MACHINE LEARNING METHODS AND SOFTWARE TOOLS FOR PREDICTING CYCLICAL ECONOMIC PROCESSES IN A STOCK MARKET

by Saveliev Myron

A Capstone Project

Presented in Partial Fulfilment of the Requirements for the Degree

Master

American University Kyiv

2025

APPROVED BY:

Professor Serhii Lupenko, Ph.D., Dr. Habil

Acknowledgements

I would like to express my deep gratitude to Dr. Habil Serhii Lupenko, a visiting professor at the American University of Kyiv, for his guidance and support during the capstone project. Dr. Lupenko's expertise in mathematical modeling and forecasting was highly influential in directing the research. His constructive feedback and encouragement helped me refine ideas and overcome challenges.

ABSTRACT

This capstone project focuses on forecasting cyclical economic trends in the stock market, using Apple Inc. (AAPL) stock data as a case study. Key research questions include: (1) How effective are machine learning models in forecasting market cycles? (2) Which models provide the most accurate predictions? Historical stock price data from 2010 to 2023 was collected from Yahoo Finance, and seven forecasting models were implemented: ARIMA, SARIMA, Gradient Boosting, Random Forest, LSTM, GRU, and Prophet.

The research involved preprocessing the data, performing exploratory analysis, and evaluating the performance of each model using key metrics, including RMSE, MAPE, and R^2. Results demonstrated that ARIMA and SARIMA provided the most accurate forecasts, achieving an RMSE of 3.11, MAPE of 0.01, and R^2 of 0.98. Among deep learning models, LSTM outperformed GRU with an RMSE of 4.35 and R^2 of 0.96, effectively capturing non-linear dependencies. Gradient Boosting, Random Forest, and Prophet models underperformed, with higher RMSE and negative R^2 values, making them less suitable for this task.

The study concludes that ARIMA and SARIMA are optimal for efficient and accurate forecasting, while LSTM is well-suited for complex, non-linear patterns. Future work could explore the incorporation of macroeconomic indicators, hybrid models, or alternative data sources, such as sentiment analysis, to enhance predictive accuracy. This research offers practical tips for utilizing machine learning in financial forecasting.

Keywords: stock market forecasting, machine learning, ARIMA, LSTM, time series analysis, App

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Financial instruments including stocks and bonds enable market participants to create wealth and allocate capital across the globe through the complex stock market environment. Stock price and market trend prediction poses an essential challenge for traders and policymakers because financial data remains unpredictable and volatile throughout time.

The recent developments in data science and machine learning have established new methods to address this challenge. The financial prediction field has adopted machine learning algorithms including Random Forest and Gradient Boosting and deep learning models such as Long Short Term Memory (LSTM) networks. The modern statistical models ARIMA and SARIMA which dominated financial predictions before now share the stage with contemporary machine learning algorithms such as Random Forest and Gradient Boosting and deep learning techniques like Long Short Term Memory (LSTM) networks. These methods are particularly suitable for forecasting stock market trends as they can reveal concealed patterns and complex relationships, in data.

Effective stock market trend forecasting in the realm of informed decision making requires the ability to predict both bull and bear market patterns accurately. Accurate predictions can enhance security refine investment strategies and mitigate risks. The research field still lacks adequate studies comparing current machine learning and deep learning approaches with established statistical models. Research investigations fail to address the problem of merging these predictive models into interfaces for applications.

The research focuses on model prediction evaluation and interface development to make forecasting system predictions accessible while bridging the existing gaps. The research findings demonstrate machine learning's practical application in financial prediction and add to existing knowledge about financial forecasting methods.

This capstone project seeks to forecast stock market patterns through a comparison of statistical and machine learning models to discover the best prediction method for market cycles with a focus on creating an intuitive user interface.

The following research tasks are established in order to accomplish this goal:

1. Collect historical stock price data and economic indicators for analysis.

2. Preprocess the data to ensure consistency and quality.
3. Implement and compare the performance of various forecasting models, including:
    - Statistical models: ARIMA, SARIMA
    - Machine learning models: Random Forest, Gradient Boosting
    - Deep learning models: LSTM, GRU
    - Specialised framework: Prophet
4. Evaluate the models using key metrics, including RMSE, MAPE, and R^2.
5. Develop a user interface to enable intuitive data loading, model selection, visualisation of results, and report generation.
6. Analyse the results and conclude the models' performance and practical applications.

