=False):**

668 **if** overwrite **or** **not** os**.***path***.***exists***(**filename**):**

669 df**.***to\_csv***(**filename**,** index**=**index**)**

670 **else:**

671 **print(**

672 f"File {filename} already exists. Set overwrite=True to overwrite it."**)**

673

674 **def** plot\_forecast**(**self**,** forecast\_series**,** lower\_conf\_int**,** upper\_conf\_int**):**

675 plt**.***figure***(**figsize**=(**12**,** 6**))**

676 plt**.***plot***(**self**.***train***,** label**=**'Train'**)**

677 plt**.***plot***(**self**.***test***,** label**=**'Test'**)**

678 plt**.***plot***(**forecast\_series**,** label**=**'Forecast'**,** color**=**'red'**)**

679 plt**.***fill\_between***(**self**.***test***.***index***,** lower\_conf\_int**,**

680 upper\_conf\_int**,** color**=**'pink'**,** alpha**=**0.3**)**

681 plt**.***title***(**

682 f'{self**.***stock\_name*} Stock Prices - Train, Test and Forecast (Exponential Smoothing)'**)**

683 plt**.***xlabel***(**'Date'**)**

684 plt**.***ylabel***(**'Adjusted Close Price'**)**

685 plt**.***legend***()**

686 plt**.***show***()**

687

688 **def** predict\_and\_evaluate**(**self**):**

689 steps **=** **len(**self**.***test***)**

690 forecast\_values **=** self**.***forecast***(**steps**)**

691

692 **if** forecast\_values **is** **None:**

693 **return**

694

695 forecast\_series **=** pd**.***Series***(**

696 forecast\_values**.***values***,** index**=**self**.***test***.***index***)**

697

698 residuals **=** self**.***train* **-** self**.***model\_fit***.***fittedvalues*

699 residual\_std **=** residuals**.***std***()**

700 z **=** 1.96 # For a 95% confidence interval

701 lower\_conf\_int **=** forecast\_values **-** z **\*** residual\_std

702 upper\_conf\_int **=** forecast\_values **+** z **\*** residual\_std

703

704 # Debugging prints to check the values

705 **print(**"Forecast values:\n"**,** forecast\_values**)**

706 **print(**"Lower confidence interval:\n"**,** lower\_conf\_int**)**

707 **print(**"Upper confidence interval:\n"**,** upper\_conf\_int**)**

708

709 forecast\_data **=** pd**.***DataFrame***({**

710 'Forecast'**:** forecast\_values**,**

711 'Lower\_CI'**:** lower\_conf\_int**,**

712 'Upper\_CI'**:** upper\_conf\_int

713 **},** index**=**self**.***test***.***index***)**

714

715 # Ensure the forecast\_data does not contain NaNs

716 **print(**"Forecast data before saving:\n"**,** forecast\_data**)**

717

718 self**.***save\_dataframe\_to\_file***(**

719 forecast\_data**,** f"C:/Stock Price lists/{self**.***stock\_name*}\_Exponential\_Smoothing\_Predicted.csv"**,** overwrite**=True,** index**=True)**

720

721 # Align the indices and remove NaN values from both series

722 aligned\_test**,** aligned\_forecast **=** self**.***test***.***align***(**

723 forecast\_series**,** join**=**'inner'**)**

724 aligned\_test **=** aligned\_test**.***dropna***()**

725 aligned\_forecast **=** aligned\_forecast**.***dropna***()**

726

727 mse **=** mean\_squared\_error**(**aligned\_test**,** aligned\_forecast**)**

728 rmse **=** np**.***sqrt***(**mse**)**

729 mae **=** mean\_absolute\_error**(**aligned\_test**,** aligned\_forecast**)**

730 mape **=** mean\_absolute\_percentage\_error**(**aligned\_test**,** aligned\_forecast**)**

731 r2 **=** r2\_score**(**aligned\_test**,** aligned\_forecast**)**

732

733 **print(**f"Mean Squared Error: {mse}"**)**

734 **print(**f"Root Mean Squared Error: {rmse}"**)**

735 **print(**f"Mean Absolute Error: {mae}"**)**

736 **print(**f"Mean Absolute Percentage Error: {mape}%"**)**

737 **print(**f"R-squared: {r2}"**)**

738

739 self**.***plot\_forecast***(**forecast\_series**,** lower\_conf\_int**,** upper\_conf\_int**)**

740

741 **return** forecast\_data

742

743 **def** compare\_with\_real\_data**(**self**,** real\_data**):**

744 **if** **not** **isinstance(**real\_data**.***index***,** pd**.***DatetimeIndex***):**

745 **raise** **ValueError(**"The real\_data series must be indexed by dates"**)**

746

747 **if** self**.***model\_fit* **is** **None:**

748 **raise** **ValueError(**

749 "Model is not fitted yet. Call fit\_model() first."**)**

750

751 steps **=** **len(**real\_data**)**

752 forecast\_values **=** self**.***forecast***(**steps**)**

753 **if** forecast\_values **is** **None:**

754 **return**

755

756 forecast\_series **=** pd**.***Series***(**

757 forecast\_values**.***values***,** index**=**real\_data**.***index***)**

758 residuals **=** real\_data **-** forecast\_series

759

760 # Align the indices and remove NaN values from both series

761 aligned\_real**,** aligned\_forecast **=** real\_data**.***align***(**

762 forecast\_series**,** join**=**'inner'**)**

763 aligned\_real **=** aligned\_real**.***dropna***()**

764 aligned\_forecast **=** aligned\_forecast**.***dropna***()**

765

766 mse **=** mean\_squared\_error**(**aligned\_real**,** aligned\_forecast**)**

767 rmse **=** np**.***sqrt***(**mse**)**

768 mae **=** mean\_absolute\_error**(**aligned\_real**,** aligned\_forecast**)**

769 mape **=** mean\_absolute\_percentage\_error**(**aligned\_real**,** aligned\_forecast**)**

770 r2 **=** r2\_score**(**aligned\_real**,** aligned\_forecast**)**

771

772 **print(**f"Mean Squared Error against real data: {mse}"**)**

773 **print(**f"Root Mean Squared Error against real data: {rmse}"**)**

774 **print(**f"Mean Absolute Error against real data: {mae}"**)**

775 **print(**f"Mean Absolute Percentage Error: {mape}%"**)**

776 **print(**f"R-squared: {r2}"**)**

777

778 plt**.***figure***(**figsize**=(**12**,** 6**))**

779 plt**.***plot***(**real\_data**,** label**=**'Real Data'**)**

780 plt**.***plot***(**forecast\_series**,** label**=**'Forecast'**,** color**=**'red'**)**

781 plt**.***title***(**f'Comparison of Real Data and Forecast (Exponential Smoothing)'**)**

782 plt**.***xlabel***(**'Date'**)**

783 plt**.***ylabel***(**'Adjusted Close Price'**)**

784 plt**.***legend***()**

785 plt**.***show***()**

786

787

788 **def** fit\_exponential\_Smoothing\_forecaster\_class**(**prices**,** stock\_name**,** forecasting\_Duration**,** test\_size**=**0.2**,** seasonal\_periods**=None,** confidence\_level**=**0.95**,** predict\_future\_value**=False):**

789

790 # Initialize the predictor

791 predictor **=** ExponentialSmoothingTimeSeriesPredictor**(**

792 prices**,** stock\_name**,** test\_size**,** seasonal\_periods**,** confidence\_level**)**

793

794 # Fit the model

795 predictor**.***fit\_model***()**

796

797 # Predict and evaluate using the length of the test set

798 forecast\_data **=** predictor**.***predict\_and\_evaluate***()**

799

800 **if** predict\_future\_value**:**

801 # Predict for a fixed future duration (e.g., 90 days)

802 forecast\_duration **=** predictor**.***forecast***(**steps**=**forecasting\_Duration**)**

803 **if** forecast\_duration **is** **not** **None:**

804 forecast\_index\_n\_days **=** pd**.***date\_range***(**

805 start**=**prices**.***index***[-**1**],** periods**=**forecasting\_Duration **+** 1**,** freq**=**'D'**)[**1**:]**

806 forecast\_series\_n\_days **=** pd**.***Series***(**

807 forecast\_duration**.***values***,** index**=**forecast\_index\_n\_days**)**

808

809 plt**.***figure***(**figsize**=(**12**,** 6**))**

810 plt**.***plot***(**prices**,** label**=**'Original Data'**)**

811 plt**.***plot***(**forecast\_series\_n\_days**,** label**=str(**

812 forecasting\_Duration**)** **+** ' Days Forecast'**,** color**=**'red'**)**

813 plt**.***title***(**

814 f'{stock\_name} Stock Prices - ' **+** **str(**forecasting\_Duration**)** **+** ' Days Forecast (Exponential Smoothing)'**)**

815 plt**.***xlabel***(**'Date'**)**

816 plt**.***ylabel***(**'Adjusted Close Price'**)**

817 plt**.***legend***()**

818 plt**.***show***()**

819

820

821 **def** fit\_ARIMA**(**prices**,** Arima\_order**,** test\_size**=**0.2**):**

822 # Fit ARIMA model to original series (if trend is detected)

823

824 # Ensure the prices data has a DateTime index

825 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

826 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

827

828 # Split the data into training and test sets

829 split\_index **=** **int(len(**prices**)** **\*** **(**1 **-** test\_size**))**

830 train**,** test **=** prices**[:**split\_index**],** prices**[**split\_index**:]**

831

832 # Adjust the order based on ACF/PACF analysis

833 # Fit ARIMA model to original series

834 **try:**

835 model **=** ARIMA**(**train**,** order**=**Arima\_order**)**

836 model\_fit **=** model**.***fit***()**

837 **print(**model\_fit**.***summary***())**

838 **except** **Exception** **as** e**:**

839 **print(**"Error in fitting ARIMA model:"**,** e**)**

840 **return**

841

842 # Forecast future values

843 steps **=** **len(**test**)**

844 **try:**

845 forecast **=** model\_fit**.***get\_forecast***(**steps**=**steps**)**

846 forecast\_values **=** forecast**.***predicted\_mean*

847 forecast\_conf\_int **=** forecast**.***conf\_int***()**

848 **except** **Exception** **as** e**:**

849 **print(**"Error in forecasting:"**,** e**)**

850 **return**

851

852 forecast\_series **=** pd**.***Series***(**forecast\_values**.***values***,** index**=**test**.***index***)**

853

854 # Combine forecasted values and confidence intervals into a single DataFrame

855 Complete\_forecast\_data **=** pd**.***DataFrame***({**

856 'Date'**:** test**.***index***,**

857 'Forecast'**:** forecast\_values**,**

858 # Lower bound of confidence interval

859 'Lower\_CI'**:** forecast\_conf\_int**.***iloc***[:,** 0**],**

860 # Upper bound of confidence interval

861 'Upper\_CI'**:** forecast\_conf\_int**.***iloc***[:,** 1**]**

862 **})**

863 save\_dataframe\_to\_file**(**Complete\_forecast\_data**,** "C:/Stock Price lists/" **+**

864 stock\_name **+** "\_ARIMA\_Predicted.csv"**,** overwrite**=True,** index**=True)**

865

866 # Debugging statements to check the forecast series

867 **print(**"Train data:"**)**

868 **print(**train**.***tail***())**

869 **print(**"test data:"**)**

870 **print(**test**.***head***)**

871 **print(**"Forecast data:"**)**

872 **print(**forecast\_series**)**

873 **print(**"Forecast confidence intervals:"**)**

874 **print(**forecast\_conf\_int**)**

875

876 # Calculate and print the mean squared error, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE)

877 mse **=** mean\_squared\_error**(**test**,** forecast\_series**)**

878 rmse **=** np**.***sqrt***(**mse**)**

879 mae **=** mean\_absolute\_error**(**test**,** forecast\_series**)**

880

881 **print(**f"Mean Squared Error: {mse}"**)**

882 **print(**f"Root Mean Squared Error: {rmse}"**)**

883 **print(**f"Mean Absolute Error: {mae}"**)**

884

885 # Plot original, train, test, and forecasted series

886 plt**.***figure***(**figsize**=(**12**,** 6**))**

887 plt**.***plot***(**train**,** label**=**'Train'**)**

888 plt**.***plot***(**test**,** label**=**'Test'**)**

889 plt**.***plot***(**forecast\_series**,** label**=**'Forecast'**,** color**=**'red'**)**

890 plt**.***fill\_between***(**test**.***index***,**

891 forecast\_conf\_int**.***iloc***[:,** 0**],**

892 forecast\_conf\_int**.***iloc***[:,** 1**],**

893 color**=**'pink'**,** alpha**=**0.3**)**

894 plt**.***title***(**

895 stock\_name **+** ' Stock Prices - Train, Test and Forecast (ARIMA' **+** **str(**Arima\_order**)** **+** ')'**)**

896 plt**.***xlabel***(**'Date'**)**

897 plt**.***ylabel***(**'Adjusted Close Price'**)**

898 plt**.***legend***()**

899 plt**.***show***()**

900

901 # Use auto\_arima to find the best ARIMA order

902

903

904 **def** automatic\_arima**(**prices**,** max\_d**=**3**):**

905 stepwise\_fit **=** auto\_arima**(**prices**,** start\_p**=**1**,** start\_q**=**1**,** max\_p**=**5**,** max\_q**=**5**,** max\_d**=**max\_d**,** seasonal**=False,**

906 trace**=True,** error\_action**=**'ignore'**,** suppress\_warnings**=True,** stepwise**=True)**

907 **print(**stepwise\_fit**.***summary***())**

908

909 # Use the best found order for ARIMA

910 best\_arima\_order **=** stepwise\_fit**.***order*

911 **print(**best\_arima\_order**)**

912

913

914 **def** fit\_ARIMA\_GARCH**(**prices**,** Arima\_order**,** Garch\_order**=(**1**,** 1**),** test\_size**=**0.2**):**

915 # Ensure the prices data has a DateTime index and set frequency

916 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

917 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

918

919 # Assuming business day frequency, adjust as needed

920 prices **=** prices**.***asfreq***(**'B'**)**

921 prices **=** prices**.***ffill***()** # Fill missing values if any

922

923 # Split the data into training and test sets

924 split\_index **=** **int(len(**prices**)** **\*** **(**1 **-** test\_size**))**

925 train**,** test **=** prices**[:**split\_index**],** prices**[**split\_index**:]**

926

927 # Fit ARIMA model to original series

928 **try:**

929 arima\_model **=** ARIMA**(**train**,** order**=**Arima\_order**)**

930 arima\_model\_fit **=** arima\_model**.***fit***()**

931 **print(**arima\_model\_fit**.***summary***())**

932 **except** **Exception** **as** e**:**

933 **print(**"Error in fitting ARIMA model:"**,** e**)**

934 **return**

935

936 # Extract residuals from ARIMA model

937 residuals **=** arima\_model\_fit**.***resid*

938

939 # Ensure no NaN or infinite values in residuals

940 residuals **=** residuals**.***replace***([**np**.***inf***,** **-**np**.***inf***],** np**.***nan***).***dropna***()**

941

942 # Fit GARCH model to the residuals

943 **try:**

944 garch\_model **=** arch\_model**(**

945 residuals**,** vol**=**'Garch'**,** p**=**Garch\_order**[**0**],** q**=**Garch\_order**[**1**])**

946 garch\_model\_fit **=** garch\_model**.***fit***(**disp**=**"off"**)**

947 **print(**garch\_model\_fit**.***summary***())**

948 **except** **Exception** **as** e**:**

949 **print(**"Error in fitting GARCH model:"**,** e**)**

950 **return**

951

952 # Forecast future values of ARIMA

953 steps **=** **len(**test**)**

954 **try:**

955 forecast\_ARIMA **=** arima\_model\_fit**.***get\_forecast***(**steps**=**steps**)**

956 forecast\_mean\_ARIMA **=** forecast\_ARIMA**.***predicted\_mean*

957 forecast\_conf\_int\_ARIMA **=** forecast\_ARIMA**.***conf\_int***()**

958 **except** **Exception** **as** e**:**

959 **print(**"Error in forecasting:"**,** e**)**

960 **return**

961

962 # Forecast future values of GARCH

963 **try:**

964 forecast\_GARCH **=** garch\_model\_fit**.***forecast***(**horizon**=**steps**)**

965 forecast\_mean\_GARCH **=** forecast\_GARCH**.***mean***.***iloc***[-**steps**:].***values***.***flatten***(**

966 **)**

967 forecast\_variance\_GARCH **=** forecast\_GARCH**.***variance***.***iloc***[-**steps**:].***values***.***flatten***(**

968 **)**

969 **except** **Exception** **as** e**:**

970 **print(**"Error in forecasting with GARCH model:"**,** e**)**

971 **return**

972

973 # Ensure forecast has produced the expected results

974 **if** **len(**forecast\_mean\_GARCH**)** **==** 0 **or** **len(**forecast\_variance\_GARCH**)** **==** 0**:**

975 **print(**"Forecast mean or variance is empty."**)**

976 **return**

977

978 # Ensure the lengths of forecasted series match the length of the test data

979 min\_length **=** **min(len(**forecast\_mean\_ARIMA**),**

980 **len(**forecast\_mean\_GARCH**),** **len(**test**))**

981 forecast\_mean\_ARIMA **=** forecast\_mean\_ARIMA**[:**min\_length**]**

982 forecast\_mean\_GARCH **=** forecast\_mean\_GARCH**[:**min\_length**]**

983 forecast\_conf\_int\_ARIMA **=** forecast\_conf\_int\_ARIMA**[:**min\_length**]**

984 test **=** test**[:**min\_length**]**

985

986 # Create combined forecast series

987 combined\_forecast\_mean **=** forecast\_mean\_ARIMA **+** forecast\_mean\_GARCH

988 combined\_forecast\_lower\_CI **=** forecast\_conf\_int\_ARIMA**.***iloc***[:,** 0**]** **+** **(**

989 forecast\_mean\_GARCH **-** 1.96 **\*** np**.***sqrt***(**forecast\_variance\_GARCH**))**

990 combined\_forecast\_upper\_CI **=** forecast\_conf\_int\_ARIMA**.***iloc***[:,** 1**]** **+** **(**

991 forecast\_mean\_GARCH **+** 1.96 **\*** np**.***sqrt***(**forecast\_variance\_GARCH**))**

992

993 # Ensure the combined forecast and test series have the same index

994 combined\_forecast\_mean**.***index* **=** test**.***index*

995 combined\_forecast\_lower\_CI**.***index* **=** test**.***index*

996 combined\_forecast\_upper\_CI**.***index* **=** test**.***index*

997

998 # Combine forecasted values and confidence intervals into a single DataFrame

999 ARIMA\_GARCH\_Model\_Prediction **=** pd**.***DataFrame***({**

1000 'Date'**:** test**.***index***,**

1001 'Forecast'**:** combined\_forecast\_mean**,**

1002 'Lower\_CI'**:** combined\_forecast\_lower\_CI**,**

1003 'Upper\_CI'**:** combined\_forecast\_upper\_CI

1004 **})**

1005

1006 # Save forecast data to a CSV file

1007 ARIMA\_GARCH\_Model\_Prediction**.***to\_csv***(**

1008 "C:/Stock Price lists/" **+** stock\_name **+** "\_ARIMA\_GARCH\_Predicted.csv"**,** index**=False)**

1009

1010 # Calculate and print the mean squared error, Root Mean Squared Error (RMSE),Mean Absolute Error (MAE)