The capstone project is structured into several chapters for clarity and logical progression:

- Introduction. Outlines the background, relevance, goals, and research tasks.
- Literature Review and Theoretical Foundations. Reviews the existing methods and approaches in time series forecasting and stock market prediction.
- Methodology. Describes the data sources, preprocessing steps, model implementation, and evaluation metrics used in the research.
- Results and Analysis. Presents the performance of different forecasting models and compares their effectiveness.
- Implementation and Deployment. Details the design and functionality of the developed application.
- Conclusions. Summarise the key findings, discuss their implications, and offer recommendations for future research.
- Appendices. Includes supplementary materials such as sample reports and screenshots of the user interface.

CHAPTER 2. LITERATURE REVIEW AND THEORETICAL FOUNDATIONS

2.1. History of Time Series Forecasting

Time series forecasting functions as an adaptable analytical method which operates in various fields beyond financial markets. The first application of forecasting took place during the early 19th century for weather pattern and agricultural yield predictions (Hyndman & Athanasopoulos, 2018). The analytical instrument has developed into an essential tool which serves multiple sectors including supply chain management and healthcare analytics and energy load forecasting and sales prediction.

The energy sector benefits from forecasting models that predict electricity demand which enables grid operators to maintain reliability and reduce costs (Hyndman & Athanasopoulos, 2018). Time series forecasting enables better hospital resource management and patient admission prediction in similar ways. The various applications demonstrate that forecasting methods work effectively across different dataset types and problem domains.

Financial forecasting presents distinct challenges to other applications. The stock market presents unique challenges to accurate predictions because of its high volatility and sudden regime changes and noise which distinguish it from other fields. Financial data requires substantial modifications to forecasting methods which have proven successful in other domains. The ability to adapt forecasting techniques to specific task requirements and goals demonstrates their critical importance.

The fundamental analytical value of time series forecasting becomes evident through its various applications across different disciplines. The application of forecasting techniques in finance has received improvements through knowledge and advancements from meteorology as well as healthcare and energy fields (Hyndman & Athanasopoulos, 2018). The exchange of ideas demonstrates the necessity of combining knowledge from multiple disciplines to enhance both the precision and practicality of financial time series forecasting.

2.2. Forecasting Time Series in the Financial Market

Financial analysis depends on time series forecasting to predict market trends from historical data. The stock market uses these techniques to perform risk management and portfolio optimization as well as market trend identification (Hyndman & Athanasopoulos, 2018). The statistical models ARIMA

and SARIMA detect linear patterns together with seasonality within data through their analysis. The models work well for basic datasets yet fail to detect intricate and non-linear relationships between data points.

Time series forecasting has undergone substantial development since its first use in economics and engineering. The development of statistical models ARIMA and SARIMA became possible through earlier techniques such as exponential smoothing and simple moving averages which formed the basis for these models during the 20th century. The models generate easy-to-understand results for basic datasets and were created to analyze seasonal patterns in stationary data (Hyndman & Athanasopoulos, 2018).

The investigation of machine learning techniques during the late 20th and early 21st centuries emerged because statistical methods failed to model non-linear dependencies (Joseph, 2022). Random Forest and Gradient Boosting techniques provided a solution to handle complex data relationships in these models. The financial market's prevalent non-linear dynamics and long-term dependencies can be modeled by deep learning techniques including LSTM and GRU.

Without making strong assumptions about the data structure, machine learning models are excellent at processing big datasets and spotting complex patterns (Joseph, 2022).

- **Random Forest**. This ensemble learning method, based on decision trees, is known for its robustness against noise and overfitting. While effective for feature-rich datasets, Random Forest is limited in its ability to account for temporal dependencies, making it less ideal for time series data (Joseph, 2022).
- **Gradient Boosting**. Iterative models like XGBoost and LightGBM improve predictive accuracy by minimising errors in successive models. Although they are strong in regression tasks, they lack inherent mechanisms for modelling sequential relationships (López de Prado, 2018).

Deep learning models have emerged as powerful tools for financial time series analysis because they can model nonlinear and complex dependencies over time.

- **Long Short-Term Memory (LSTM)**. A variant of recurrent neural networks (RNNs), LSTM addresses the vanishing gradient problem by incorporating memory cells and gates, enabling it to retain long-term information (Nielsen, 2019). This makes LSTM particularly effective for capturing long-term dependencies in financial data.