1011 mse **=** mean\_squared\_error**(**test**,** combined\_forecast\_mean**)**

1012 rmse **=** np**.***sqrt***(**mse**)**

1013 mae **=** mean\_absolute\_error**(**test**,** combined\_forecast\_mean**)**

1014 **print(**f"Mean Squared Error: {mse}"**)**

1015 **print(**f"Root Mean Squared Error: {rmse}"**)**

1016 **print(**f"Mean Absolute Error: {mae}"**)**

1017

1018 # Plot original, train, test, and forecasted series

1019 plt**.***figure***(**figsize**=(**12**,** 6**))**

1020 plt**.***plot***(**train**,** label**=**'Train'**)**

1021 plt**.***plot***(**test**,** label**=**'Test'**)**

1022 plt**.***plot***(**combined\_forecast\_mean**.***index***,** combined\_forecast\_mean**,**

1023 label**=**'Forecast'**,** color**=**'red'**)**

1024 plt**.***fill\_between***(**combined\_forecast\_mean**.***index***,**

1025 combined\_forecast\_lower\_CI**,** combined\_forecast\_upper\_CI**,** color**=**'gray'**,** alpha**=**0.3**)**

1026 plt**.***title***(**'Stock Prices - Train, Test and Forecast (ARIMA' **+**

1027 **str(**Arima\_order**)** **+** '-GARCH' **+** **str(**Garch\_order**)** **+** ')'**)**

1028 plt**.***xlabel***(**'Date'**)**

1029 plt**.***ylabel***(**'Adjusted Close Price'**)**

1030 plt**.***legend***()**

1031 plt**.***show***()**

1032

1033

1034class ARIMAGARCHForecaster**:**

1035 **def** \_\_init\_\_**(**self**,** arima\_order**,** garch\_order**=(**1**,** 1**),** test\_size**=**0.2**):**

1036 self**.***arima\_order* **=** arima\_order

1037 self**.***garch\_order* **=** garch\_order

1038 self**.***test\_size* **=** test\_size

1039 self**.***arima\_model\_fit* **=** **None**

1040 self**.***garch\_model\_fit* **=** **None**

1041

1042 **def** fit**(**self**,** prices**):**

1043 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

1044 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

1045

1046 prices **=** prices**.***asfreq***(**'B'**)**

1047 prices **=** prices**.***ffill***()**

1048

1049 split\_index **=** **int(len(**prices**)** **\*** **(**1 **-** self**.***test\_size***))**

1050 train**,** test **=** prices**[:**split\_index**],** prices**[**split\_index**:]**

1051

1052 **try:**

1053 arima\_model **=** ARIMA**(**train**,** order**=**self**.***arima\_order***)**

1054 self**.***arima\_model\_fit* **=** arima\_model**.***fit***()**

1055 **print(**self**.***arima\_model\_fit***.***summary***())**

1056 **except** **Exception** **as** e**:**

1057 **print(**"Error in fitting ARIMA model:"**,** e**)**

1058 **return**

1059

1060 residuals **=** self**.***arima\_model\_fit***.***resid*

1061 residuals **=** residuals**.***replace***([**np**.***inf***,** **-**np**.***inf***],** np**.***nan***).***dropna***()**

1062

1063 **try:**

1064 garch\_model **=** arch\_model**(**

1065 residuals**,** vol**=**'Garch'**,** p**=**self**.***garch\_order***[**0**],** q**=**self**.***garch\_order***[**1**])**

1066 self**.***garch\_model\_fit* **=** garch\_model**.***fit***(**disp**=**"off"**)**

1067 **print(**self**.***garch\_model\_fit***.***summary***())**

1068 **except** **Exception** **as** e**:**

1069 **print(**"Error in fitting GARCH model:"**,** e**)**

1070 **return**

1071

1072 steps **=** **len(**test**)**

1073 **try:**

1074 forecast\_ARIMA **=** self**.***arima\_model\_fit***.***get\_forecast***(**steps**=**steps**)**

1075 forecast\_mean\_ARIMA **=** forecast\_ARIMA**.***predicted\_mean*

1076 forecast\_conf\_int\_ARIMA **=** forecast\_ARIMA**.***conf\_int***()**

1077 **except** **Exception** **as** e**:**

1078 **print(**"Error in forecasting:"**,** e**)**

1079 **return**

1080

1081 **try:**

1082 forecast\_GARCH **=** self**.***garch\_model\_fit***.***forecast***(**horizon**=**steps**)**

1083 forecast\_mean\_GARCH **=** forecast\_GARCH**.***mean***.***iloc***[-**steps**:].***values***.***flatten***(**

1084 **)**

1085 forecast\_variance\_GARCH **=** forecast\_GARCH**.***variance***.***iloc***[-**steps**:].***values***.***flatten***(**

1086 **)**

1087 **except** **Exception** **as** e**:**

1088 **print(**"Error in forecasting with GARCH model:"**,** e**)**

1089 **return**

1090

1091 **if** **len(**forecast\_mean\_GARCH**)** **==** 0 **or** **len(**forecast\_variance\_GARCH**)** **==** 0**:**

1092 **print(**"Forecast mean or variance is empty."**)**

1093 **return**

1094

1095 min\_length **=** **min(len(**forecast\_mean\_ARIMA**),**

1096 **len(**forecast\_mean\_GARCH**),** **len(**test**))**

1097 forecast\_mean\_ARIMA **=** forecast\_mean\_ARIMA**[:**min\_length**]**

1098 forecast\_mean\_GARCH **=** forecast\_mean\_GARCH**[:**min\_length**]**

1099 forecast\_conf\_int\_ARIMA **=** forecast\_conf\_int\_ARIMA**[:**min\_length**]**

1100 test **=** test**[:**min\_length**]**

1101

1102 combined\_forecast\_mean **=** forecast\_mean\_ARIMA **+** forecast\_mean\_GARCH

1103 combined\_forecast\_lower\_CI **=** forecast\_conf\_int\_ARIMA**.***iloc***[:,** 0**]** **+** **(**

1104 forecast\_mean\_GARCH **-** 1.96 **\*** np**.***sqrt***(**forecast\_variance\_GARCH**))**

1105 combined\_forecast\_upper\_CI **=** forecast\_conf\_int\_ARIMA**.***iloc***[:,** 1**]** **+** **(**

1106 forecast\_mean\_GARCH **+** 1.96 **\*** np**.***sqrt***(**forecast\_variance\_GARCH**))**

1107

1108 combined\_forecast\_mean**.***index* **=** test**.***index*

1109 combined\_forecast\_lower\_CI**.***index* **=** test**.***index*

1110 combined\_forecast\_upper\_CI**.***index* **=** test**.***index*

1111

1112 ARIMA\_GARCH\_Model\_Prediction **=** pd**.***DataFrame***({**

1113 'Date'**:** test**.***index***,**

1114 'Forecast'**:** combined\_forecast\_mean**,**

1115 'Lower\_CI'**:** combined\_forecast\_lower\_CI**,**

1116 'Upper\_CI'**:** combined\_forecast\_upper\_CI

1117 **})**

1118

1119 ARIMA\_GARCH\_Model\_Prediction**.***to\_csv***(**

1120 "C:/Stock Price lists/" **+** stock\_name **+** "\_ARIMA\_GARCH\_Predicted.csv"**,** index**=False)**

1121

1122 mse **=** mean\_squared\_error**(**test**,** combined\_forecast\_mean**)**

1123 rmse **=** np**.***sqrt***(**mse**)**

1124 mae **=** mean\_absolute\_error**(**test**,** combined\_forecast\_mean**)**

1125 **print(**f"Mean Squared Error: {mse}"**)**

1126 **print(**f"Root Mean Squared Error: {rmse}"**)**

1127 **print(**f"Mean Absolute Error: {mae}"**)**

1128

1129 plt**.***figure***(**figsize**=(**12**,** 6**))**

1130 plt**.***plot***(**train**,** label**=**'Train'**)**

1131 plt**.***plot***(**test**,** label**=**'Test'**)**

1132 plt**.***plot***(**combined\_forecast\_mean**.***index***,** combined\_forecast\_mean**,**

1133 label**=**'Forecast'**,** color**=**'red'**)**

1134 plt**.***fill\_between***(**combined\_forecast\_mean**.***index***,** combined\_forecast\_lower\_CI**,**

1135 combined\_forecast\_upper\_CI**,** color**=**'gray'**,** alpha**=**0.3**)**

1136 plt**.***title***(**stock\_name **+** ' Stock Prices - Train, Test and Forecast (ARIMA' **+**

1137 **str(**self**.***arima\_order***)** **+** '-GARCH' **+** **str(**self**.***garch\_order***)** **+** ')'**)**

1138 plt**.***xlabel***(**'Date'**)**

1139 plt**.***ylabel***(**'Adjusted Close Price'**)**

1140 plt**.***legend***()**

1141 plt**.***show***()**

1142

1143 **return** ARIMA\_GARCH\_Model\_Prediction

1144

1145 **def** forecast\_next\_days**(**self**,** prices**,** n\_days**):**

1146 **if** self**.***arima\_model\_fit* **is** **None** **or** self**.***garch\_model\_fit* **is** **None:**

1147 **raise** **ValueError(**"Model must be fitted before forecasting."**)**

1148

1149 last\_date **=** prices**.***index***[-**1**]**

1150 forecast\_dates **=** pd**.***bdate\_range***(**

1151 start**=**last\_date **+** pd**.***Timedelta***(**days**=**1**),** periods**=**n\_days**)**

1152

1153 forecast\_ARIMA **=** self**.***arima\_model\_fit***.***get\_forecast***(**steps**=**n\_days**)**

1154 forecast\_mean\_ARIMA **=** forecast\_ARIMA**.***predicted\_mean*

1155 forecast\_conf\_int\_ARIMA **=** forecast\_ARIMA**.***conf\_int***()**

1156

1157 forecast\_GARCH **=** self**.***garch\_model\_fit***.***forecast***(**horizon**=**n\_days**)**

1158 forecast\_mean\_GARCH **=** forecast\_GARCH**.***mean***.***iloc***[-**n\_days**:].***values***.***flatten***(**

1159 **)**

1160 forecast\_variance\_GARCH **=** forecast\_GARCH**.***variance***.***iloc***[-**n\_days**:].***values***.***flatten***(**

1161 **)**

1162

1163 combined\_forecast\_mean **=** forecast\_mean\_ARIMA **+** forecast\_mean\_GARCH

1164 combined\_forecast\_lower\_CI **=** forecast\_conf\_int\_ARIMA**.***iloc***[:,** 0**]** **+** **(**

1165 forecast\_mean\_GARCH **-** 1.96 **\*** np**.***sqrt***(**forecast\_variance\_GARCH**))**

1166 combined\_forecast\_upper\_CI **=** forecast\_conf\_int\_ARIMA**.***iloc***[:,** 1**]** **+** **(**

1167 forecast\_mean\_GARCH **+** 1.96 **\*** np**.***sqrt***(**forecast\_variance\_GARCH**))**

1168

1169 combined\_forecast\_mean**.***index* **=** forecast\_dates

1170 combined\_forecast\_lower\_CI**.***index* **=** forecast\_dates

1171 combined\_forecast\_upper\_CI**.***index* **=** forecast\_dates

1172

1173 forecast\_df **=** pd**.***DataFrame***({**

1174 'Forecast'**:** combined\_forecast\_mean**,**

1175 'Lower\_CI'**:** combined\_forecast\_lower\_CI**,**

1176 'Upper\_CI'**:** combined\_forecast\_upper\_CI

1177 **})**

1178

1179 forecast\_df**.***to\_csv***(**

1180 "C:/Stock Price lists/" **+** stock\_name **+** "\_ARIMA\_GARCH\_Forecast.csv"**)**

1181

1182 **return** forecast\_df

1183

1184 **def** evaluate\_forecast**(**self**,** actual**,** forecast**,** prices**):**

1185 forecast\_aligned**,** actual\_aligned **=** forecast**.***align***(**

1186 actual**,** join**=**'inner'**,** axis**=**0**)**

1187

1188 forecast\_rmse **=** np**.***sqrt***(**mean\_squared\_error**(**

1189 actual\_aligned**,** forecast\_aligned**[**'Forecast'**]))**

1190 forecast\_mae **=** mean\_absolute\_error**(**

1191 actual\_aligned**,** forecast\_aligned**[**'Forecast'**])**

1192

1193 **print(**f"Forecast RMSE: {forecast\_rmse}"**)**

1194 **print(**f"Forecast MAE: {forecast\_mae}"**)**

1195

1196 plt**.***figure***(**figsize**=(**12**,** 6**))**

1197 plt**.***plot***(**prices**.***index***,** prices**,** label**=**'Original Series'**)**

1198 plt**.***plot***(**forecast**.***index***,**

1199 forecast**[**'Forecast'**],** label**=**'Forecast'**,** color**=**'red'**)**

1200 plt**.***plot***(**actual**.***index***,** actual**,** label**=**'Actual'**,** color**=**'orange'**)**

1201 plt**.***fill\_between***(**

1202 forecast**.***index***,** forecast**[**'Lower\_CI'**],** forecast**[**'Upper\_CI'**],** color**=**'gray'**,** alpha**=**0.3**)**

1203 plt**.***title***(**stock\_name **+** ' Stock Prices - Actual vs Forecast (ARIMA-GARCH)'**)**

1204 plt**.***xlabel***(**'Date'**)**

1205 plt**.***ylabel***(**'Price'**)**

1206 plt**.***legend***()**

1207 plt**.***show***()**

1208

1209

1210# Tuned RNN time series forecaster

1211class RNNTimeSeriesForecaster**:**

1212 **def** \_\_init\_\_**(**self**,** look\_back**=**1**,** units**=**50**,** learning\_rate**=**0.001**,** epochs**=**50**,** batch\_size**=**1**,** dropout**=**0**,** return\_sequences**=False):**

1213 self**.***look\_back* **=** look\_back

1214 self**.***units* **=** units

1215 self**.***learning\_rate* **=** learning\_rate

1216 self**.***epochs* **=** epochs

1217 self**.***batch\_size* **=** batch\_size

1218 self**.***dropout* **=** dropout

1219 self**.***return\_sequences* **=** return\_sequences

1220 self**.***model* **=** **None**

1221 self**.***scaler* **=** MinMaxScaler**(**feature\_range**=(**0**,** 1**))**

1222

1223 **def** create\_dataset**(**self**,** data**):**

1224 X**,** Y **=** **[],** **[]**

1225 **for** i **in** **range(len(**data**)** **-** self**.***look\_back* **-** 1**):**

1226 X**.***append***(**data**[**i**:(**i **+** self**.***look\_back***),** 0**])**

1227 Y**.***append***(**data**[**i **+** self**.***look\_back***,** 0**])**

1228 **return** np**.***array***(**X**),** np**.***array***(**Y**)**

1229

1230 **def** build\_model**(**self**):**

1231 model **=** Sequential**()**

1232 model**.***add***(**SimpleRNN**(**self**.***units***,** input\_shape**=(**

1233 self**.***look\_back***,** 1**),** return\_sequences**=**self**.***return\_sequences***))**

1234 **if** self**.***dropout* **>** 0**:**

1235 model**.***add***(**Dropout**(**self**.***dropout***))**

1236 **if** self**.***return\_sequences***:**

1237 model**.***add***(**SimpleRNN**(**self**.***units***))**

1238 model**.***add***(**Dense**(**1**))**

1239 model**.compile(**optimizer**=**tf**.***keras***.***optimizers***.***Adam***(**

1240 learning\_rate**=**self**.***learning\_rate***),** loss**=**'mean\_squared\_error'**)**

1241 self**.***model* **=** model

1242

1243 **def** fit**(**self**,** prices**):**

1244 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

1245 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

1246

1247 prices\_scaled **=** self**.***scaler***.***fit\_transform***(**prices**.***values***.***reshape***(-**1**,** 1**))**

1248

1249 train\_size **=** **int(len(**prices\_scaled**)** **\*** 0.8**)**

1250 train**,** test **=** prices\_scaled**[:**train\_size**],** prices\_scaled**[**train\_size**:]**

1251

1252 trainX**,** trainY **=** self**.***create\_dataset***(**train**)**

1253 testX**,** testY **=** self**.***create\_dataset***(**test**)**

1254

1255 trainX **=** np**.***reshape***(**trainX**,** **(**trainX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1256 testX **=** np**.***reshape***(**testX**,** **(**testX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1257

1258 self**.***build\_model***()**

1259 self**.***model***.***fit***(**trainX**,** trainY**,** epochs**=**self**.***epochs***,**

1260 batch\_size**=**self**.***batch\_size***,** verbose**=**2**,** validation\_data**=(**testX**,** testY**))**

1261

1262 trainPredict **=** self**.***model***.***predict***(**trainX**)**

1263 testPredict **=** self**.***model***.***predict***(**testX**)**

1264

1265 trainPredict **=** self**.***scaler***.***inverse\_transform***(**trainPredict**)**

1266 trainY **=** self**.***scaler***.***inverse\_transform***([**trainY**])**

1267 testPredict **=** self**.***scaler***.***inverse\_transform***(**testPredict**)**

1268 testY **=** self**.***scaler***.***inverse\_transform***([**testY**])**

1269

1270 train\_score **=** mean\_squared\_error**(**trainY**[**0**],** trainPredict**[:,** 0**])**

1271 test\_score **=** mean\_squared\_error**(**testY**[**0**],** testPredict**[:,** 0**])**

1272

1273 train\_rmse **=** np**.***sqrt***(**train\_score**)**

1274 test\_rmse **=** np**.***sqrt***(**test\_score**)**

1275

1276 train\_mae **=** mean\_absolute\_error**(**trainY**[**0**],** trainPredict**[:,** 0**])**

1277 test\_mae **=** mean\_absolute\_error**(**testY**[**0**],** testPredict**[:,** 0**])**

1278

1279 **print(**f"Train Mean Squared Error: {train\_score}"**)**

1280 **print(**f"Test Mean Squared Error: {test\_score}"**)**

1281 **print(**f"Train Root Mean Squared Error: {train\_rmse}"**)**

1282 **print(**f"Test Root Mean Squared Error: {test\_rmse}"**)**

1283 **print(**f"Train Mean Absolute Error: {train\_mae}"**)**

1284 **print(**f"Test Mean Absolute Error: {test\_mae}"**)**

1285

1286 trainPredictPlot **=** np**.***empty\_like***(**prices\_scaled**)**

1287 trainPredictPlot**[:,** **:]** **=** np**.***nan*

1288 trainPredictPlot**[**self**.***look\_back***:len(**

1289 trainPredict**)** **+** self**.***look\_back***,** **:]** **=** trainPredict

1290

1291 testPredictPlot **=** np**.***empty\_like***(**prices\_scaled**)**

1292 testPredictPlot**[:,** **:]** **=** np**.***nan*

1293 testPredictPlot**[len(**trainPredict**)** **+** **(**self**.***look\_back* **\*** 2**)** **+**

1294 1**:len(**prices\_scaled**)** **-** 1**,** **:]** **=** testPredict

1295

1296 plt**.***figure***(**figsize**=(**12**,** 6**))**

1297 plt**.***plot***(**self**.***scaler***.***inverse\_transform***(**

1298 prices\_scaled**),** label**=**'Original Series'**)**

1299 plt**.***plot***(**trainPredictPlot**,** label**=**'Train Predict'**,** color**=**'green'**)**

1300 plt**.***plot***(**testPredictPlot**,** label**=**'Test Predict'**,** color**=**'red'**)**

1301 plt**.***title***(**

1302 stock\_name **+** ' Stock Prices - Original Series and RNN Predictions'**)**

1303 plt**.***xlabel***(**'Date'**)**

1304 plt**.***ylabel***(**'Price'**)**

1305 plt**.***legend***()**

1306 plt**.***show***()**

1307

1308 **def** tune\_hyperparameters**(**self**,** prices**,** param\_dist**,** n\_iter\_search**=**20**):**

1309 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

1310 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

1311

1312 prices\_scaled **=** self**.***scaler***.***fit\_transform***(**prices**.***values***.***reshape***(-**1**,** 1**))**