- **Gated Recurrent Unit (GRU)**. A simplified version of LSTM, GRU achieves similar performance with fewer parameters and computational requirements, making it a practical alternative for resource-constrained environments (Joseph, 2022).

The stock market prediction field uses LSTM and GRU models which demonstrate better performance than statistical models according to Nielsen (2019).

The forecasting tool Prophet operates with scalability features and user-friendly design for business and financial applications. The forecasting tool Prophet developed by Meta enables users to model seasonality together with holidays and missing data points (Hyndman & Athanasopoulos, 2018). The Prophet's forecasting accuracy suffers from its assumptions because stock market data lacks strong or consistent seasonality patterns (Joseph, 2022). The tool provides valuable forecasting capabilities through its ability to generate quick and interpretable results.

The stock market forecasting field has traditionally employed statistical models including ARIMA and SARIMA. The widespread adoption of these methods stems from their basic design and clear understanding but they struggle to handle complex financial data characteristics like non-linear patterns and random variations (Hyndman & Athanasopoulos, 2018). Machine learning and deep learning models solve these problems through direct pattern learning from data.

The research shows that LSTM and GRU models excel at stock price prediction because they effectively handle temporal relationships and non-linear patterns in financial data (Nielsen, 2019). Hybrid approaches that unite statistical techniques with machine learning algorithms demonstrate potential by combining the advantages of both approaches (López de Prado, 2018).

2.3. Gaps And Potential Improvements In Time Series Forecasting

The forecasting techniques have advanced, yet several gaps persist:

- Most research studies analyze individual models without performing comprehensive evaluations between traditional and machine learning and deep learning approaches on the same dataset.
- The research fails to connect academic forecasting models with practical real-world applications through the development of user-friendly tools.
- The majority of studies analyze stock price data exclusively yet adding economic indicators like GDP and inflation would improve model precision (Joseph, 2022).

The capstone project resolves these research gaps through model performance evaluation between statistical and machine learning and deep learning approaches and practical user-friendly interface development.

2.4. Data Challenges In Time Series Forecasting

Financial market time series forecasting requires unique approaches to solve problems that distinguish it from other fields. The modeling process becomes difficult because economic data contains high volatility and non-stationarity and noise. The challenges require strong methodologies together with thorough examination of financial data complexities to achieve successful results.

The main difficulty in financial forecasting arises from non-stationarity. Stock prices violate the stationarity requirement of statistical models ARIMA and SARIMA because they exhibit trends and abrupt regime shifts and heteroskedasticity which means volatility changes across time (Hyndman & Athanasopoulos, 2018). The preprocessing methods of rolling averages, logarithmic transformations and differencing are commonly used to stabilize the mean and variance but they may simplify the underlying data structure.

The forecasting process becomes more complex because financial data exhibits non-linear and chaotic characteristics. Traditional statistical models prove useful for linear dependency detection but they fail to detect the complex economic time series relationships and dependencies. The machine learning models Random Forest and Gradient Boosting provide flexibility yet they depend mostly on feature engineering to detect temporal dependencies (Joseph, 2022). These models become ineffective at detecting important relationships when lag features and rolling statistics are not properly designed.

Sensitivity to outliers and extreme events, like market crashes or economic shocks, is another significant obstacle. Model accuracy may be disproportionately impacted by these anomalies, especially for metrics like RMSE that penalize significant deviations more severely (Lopez de Prado, 2018). Certain machine learning models and statistical models have a tendency to overreact to these anomalies, producing forecasts that are not reliable in harsh circumstances.

Some of these issues are addressed by deep learning models, like LSTM and GRU, which directly learn nonlinear relationships and temporal dependencies. To prevent overfitting, these models need a lot of data and are computationally costly (Nielsen, 2019). Additionally, their "black-box" nature makes interpretation challenging, a significant drawback in finance, where transparency and explainability are critical for decision-making.

Financial forecasting faces challenges because of both noise and limited data availability. Stock market data lacks obvious periodicity which makes SARIMA models less effective compared to domains with well-defined seasonal patterns (Hyndman & Athanasopoulos, 2018). High-frequency trading data introduces noise that obscures vital trends and patterns in the data. The problems can be reduced through data smoothing and outlier removal but precise calibration is required to avoid damaging the original signal.