1313

1314 train\_size **=** **int(len(**prices\_scaled**)** **\*** 0.8**)**

1315 train**,** test **=** prices\_scaled**[:**train\_size**],** prices\_scaled**[**train\_size**:]**

1316

1317 best\_params **=** **None**

1318 lowest\_val\_loss **=** **float(**'inf'**)**

1319

1320 param\_list **=** **list(**ParameterSampler**(**param\_dist**,** n\_iter**=**n\_iter\_search**))**

1321

1322 **for** params **in** param\_list**:**

1323 self**.***look\_back* **=** params**[**'look\_back'**]**

1324 self**.***units* **=** params**[**'units'**]**

1325 self**.***learning\_rate* **=** params**[**'learning\_rate'**]**

1326 self**.***epochs* **=** params**[**'epochs'**]**

1327 self**.***batch\_size* **=** params**[**'batch\_size'**]**

1328 self**.***dropout* **=** params**[**'dropout'**]**

1329 self**.***return\_sequences* **=** params**[**'return\_sequences'**]**

1330

1331 trainX**,** trainY **=** self**.***create\_dataset***(**train**)**

1332 validationX**,** validationY **=** self**.***create\_dataset***(**test**)**

1333

1334 trainX **=** np**.***reshape***(**trainX**,** **(**trainX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1335 validationX **=** np**.***reshape***(**

1336 validationX**,** **(**validationX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1337

1338 self**.***build\_model***()**

1339 self**.***model***.***fit***(**trainX**,** trainY**,** epochs**=**self**.***epochs***,** batch\_size**=**self**.***batch\_size***,**

1340 verbose**=**0**,** validation\_data**=(**validationX**,** validationY**))**

1341

1342 val\_predict **=** self**.***model***.***predict***(**validationX**)**

1343 val\_predict **=** self**.***scaler***.***inverse\_transform***(**val\_predict**)**

1344 validationY\_inverse **=** self**.***scaler***.***inverse\_transform***([**validationY**])**

1345 val\_loss **=** mean\_squared\_error**(**

1346 validationY\_inverse**[**0**],** val\_predict**[:,** 0**])**

1347

1348 **if** val\_loss **<** lowest\_val\_loss**:**

1349 lowest\_val\_loss **=** val\_loss

1350 best\_params **=** params

1351

1352 **print(**f"Best Hyperparameters: {best\_params}"**)**

1353 **print(**f"Lowest Validation Loss: {lowest\_val\_loss}"**)**

1354

1355 self**.***look\_back* **=** best\_params**[**'look\_back'**]**

1356 self**.***units* **=** best\_params**[**'units'**]**

1357 self**.***learning\_rate* **=** best\_params**[**'learning\_rate'**]**

1358 self**.***epochs* **=** best\_params**[**'epochs'**]**

1359 self**.***batch\_size* **=** best\_params**[**'batch\_size'**]**

1360 self**.***dropout* **=** best\_params**[**'dropout'**]**

1361 self**.***return\_sequences* **=** best\_params**[**'return\_sequences'**]**

1362

1363

1364def fit\_rnn\_forecaster\_class**(**prices**):**

1365 # Initialize the forecaster

1366 forecaster **=** RNNTimeSeriesForecaster**()**

1367 # Fit the model

1368 forecaster**.***fit***(**prices**)**

1369

1370 # Tune hyperparameters

1371 param\_dist **=** **{**

1372 'units'**:** randint**(**20**,** 100**),**

1373 'look\_back'**:** randint**(**1**,** 10**),**

1374 'learning\_rate'**:** uniform**(**0.001**,** 0.01**),**

1375 'epochs'**:** randint**(**10**,** 100**),**

1376 'batch\_size'**:** randint**(**1**,** 32**),**

1377 'dropout'**:** uniform**(**0**,** 0.5**),**

1378 'return\_sequences'**:** **[True,** **False]**

1379 **}**

1380 forecaster**.***tune\_hyperparameters***(**prices**,** param\_dist**,** n\_iter\_search**=**50**)**

1381 # Fit the model with the best hyperparameters found

1382 forecaster**.***fit***(**prices**)**

1383

1384

1385class LSTMTimeSeriesForecaster**:**

1386 **def** \_\_init\_\_**(**self**,** look\_back**=**1**,** units**=**50**,** learning\_rate**=**0.001**,** epochs**=**50**,** batch\_size**=**1**,** dropout**=**0**,** return\_sequences**=False):**

1387 self**.***look\_back* **=** look\_back

1388 self**.***units* **=** units

1389 self**.***learning\_rate* **=** learning\_rate

1390 self**.***epochs* **=** epochs

1391 self**.***batch\_size* **=** batch\_size

1392 self**.***dropout* **=** dropout

1393 self**.***return\_sequences* **=** return\_sequences

1394 self**.***model* **=** **None**

1395 self**.***scaler* **=** MinMaxScaler**(**feature\_range**=(**0**,** 1**))**

1396

1397 **def** create\_dataset**(**self**,** data**):**

1398 X**,** Y **=** **[],** **[]**

1399 **for** i **in** **range(len(**data**)** **-** self**.***look\_back* **-** 1**):**

1400 X**.***append***(**data**[**i**:(**i **+** self**.***look\_back***),** 0**])**

1401 Y**.***append***(**data**[**i **+** self**.***look\_back***,** 0**])**

1402 **return** np**.***array***(**X**),** np**.***array***(**Y**)**

1403

1404 **def** build\_model**(**self**):**

1405 model **=** Sequential**()**

1406 model**.***add***(**LSTM**(**self**.***units***,** input\_shape**=(**self**.***look\_back***,** 1**),**

1407 return\_sequences**=**self**.***return\_sequences***))**

1408 **if** self**.***dropout* **>** 0**:**

1409 model**.***add***(**Dropout**(**self**.***dropout***))**

1410 **if** self**.***return\_sequences***:**

1411 model**.***add***(**LSTM**(**self**.***units***))**

1412 model**.***add***(**Dense**(**1**))**

1413 model**.compile(**optimizer**=**tf**.***keras***.***optimizers***.***Adam***(**

1414 learning\_rate**=**self**.***learning\_rate***),** loss**=**'mean\_squared\_error'**)**

1415 self**.***model* **=** model

1416

1417 **def** fit**(**self**,** prices**):**

1418 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

1419 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

1420

1421 prices\_scaled **=** self**.***scaler***.***fit\_transform***(**prices**.***values***.***reshape***(-**1**,** 1**))**

1422

1423 train\_size **=** **int(len(**prices\_scaled**)** **\*** 0.8**)**

1424 train**,** test **=** prices\_scaled**[:**train\_size**],** prices\_scaled**[**train\_size**:]**

1425

1426 trainX**,** trainY **=** self**.***create\_dataset***(**train**)**

1427 testX**,** testY **=** self**.***create\_dataset***(**test**)**

1428

1429 trainX **=** np**.***reshape***(**trainX**,** **(**trainX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1430 testX **=** np**.***reshape***(**testX**,** **(**testX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1431

1432 self**.***build\_model***()**

1433 self**.***model***.***fit***(**trainX**,** trainY**,** epochs**=**self**.***epochs***,**

1434 batch\_size**=**self**.***batch\_size***,** verbose**=**2**,** validation\_data**=(**testX**,** testY**))**

1435

1436 trainPredict **=** self**.***model***.***predict***(**trainX**)**

1437 testPredict **=** self**.***model***.***predict***(**testX**)**

1438

1439 trainPredict **=** self**.***scaler***.***inverse\_transform***(**trainPredict**)**

1440 trainY **=** self**.***scaler***.***inverse\_transform***([**trainY**])**

1441 testPredict **=** self**.***scaler***.***inverse\_transform***(**testPredict**)**

1442 testY **=** self**.***scaler***.***inverse\_transform***([**testY**])**

1443

1444 train\_score **=** mean\_squared\_error**(**trainY**[**0**],** trainPredict**[:,** 0**])**

1445 test\_score **=** mean\_squared\_error**(**testY**[**0**],** testPredict**[:,** 0**])**

1446

1447 train\_rmse **=** np**.***sqrt***(**train\_score**)**

1448 test\_rmse **=** np**.***sqrt***(**test\_score**)**

1449

1450 train\_mae **=** mean\_absolute\_error**(**trainY**[**0**],** trainPredict**[:,** 0**])**

1451 test\_mae **=** mean\_absolute\_error**(**testY**[**0**],** testPredict**[:,** 0**])**

1452

1453 **print(**f"Train Mean Squared Error: {train\_score}"**)**

1454 **print(**f"Test Mean Squared Error: {test\_score}"**)**

1455 **print(**f"Train Root Mean Squared Error: {train\_rmse}"**)**

1456 **print(**f"Test Root Mean Squared Error: {test\_rmse}"**)**

1457 **print(**f"Train Mean Absolute Error: {train\_mae}"**)**

1458 **print(**f"Test Mean Absolute Error: {test\_mae}"**)**

1459

1460 trainPredictPlot **=** np**.***empty\_like***(**prices\_scaled**)**

1461 trainPredictPlot**[:,** **:]** **=** np**.***nan*

1462 trainPredictPlot**[**self**.***look\_back***:len(**

1463 trainPredict**)** **+** self**.***look\_back***,** **:]** **=** trainPredict

1464

1465 testPredictPlot **=** np**.***empty\_like***(**prices\_scaled**)**

1466 testPredictPlot**[:,** **:]** **=** np**.***nan*

1467 testPredictPlot**[len(**trainPredict**)** **+** **(**self**.***look\_back* **\*** 2**)** **+**

1468 1**:len(**prices\_scaled**)** **-** 1**,** **:]** **=** testPredict

1469

1470 plt**.***figure***(**figsize**=(**12**,** 6**))**

1471 plt**.***plot***(**self**.***scaler***.***inverse\_transform***(**

1472 prices\_scaled**),** label**=**'Original Series'**)**

1473 plt**.***plot***(**trainPredictPlot**,** label**=**'Train Predict'**,** color**=**'green'**)**