The integration of external indicators remains an area where significant research is needed. The inclusion of macroeconomic indicators and alternative data sources such as social media sentiment can enhance forecasting precision even though financial data remains valuable (Joseph, 2022). The process of preprocessing and feature selection and model integration becomes complex when these data sources are merged.

The challenges demonstrate why it is essential to implement a hybrid strategy which leverages multiple approaches. Financial time series consist of multiple complex elements which become more understandable when statistical models are combined with machine learning or deep learning techniques (Lopez de Prado, 2018).

2.5. Practical Challenges and Considerations in Financial Forecasting

The practical application of stock market trend forecasting demands a combination of theoretical precision with real-world operational limitations. The deployment of statistical and machine learning and deep learning models for financial forecasting needs to solve three main challenges: computational efficiency and interpretability and adaptability to changing market conditions.

The selection of models depends on both the nature of the data and the needs of the users. Statistical models such as ARIMA and SARIMA remain popular choices because they are easy to understand and work well with linear and seasonal patterns. The models fail to handle financial data that exhibits high volatility together with non-linear patterns. Random Forest and Gradient Boosting machine learning models perform better on datasets containing complex feature interactions. These models need extensive preprocessing and feature engineering work to reach their highest possible performance levels (Joseph, 2022).

The practical forecasting process depends heavily on computational requirements. Deep learning models including LSTM and GRU need substantial processing power to perform training and prediction operations. The excellent ability of these models to detect non-linear temporal dependencies makes them

difficult to implement in real-time or resource-limited environments because of their extended training duration and substantial data needs (Nielsen, 2019). Deep learning models become impractical for numerous organizations because they need expensive infrastructure even though simpler models achieve comparable accuracy levels.

Financial forecasting requires interpretability as a crucial factor because predictions directly affect important business decisions. The "black-box" nature of deep learning and machine learning models hinders transparency although they provide better predictive capabilities. Statistical models produce transparent results which make them more suitable for applications requiring auditability and trust (Lopez de Prado, 2018). The combination of machine learning or deep learning predictive power with statistical interpretability through hybrid approaches represents a promising solution.

Financial markets require forecasting models to demonstrate both resilience and adaptability because of their dynamic nature. Models need to handle unexpected market changes including geopolitical events and economic shocks that disrupt established market trends. The accuracy of models in dynamic environments depends on regular model updates and retraining. Online learning and dynamic model updates enable models to adapt to new data without needing complete retraining (Joseph, 2022).

The practical deployment of forecasting systems depends on their scalability features. The deployment process becomes simpler through Docker and other containerization tools because they ensure uniform model and dependency operation across different platforms. The method enables model deployment on cloud platforms such as AWS or Google Cloud through its scalability features which also supports multi-user environments and expands accessibility. Real-time applications require both low latency and optimized prediction pipelines as essential factors.

The integration of external data sources including macroeconomic indicators and social media sentiment and ESG (Environmental, Social, Governance) metrics enhances both forecasting accuracy and contextual relevance. The integration of these additional signals proves beneficial yet presents difficulties in data cleaning and feature extraction and stock price data temporal alignment (Hyndman & Athanasopoulos, 2018).

The success of practical financial forecasting depends on matching model selection with deployment strategies and resource allocation to specific task requirements. A balanced approach between these factors leads to the development of stronger forecasting systems that produce actionable results.

2.5. Interim Conclusions

Time series forecasting functions as an analytical necessity which operates throughout energy management and healthcare and finance industries. The development of time series forecasting models from basic statistical approaches to complex machine learning and deep learning systems shows how modern data complexity needs more advanced and flexible predictive models. The field has made progress but financial forecasting remains challenging because of noise and volatility and non-stationarity which persistently affect forecasting models.

The basic statistical models ARIMA and SARIMA maintain their fundamental position because they offer straightforward application and straightforward interpretation. These models maintain their financial applications but their ability to detect non-linear market relationships and sudden market fluctuations remains limited. The predictive power of machine learning models Random Forest and Gradient Boosting depends on preprocessing and feature engineering steps to detect temporal relationships. Deep learning models LSTM and GRU demonstrate exceptional ability to handle both sequential and non-linear data streams. These models present difficulties for real-world application because they lack interpretability and require significant computational resources.