1474 plt**.***plot***(**testPredictPlot**,** label**=**'Test Predict'**,** color**=**'red'**)**

1475 plt**.***title***(**

1476 stock\_name **+** ' Stock Prices - Original Series and LSTM Predictions'**)**

1477 plt**.***xlabel***(**'Date'**)**

1478 plt**.***ylabel***(**'Price'**)**

1479 plt**.***legend***()**

1480 plt**.***show***()**

1481

1482 **def** tune\_hyperparameters**(**self**,** prices**,** param\_dist**,** n\_iter\_search**=**20**):**

1483 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

1484 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

1485

1486 prices\_scaled **=** self**.***scaler***.***fit\_transform***(**prices**.***values***.***reshape***(-**1**,** 1**))**

1487

1488 train\_size **=** **int(len(**prices\_scaled**)** **\*** 0.8**)**

1489 train**,** test **=** prices\_scaled**[:**train\_size**],** prices\_scaled**[**train\_size**:]**

1490

1491 best\_params **=** **None**

1492 lowest\_val\_loss **=** **float(**'inf'**)**

1493

1494 param\_list **=** **list(**ParameterSampler**(**param\_dist**,** n\_iter**=**n\_iter\_search**))**

1495

1496 **for** params **in** param\_list**:**

1497 self**.***look\_back* **=** params**[**'look\_back'**]**

1498 self**.***units* **=** params**[**'units'**]**

1499 self**.***learning\_rate* **=** params**[**'learning\_rate'**]**

1500 self**.***epochs* **=** params**[**'epochs'**]**

1501 self**.***batch\_size* **=** params**[**'batch\_size'**]**

1502 self**.***dropout* **=** params**[**'dropout'**]**

1503 self**.***return\_sequences* **=** params**[**'return\_sequences'**]**

1504

1505 trainX**,** trainY **=** self**.***create\_dataset***(**train**)**

1506 validationX**,** validationY **=** self**.***create\_dataset***(**test**)**

1507

1508 trainX **=** np**.***reshape***(**trainX**,** **(**trainX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1509 validationX **=** np**.***reshape***(**

1510 validationX**,** **(**validationX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1511

1512 self**.***build\_model***()**

1513 self**.***model***.***fit***(**trainX**,** trainY**,** epochs**=**self**.***epochs***,** batch\_size**=**self**.***batch\_size***,**

1514 verbose**=**0**,** validation\_data**=(**validationX**,** validationY**))**

1515

1516 val\_predict **=** self**.***model***.***predict***(**validationX**)**

1517 val\_predict **=** self**.***scaler***.***inverse\_transform***(**val\_predict**)**

1518 validationY\_inverse **=** self**.***scaler***.***inverse\_transform***([**validationY**])**

1519 val\_loss **=** mean\_squared\_error**(**

1520 validationY\_inverse**[**0**],** val\_predict**[:,** 0**])**

1521

1522 **if** val\_loss **<** lowest\_val\_loss**:**

1523 lowest\_val\_loss **=** val\_loss

1524 best\_params **=** params

1525

1526 **print(**f"Best Hyperparameters: {best\_params}"**)**

1527 **print(**f"Lowest Validation Loss: {lowest\_val\_loss}"**)**

1528

1529 self**.***look\_back* **=** best\_params**[**'look\_back'**]**

1530 self**.***units* **=** best\_params**[**'units'**]**

1531 self**.***learning\_rate* **=** best\_params**[**'learning\_rate'**]**

1532 self**.***epochs* **=** best\_params**[**'epochs'**]**

1533 self**.***batch\_size* **=** best\_params**[**'batch\_size'**]**

1534 self**.***dropout* **=** best\_params**[**'dropout'**]**

1535 self**.***return\_sequences* **=** best\_params**[**'return\_sequences'**]**

1536

1537

1538def fit\_lstm\_forecaster\_class**(**prices**):**

1539

1540 # Initialize the forecaster

1541 forecaster **=** LSTMTimeSeriesForecaster**()**

1542

1543 # Fit the model

1544 forecaster**.***fit***(**prices**)**

1545

1546 # Tune hyperparameters

1547 param\_dist **=** **{**

1548 'units'**:** randint**(**20**,** 100**),**

1549 'look\_back'**:** randint**(**1**,** 10**),**

1550 'learning\_rate'**:** uniform**(**0.001**,** 0.01**),**

1551 'epochs'**:** randint**(**10**,** 100**),**

1552 'batch\_size'**:** randint**(**1**,** 32**),**

1553 'dropout'**:** uniform**(**0**,** 0.5**),**

1554 'return\_sequences'**:** **[True,** **False]**

1555 **}**

1556 forecaster**.***tune\_hyperparameters***(**prices**,** param\_dist**,** n\_iter\_search**=**50**)**

1557

1558 # Fit the model with the best hyperparameters found

1559 forecaster**.***fit***(**prices**)**

1560

1561

1562class GRUTimeSeriesForecaster**:**

1563 **def** \_\_init\_\_**(**self**,** look\_back**=**1**,** units**=**50**,** learning\_rate**=**0.001**,** epochs**=**50**,** batch\_size**=**1**,** dropout**=**0**,** return\_sequences**=False):**

1564 self**.***look\_back* **=** look\_back

1565 self**.***units* **=** units

1566 self**.***learning\_rate* **=** learning\_rate

1567 self**.***epochs* **=** epochs

1568 self**.***batch\_size* **=** batch\_size

1569 self**.***dropout* **=** dropout

1570 self**.***return\_sequences* **=** return\_sequences

1571 self**.***model* **=** **None**

1572 self**.***scaler* **=** MinMaxScaler**(**feature\_range**=(**0**,** 1**))**

1573

1574 **def** create\_dataset**(**self**,** data**):**

1575 X**,** Y **=** **[],** **[]**

1576 **for** i **in** **range(len(**data**)** **-** self**.***look\_back* **-** 1**):**

1577 X**.***append***(**data**[**i**:(**i **+** self**.***look\_back***),** 0**])**

1578 Y**.***append***(**data**[**i **+** self**.***look\_back***,** 0**])**

1579 **return** np**.***array***(**X**),** np**.***array***(**Y**)**

1580

1581 **def** build\_model**(**self**):**

1582 model **=** Sequential**()**

1583 model**.***add***(**GRU**(**self**.***units***,** input\_shape**=(**self**.***look\_back***,** 1**),**

1584 return\_sequences**=**self**.***return\_sequences***))**

1585 **if** self**.***dropout* **>** 0**:**

1586 model**.***add***(**Dropout**(**self**.***dropout***))**

1587 **if** self**.***return\_sequences***:**

1588 model**.***add***(**GRU**(**self**.***units***))**

1589 model**.***add***(**Dense**(**1**))**

1590 model**.compile(**optimizer**=**tf**.***keras***.***optimizers***.***Adam***(**

1591 learning\_rate**=**self**.***learning\_rate***),** loss**=**'mean\_squared\_error'**)**

1592 self**.***model* **=** model

1593

1594 **def** fit**(**self**,** prices**):**

1595 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

1596 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

1597

1598 prices\_scaled **=** self**.***scaler***.***fit\_transform***(**prices**.***values***.***reshape***(-**1**,** 1**))**

1599

1600 train\_size **=** **int(len(**prices\_scaled**)** **\*** 0.8**)**

1601 train**,** test **=** prices\_scaled**[:**train\_size**],** prices\_scaled**[**train\_size**:]**

1602

1603 trainX**,** trainY **=** self**.***create\_dataset***(**train**)**

1604 testX**,** testY **=** self**.***create\_dataset***(**test**)**

1605

1606 trainX **=** np**.***reshape***(**trainX**,** **(**trainX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1607 testX **=** np**.***reshape***(**testX**,** **(**testX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1608

1609 self**.***build\_model***()**

1610 self**.***model***.***fit***(**trainX**,** trainY**,** epochs**=**self**.***epochs***,**

1611 batch\_size**=**self**.***batch\_size***,** verbose**=**2**,** validation\_data**=(**testX**,** testY**))**

1612

1613 trainPredict **=** self**.***model***.***predict***(**trainX**)**

1614 testPredict **=** self**.***model***.***predict***(**testX**)**

1615

1616 trainPredict **=** self**.***scaler***.***inverse\_transform***(**trainPredict**)**

1617 trainY **=** self**.***scaler***.***inverse\_transform***([**trainY**])**

1618 testPredict **=** self**.***scaler***.***inverse\_transform***(**testPredict**)**

1619 testY **=** self**.***scaler***.***inverse\_transform***([**testY**])**

1620

1621 train\_score **=** mean\_squared\_error**(**trainY**[**0**],** trainPredict**[:,** 0**])**

1622 test\_score **=** mean\_squared\_error**(**testY**[**0**],** testPredict**[:,** 0**])**

1623

1624 train\_rmse **=** np**.***sqrt***(**train\_score**)**

1625 test\_rmse **=** np**.***sqrt***(**test\_score**)**

1626

1627 train\_mae **=** mean\_absolute\_error**(**trainY**[**0**],** trainPredict**[:,** 0**])**

1628 test\_mae **=** mean\_absolute\_error**(**testY**[**0**],** testPredict**[:,** 0**])**

1629

1630 **print(**f"Train Mean Squared Error: {train\_score}"**)**

1631 **print(**f"Test Mean Squared Error: {test\_score}"**)**

1632 **print(**f"Train Root Mean Squared Error: {train\_rmse}"**)**

1633 **print(**f"Test Root Mean Squared Error: {test\_rmse}"**)**

1634 **print(**f"Train Mean Absolute Error: {train\_mae}"**)**

1635 **print(**f"Test Mean Absolute Error: {test\_mae}"**)**

1636

1637 trainPredictPlot **=** np**.***empty\_like***(**prices\_scaled**)**

1638 trainPredictPlot**[:,** **:]** **=** np**.***nan*

1639 trainPredictPlot**[**self**.***look\_back***:len(**

1640 trainPredict**)** **+** self**.***look\_back***,** **:]** **=** trainPredict

1641

1642 testPredictPlot **=** np**.***empty\_like***(**prices\_scaled**)**

1643 testPredictPlot**[:,** **:]** **=** np**.***nan*

1644 testPredictPlot**[len(**trainPredict**)** **+** **(**self**.***look\_back* **\*** 2**)** **+**

1645 1**:len(**prices\_scaled**)** **-** 1**,** **:]** **=** testPredict

1646

1647 plt**.***figure***(**figsize**=(**12**,** 6**))**

1648 plt**.***plot***(**self**.***scaler***.***inverse\_transform***(**

1649 prices\_scaled**),** label**=**'Original Series'**)**

1650 plt**.***plot***(**trainPredictPlot**,** label**=**'Train Predict'**,** color**=**'green'**)**

1651 plt**.***plot***(**testPredictPlot**,** label**=**'Test Predict'**,** color**=**'red'**)**

1652 plt**.***title***(**

1653 stock\_name **+** ' Stock Prices - Original Series and GRU Predictions'**)**

1654 plt**.***xlabel***(**'Date'**)**

1655 plt**.***ylabel***(**'Price'**)**

1656 plt**.***legend***()**

1657 plt**.***show***()**

1658

1659 **def** tune\_hyperparameters**(**self**,** prices**,** param\_dist**,** n\_iter\_search**=**20**):**

1660 **if** **not** **isinstance(**prices**.***index***,** pd**.***DatetimeIndex***):**

1661 **raise** **ValueError(**"The prices series must be indexed by dates"**)**

1662

1663 prices\_scaled **=** self**.***scaler***.***fit\_transform***(**prices**.***values***.***reshape***(-**1**,** 1**))**

1664

1665 train\_size **=** **int(len(**prices\_scaled**)** **\*** 0.8**)**

1666 train**,** test **=** prices\_scaled**[:**train\_size**],** prices\_scaled**[**train\_size**:]**

1667

1668 best\_params **=** **None**

1669 lowest\_val\_loss **=** **float(**'inf'**)**

1670

1671 param\_list **=** **list(**ParameterSampler**(**param\_dist**,** n\_iter**=**n\_iter\_search**))**

1672

1673 **for** params **in** param\_list**:**

1674 self**.***look\_back* **=** params**[**'look\_back'**]**

1675 self**.***units* **=** params**[**'units'**]**

1676 self**.***learning\_rate* **=** params**[**'learning\_rate'**]**

1677 self**.***epochs* **=** params**[**'epochs'**]**

1678 self**.***batch\_size* **=** params**[**'batch\_size'**]**

1679 self**.***dropout* **=** params**[**'dropout'**]**

1680 self**.***return\_sequences* **=** params**[**'return\_sequences'**]**

1681

1682 trainX**,** trainY **=** self**.***create\_dataset***(**train**)**

1683 validationX**,** validationY **=** self**.***create\_dataset***(**test**)**

1684

1685 trainX **=** np**.***reshape***(**trainX**,** **(**trainX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1686 validationX **=** np**.***reshape***(**

1687 validationX**,** **(**validationX**.***shape***[**0**],** self**.***look\_back***,** 1**))**

1688

1689 self**.***build\_model***()**

1690 self**.***model***.***fit***(**trainX**,** trainY**,** epochs**=**self**.***epochs***,** batch\_size**=**self**.***batch\_size***,**

1691 verbose**=**0**,** validation\_data**=(**validationX**,** validationY**))**

1692

1693 val\_predict **=** self**.***model***.***predict***(**validationX**)**

1694 val\_predict **=** self**.***scaler***.***inverse\_transform***(**val\_predict**)**

1695 validationY\_inverse **=** self**.***scaler***.***inverse\_transform***([**validationY**])**

1696 val\_loss **=** mean\_squared\_error**(**

1697 validationY\_inverse**[**0**],** val\_predict**[:,** 0**])**

1698

1699 **if** val\_loss **<** lowest\_val\_loss**:**

1700 lowest\_val\_loss **=** val\_loss

1701 best\_params **=** params

1702

1703 **print(**f"Best Hyperparameters: {best\_params}"**)**

1704 **print(**f"Lowest Validation Loss: {lowest\_val\_loss}"**)**

1705

1706 self**.***look\_back* **=** best\_params**[**'look\_back'**]**

1707 self**.***units* **=** best\_params**[**'units'**]**

1708 self**.***learning\_rate* **=** best\_params**[**'learning\_rate'**]**

1709 self**.***epochs* **=** best\_params**[**'epochs'**]**

1710 self**.***batch\_size* **=** best\_params**[**'batch\_size'**]**

1711 self**.***dropout* **=** best\_params**[**'dropout'**]**

1712 self**.***return\_sequences* **=** best\_params**[**'return\_sequences'**]**

1713

1714 **def** set\_hyperparameters**(**self**,** params**):**

1715 self**.***look\_back* **=** params**[**'look\_back'**]**

1716 self**.***units* **=** params**[**'units'**]**

1717 self**.***learning\_rate* **=** params**[**'learning\_rate'**]**

1718 self**.***epochs* **=** params**[**'epochs'**]**

1719 self**.***batch\_size* **=** params**[**'batch\_size'**]**

1720 self**.***dropout* **=** params**[**'dropout'**]**

1721 self**.***return\_sequences* **=** params**[**'return\_sequences'**]**

1722

1723 **def** forecast\_next\_days**(**self**,** prices**,** n\_days**):**

1724 forecast **=** **[]**

1725 last\_data **=** self**.***scaler***.***transform***(**

1726 prices**.***values***[-**self**.***look\_back***:].***reshape***(-**1**,** 1**))**

1727

1728 **for** \_ **in** **range(**n\_days**):**

1729 input\_data **=** np**.***reshape***(**last\_data**,** **(**1**,** self**.***look\_back***,** 1**))**

1730 next\_value **=** self**.***model***.***predict***(**input\_data**)**

1731 forecast**.***append***(**next\_value**[**0**,** 0**])**

1732 last\_data **=** np**.***append***(**last\_data**[**1**:],** next\_value**)**

1733

1734 forecast **=** np**.***array***(**forecast**)**

1735 forecast **=** self**.***scaler***.***inverse\_transform***(**forecast**.***reshape***(-**1**,** 1**))**

1736

1737 # Generate date range for forecast

1738 last\_date **=** prices**.***index***[-**1**]**

1739 forecast\_dates **=** pd**.***bdate\_range***(**

1740 start**=**last\_date **+** pd**.***Timedelta***(**days**=**1**),** periods**=**n\_days**)**

1741 # print(forecast\_dates)

1742 # print(f"Shape of forecast\_dates: {forecast\_dates.shape}")

1743

1744 # Create DataFrame

1745 forecast\_df **=** pd**.***DataFrame***(**

1746 forecast**,** index**=**forecast\_dates**,** columns**=[**'Forecast'**])**

1747

1748 forecast\_df**.***set\_index***(**forecast\_dates**,** inplace**=True)**

1749 forecast\_df**.***index***.***name* **=** "date"

1750 # print(forecast\_df)

1751

1752 save\_dataframe\_to\_file**(**

1753 forecast\_df**,** "C:/Stock Price lists/" **+** stock\_name **+** "\_GRU\_forecast.csv"**,** overwrite**=True,** index**=True)**

1754

1755 **return** forecast\_df

1756

1757 **def** evaluate\_forecast**(**self**,** prices**,** forecast**,** actual**):**

1758 # Align actual and forecast data by their indices

1759 forecast\_aligned**,** actual\_aligned **=** forecast**.***align***(**

1760 actual**,** join**=**'inner'**,** axis**=**0**)**

1761

1762 # Drop NaN values if any

1763 forecast\_aligned **=** forecast\_aligned**.***dropna***()**

1764 actual\_aligned **=** actual\_aligned**.***dropna***()**

1765

1766 # Filter out zero values in actual to avoid division by zero in MAPE calculation

1767 mask **=** actual\_aligned **!=** 0

1768 actual\_filtered **=** actual\_aligned**[**mask**]**

1769 forecast\_filtered **=** forecast\_aligned**[**mask**]**

1770

1771 # Compute metrics

1772 forecast\_rmse **=** np**.***sqrt***(**mean\_squared\_error**(**

1773 actual\_filtered**,** forecast\_filtered**))**

1774 forecast\_mae **=** mean\_absolute\_error**(**actual\_filtered**,** forecast\_filtered**)**

1775

1776 **print(**f"Forecast RMSE: {forecast\_rmse}"**)**

1777 **print(**f"Forecast MAE: {forecast\_mae}"**)**

1778

1779 # Plot results

1780 plt**.***figure***(**figsize**=(**12**,** 6**))**

1781 plt**.***plot***(**prices**.***index***,** prices**,** label**=**'Original Series'**)**

1782 plt**.***plot***(**forecast**.***index***,** forecast**,** label**=**'Forecast'**,** color**=**'red'**)**

1783 plt**.***plot***(**actual**.***index***,** actual**,** label**=**'Actual'**,** color**=**'orange'**)**

1784 plt**.***title***(**f'{stock\_name} Stock Prices - GRU Model Forecast vs Actual'**)**

1785 plt**.***xlabel***(**'Date'**)**

1786 plt**.***ylabel***(**'Price'**)**

1787 plt**.***legend***()**

1788 plt**.***show***()**

1789

1790

1791def fit\_gru\_forecaster\_class**(**prices**,** actual\_next\_prices**=None,** best\_params**=None):**