The literature points out multiple vital research gaps together with multiple research opportunities that need investigation. The scientific literature requires additional research that evaluates statistical models against machine learning and deep learning approaches in different market conditions. The understanding of external data source benefits including alternative data and macroeconomic indicators remains limited despite their potential to boost forecasting accuracy.

The review establishes that hybrid approaches which unite different method strengths represent an essential requirement for success. These methods address complex financial forecasting challenges to develop models which maintain both precision and adaptability. The transition from academic research to real-world implementation demands equal focus on user-friendly interfaces and deployment capabilities together with computational performance.

The capstone project aims to advance stock market forecasting through a reliable application for users while assessing different forecasting models.

CHAPTER 3. METHODOLOGY

3.1. Overview

This project focuses on developing a time series forecasting application to predict stock market trends. The application supports multiple statistical, machine learning, and deep learning models. It is built using Python with a user-friendly interface powered by Streamlit, allowing users to:

- Load stock market data from Yahoo Finance.
- Select and configure forecasting models.
- Visualise training, testing, and forecasted results.
- Generate and download reports in PDF format.

3.2. Data Collection

The project obtained its data from Yahoo Finance through the yfinance library. Users can get historical stock prices by selecting specific tickers and defining their desired date ranges. The application focuses on the main variables which include Open, close, high, low and volume prices. The method provides flexibility to analyze different stocks and time periods because it meets the needs of various users.

3.3. Data Preprocessing

The project required specific data preprocessing operations to align the dataset with multiple forecasting models used in the project. The imputation techniques handled missing data points in the dataset to prevent bias from occurring because of incomplete records.

The machine learning models required lagged feature generation to detect past value dependencies. The models need these features to understand time-based relationships in the data because time series forecasting depends on this understanding.

Deep learning models including LSTMs and GRUs need preprocessing because they need additional inputs to minimize all features in the dataset. The MinMaxScaler technique scales numerical data into a specific narrow range between 0 and 1. Normalisation enables sequential neural networks to function properly while preventing problems that stem from variable magnitude differences.

The data was split into two sections where 95% formed the training set and 5% became the testing set to enhance model performance evaluation. The models would learn historical patterns from this data while maintaining a separate portion for predictive accuracy assessment.

3.4. Forecasting Models

The application supports a diverse set of forecasting models to handle different data characteristics and requirements:

- **Statistical Models**.

  ARIMA and SARIMA are used to identify the linear patterns and seasonal trends in the data. They are effective for datasets with consistent seasonality and relatively straightforward structures, offering interpretable and robust predictions under such conditions.

- **Machine Learning Models**

  Random Forest and Gradient Boosting utilize engineered features, such as lagged variables, to make predictions. These models excel in capturing complex relationships in the data and are well-suited for datasets where non-linear interactions and feature importance play a critical role.

- **Deep Learning Models**

  Complex temporal dependencies in sequential data can be modeled using Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) networks. They are very useful for financial time series forecasting because they can use recurrent layers to learn long-term dependencies and non-linear patterns.

- **Specialised Framework**

  Because Prophet, a tool created by Meta, handles trends, seasonality, and missing data in an intuitive manner, it is included. It is a great option for rapid, comprehensible forecasts due to its user-friendly design, particularly in business and finance contexts.

The preprocessed dataset served as input for model training while the hyperparameters were optimized to achieve the best possible forecasting results. The forecasting models ARIMA and SARIMA used recursive forecasting to update their predictions dynamically when new data became available. The

method ensured that the models produced accurate and adaptable predictions which could adapt to changing data patterns.

3.5. Technical Details of the Forecasting Models

To provide a clearer understanding of the computational methods underlying the forecasting models, this section explores the technical foundations and implementation choices of each approach.

- **ARIMA and SARIMA**

ARIMA (AutoRegressive Integrated Moving Average) models predict upcoming values through the combination of three elements: the autoregressive term (AR) which links current values to their historical observations and the differencing term (I) which stabilizes the series by removing trends and the moving average term (MA) which connects current observations to residual errors from lagged observations (Hyndman & Athanasopoulos, 2018). The SARIMA model extends this framework through seasonal pattern inclusion by adding seasonal AR, MA and differencing components. The model consists of two sets of parameters: (p, d, q) for the non-seasonal component and (P, D, Q, m) for the seasonal component. The pmdarima library performed an automatic stepwise search to determine the parameters which resulted in the best fit to the observed data.