1792

1793 # Initialize the forecaster

1794 forecaster **=** GRUTimeSeriesForecaster**()**

1795

1796 **if** best\_params**:**

1797 forecaster**.***set\_hyperparameters***(**best\_params**)**

1798 **else:**

1799 # Tune hyperparameters if not provided

1800 param\_dist **=** **{**

1801 'units'**:** randint**(**20**,** 100**),**

1802 'look\_back'**:** randint**(**1**,** 10**),**

1803 'learning\_rate'**:** uniform**(**0.001**,** 0.01**),**

1804 'epochs'**:** randint**(**10**,** 100**),**

1805 'batch\_size'**:** randint**(**1**,** 32**),**

1806 'dropout'**:** uniform**(**0**,** 0.5**),**

1807 'return\_sequences'**:** **[True,** **False]**

1808 **}**

1809 forecaster**.***tune\_hyperparameters***(**prices**,** param\_dist**,** n\_iter\_search**=**50**)**

1810

1811 # Fit the model with the best hyperparameters found or provided

1812 forecaster**.***fit***(**prices**)**

1813

1814 **if** actual\_next\_prices **is** **not** **None:**

1815 # Forecast the next 3 months (approximately 63 business days)

1816 n\_days **=** 63

1817 forecast\_df **=** forecaster**.***forecast\_next\_days***(**close\_prices**,** n\_days**)**

1818 # Evaluate the forecast

1819 forecaster**.***evaluate\_forecast***(**prices**,** forecast\_df**,** actual\_next\_prices**)**

1820

1821# ------------------------------------------------------------------

1822

1823

1824# Print descriptive analysis of passed variable

1825do\_descriptive\_analysis**(**close\_prices**)**

1826

1827# ------------------------------------------------------------------

1828

1829# Identify the best probability distribution

1830best\_fit\_pd **=** fit\_best\_distribution**(**

1831 close\_prices**,** show\_best\_fit**=True,** show\_plot**=True)**

1832

1833# ------------------------------------------------------------------

1834

1835# detect outliers

1836outliers\_df **=** detect\_outliers**(list(**best\_fit\_pd**.***keys***())[**

1837 0**],** close\_prices**,** **True,** **False,** **True)**

1838

1839# ------------------------------------------------------------------

1840

1841# Check for stationarity without differencing

1842perform\_adfuller**(**close\_prices**)**

1843Perform\_Ljung\_Box\_test**(**close\_prices**,** lags**=**252**)**

1844Calculate\_correlogram\_acf\_pacf**(**close\_prices**,** nlags**=**252**)**

1845

1846# ------------------------------------------------------------------

1847

1848# check for stationarity with differencing

1849difference\_close\_prices **=** perform\_differencing**(**close\_prices**)**

1850perform\_adfuller**(**difference\_close\_prices**)**

1851Perform\_Ljung\_Box\_test**(**difference\_close\_prices**,** lags**=**252**)**

1852Calculate\_correlogram\_acf\_pacf**(**difference\_close\_prices**,** nlags**=**252**)**

1853

1854# ------------------------------------------------------------------

1855

1856

1857# inspect original trends to identify trends

1858# fit simple moving average to smooth fluctuation and then plot it with original trends

1859draw\_Original\_Trend**(**close\_prices**,** ''**)**

1860fit\_sma**(**close\_prices**,** window\_size**=**90**,** show\_plot**=True,**

1861 print\_result**=False,** save\_to\_file**=False)**

1862

1863# ------------------------------------------------------------------

1864

1865# to evaluate all models with simple moving average and exponential moving average as a baseline

1866evaluate\_vs\_baseline\_sma**(**close\_prices**,** stock\_name**,**

1867 window\_size**=**90**,** test\_size**=**0.2**)**

1868evaluate\_vs\_baseline\_ema**(**close\_prices**,** stock\_name**,**

1869 window\_size**=**90**,** test\_size**=**0.2**)**

1870# ------------------------------------------------------------------

1871

1872# fit Exponential Smoothing time series model

1873fit\_exponential\_Smoothing\_forecaster\_class**(**

1874 close\_prices**,** stock\_name**,** forecasting\_Duration**=**90**,** test\_size**=**0.2**,** seasonal\_periods**=None,** confidence\_level**=**0.95**,** predict\_future\_value**=False)**

1875

1876# ------------------------------------------------------------------

1877

1878# fit ARIMA time series model

1879fit\_ARIMA**(**close\_prices**,** **(**2**,** 2**,** 2**),** test\_size**=**0.2**)**

1880

1881# ------------------------------------------------------------------

1882

1883

1884# Fit ARIMA-GARCH model

1885fit\_ARIMA\_GARCH**(**close\_prices**,** Arima\_order**=(**2**,** 2**,** 2**),**

1886 Garch\_order**=(**2**,** 2**),** test\_size**=**0.2**)**

1887

1888

1889arima\_garch\_forecaster **=** ARIMAGARCHForecaster**(**

1890 arima\_order**=(**2**,** 2**,** 2**),** garch\_order**=(**2**,** 2**))**

1891arima\_garch\_forecaster.fit**(**close\_prices**)**

1892forecast\_df **=** arima\_garch\_forecaster**.***forecast\_next\_days***(**

1893 close\_prices**,** n\_days**=**63**)**

1894arima\_garch\_forecaster.evaluate\_forecast**(**

1895 actual\_next\_prices**,** forecast\_df**,** close\_prices**)**

1896

1897# ------------------------------------------------------------------

1898

1899# fit adjusted RNN time series model

1900fit\_rnn\_forecaster\_class**(**close\_prices**)**

1901

1902# ------------------------------------------------------------------

1903

1904# fit adjusted LSTM time series model

1905fit\_lstm\_forecaster\_class**(**close\_prices**)**

1906

1907# ------------------------------------------------------------------

1908

1909# Call the function when we don't have tuned parameters and actual next prices

1910fit\_gru\_forecaster\_class**(**close\_prices**)**

1911

1912# Call the forecasting function without best hyperparameters to trigger tuning

1913fit\_gru\_forecaster\_class**(**close\_prices**,** actual\_next\_prices**)**

1914

1915# Define the best hyperparameters for Alphabet

1916# best\_params = {

1917# 'batch\_size': 24,

1918# 'dropout': 0.0010595028648908156,

1919# 'epochs': 91,

1920# 'learning\_rate': 0.0065272063812476955,

1921# 'look\_back': 4,

1922# 'return\_sequences': False,

1923# 'units': 34

1924# }

1925

1926# Define the best hyperparameters for Microsoft

1927# best\_params = {

1928# 'batch\_size': 16,

1929# 'dropout': 0.044295524080964044,

1930# 'epochs': 73,

1931# 'learning\_rate': 0.010301134772215333,

1932# 'look\_back': 5,

1933# 'return\_sequences': False,

1934# 'units': 40

1935# }

1936

1937

1938# Call the function when we don't have the actual next prices

1939# fit\_gru\_forecaster\_class(close\_prices, best\_params=best\_params)

1940

1941

1942# Call the function with the best hyperparameters

1943# fit\_gru\_forecaster\_class(close\_prices, actual\_next\_prices, best\_params)

1944

1. **Sample of Selected and Analysed Articles for Literature Review**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Year | AIM | Findings | Journal |
| Research on Stock Returns Forecast of the Four Major Banks Based on ARMA and GARCH Model | 2020 | Compare the effectiveness of ARMA and GARCH models in forecasting daily stock returns of four major banks | The GARCH model better captured historical volatility patterns of the banks' stock returns, but, the ARMA model offered slightly more accurate forecasts for future prices. This suggests past volatility might not always predict future stock movements. | J. Phys |
| A Prediction Approach for Stock Market Volatility Based on Time Series Data | 2019 | explore the use of ARIMA models, to predict stock market movements. | The ARIMA model provided a reasonably accurate forecast | IEEE |
| Stock Price Prediction Using the ARIMA Model | 2014 | the effectiveness of ARIMA models in predicting stock prices | The ARIMA model performed well, particularly for short-term predictions for both the NYSE and NSE. It could be a competitive technique in the field of stock price forecasting. | UKSim-AMSS 16th International Conference on Computer Modelling and Simulation |
| ARMA–GARCH model with fractional generalized hyperbolic innovations | 2022 | Develop a more sophisticated model to capture the complexities of high-frequency stock returns | Introduces a multivariate ARMA-GARCH model that incorporates fractional generalized hyperbolic (fGH) innovations. The fGH model more effectively captures the fat tails, volatility clustering, and long-range dependence properties compared to the standard model. | Financial Innovation |
| Stock price prediction using LSTM, RNN and CNN-sliding window model | 2017 | develop a model-independent approach using three deep learning architectures to predict short-term stock prices for companies listed on the NSE (National Stock Exchange of India) | Convolutional Neural Network (CNN) performed best among three deep learning architectures tested for short-term stock price prediction. | Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) |
| Forecasting multistep daily stock prices for long-term investment decisions: A study of deep learning models on global indices | 2024 | the ability of deep learning models to predict long-term (up to a year) daily prices of global stock indices | LSTM emerged as the best deep learning model for long-term (up to a year) stock price prediction of global indices, outperforming other models like CNN and RNN. | Engineering Applications of Artificial Intelligence |
| A LSTM-based method for stock returns prediction: A case study of China stock market | 2015 | the effectiveness of a Long Short-Term Memory (LSTM) network in predicting stock returns in the Chinese market | LSTM model improved the accuracy of stock return prediction in China from 14.3% (random guessing) to 27.2%, highlighting LSTM's potential for this task despite the market's inherent difficulty | IEEE International Conference on Big Data (Big Data) |
| ARIMA vs LSTM on NASDAQ stock exchange data | 2022 | compares the performance of ARIMA (statistical) and LSTM (deep learning) models for predicting stock prices on the NASDAQ exchange | The study found that ARIMA outperformed LSTM for predicting NASDAQ stock prices when limited to one feature (historical price) and predicting multiple future periods (months). ARIMA's advantage increased with longer prediction windows (30 days to 9 months). | Procedia Computer Science |
| A Comparison of ARIMA and LSTM in Forecasting Time Series | 2018 | whether deep learning algorithms like LSTM outperform traditional methods like ARIMA for forecasting time series data | The study shows that LSTM, a deep learning approach, significantly outperforms the traditional ARIMA model in forecasting time series data. | IEEE International Conference on Machine Learning and Applications (ICMLA) |

Table : Sample of selected and analysed articles for literature review