- **Random Forest**

The ensemble learning algorithm Random Forest generates multiple decision trees during training which produce predictions by averaging the output of each tree (Joseph, 2022). The construction of each tree uses random subsets of features and samples which reduces variance and improves generalization. The project used scikit-learn to implement Random Forest with 100 trees and a fixed random seed for reproducibility. The algorithm suits financial data analysis because it handles complex data dimensions and prevents overfitting.

- **Gradient Boosting**

The ensemble method Gradient Boosting constructs models sequentially by having each new model try to minimize the remaining errors from the previous model (López de Prado, 2018). The project employed the scikit-learn implementation with 100 boosting iterations and a learning rate of 0.1. The repeated error correction process in Gradient Boosting leads to strong predictive results but needs proper parameter adjustment to prevent overfitting.

- **LSTM (Long Short-Term Memory)**

The vanishing gradient problem in traditional RNNs is addressed by LSTM networks through their implementation of gating mechanisms according to Nielsen (2019). The gates in LSTM networks control information flow which allows the network to discover long-term dependencies in sequential data. The project used tensorflow to build a single-layer LSTM network with 50 units. The MinMaxScaler normalization method was applied to the data before reshaping it into three-dimensional sequences that matched the network's input specifications.

- **GRU (Gated Recurrent Unit)**

GRU networks offer a simplified alternative to LSTM by merging the forget and input gates into a single update gate. This reduces computational overhead while preserving the ability to learn long-term dependencies (Joseph, 2022). The GRU model in this project employed a similar architecture to the LSTM implementation, trained over 20 epochs with a batch size of 16.

- **Prophet**

Prophet, developed by Meta, decomposes time series data into trend, seasonality, and holiday effects, fitting an additive model to capture these components (Hyndman & Athanasopoulos, 2018). It is advantageous when interpretability and quick iteration are essential. Prophet automatically handles missing data and outliers, making it robust for noisy financial data, though its underlying assumptions may limit performance in non-seasonal stock market data.

The training process utilized 95% of the data for each model while the remaining 5% served as evaluation data. The models employed recursive forecasting when needed to update their predictions automatically after receiving new data points. The selection of hyperparameters followed best practices in literature while manual adjustments were made to achieve optimal computational efficiency and predictive accuracy.

3.6. Model Evaluation

Three important metrics were used to evaluate the forecasting models' accuracy:

- *RMSE* (Root Mean Square Error)

The root mean square error (RMSE) stands as one of the most common metrics to evaluate the precision of continuous prediction models. The calculation involves taking the square root of the mean of squared differences between observed and forecasted values. The mathematical definition of RMSE is:

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

- $\hat{y}_i$ represents the predicted value
- y_i represents the actual value
- n is the number of observations

A lower RMSE value indicates better model performance, indicating fewer errors on average. RMSE is particularly useful when significant prediction errors need to be penalised more heavily, making it sensitive to outliers in the data.

- *MAPE* (Mean Absolute Percentage Error)

The MAPE reflects accuracy as a percentage. It is beneficial for interpreting model performance in percentage terms. It calculates the average absolute percentage error between the predicted and actual values, providing an intuitive understanding of accuracy. MAPE is expressed as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100$$

- $\hat{y}_i$ represents the predicted value
- y_i represents the actual value
- n is the number of observations

MAPE is easy to interpret—values closer to 0% indicate better model accuracy. However, it has a drawback: it becomes unreliable when actual values (yiy_iyi) are close to zero, as this can inflate percentage errors.

- R^2 (R-squared)

R^2 measures the proportion of variance in the actual values explained by the model. It is a dimensionless metric ranging from 0 to 1 (or negative for poorly performing models), where higher values indicate better model fit. The formula R^2 is:

$$R^2 = 1 - \frac{\Sigma_{i=1}^{n}(\hat{y}_i - y_i)^2}{\Sigma_{i=1}^{n}(y_i - \underline{y})^2}$$

- $\hat{y}_i$ represents the predicted value
- y_i represents the actual value
- n is the number of observations

A R^2 value of 1 indicates perfect predictions, while 0 means the model does no better than simply predicting the mean of the actual values. Negative R^2 values suggest that the model performs worse than a baseline prediction.

The combination of RMSE, MAPE and R^2 provides a robust framework for evaluating forecasting models:

- *RMSE* assesses the magnitude of prediction errors, penalising large deviations more heavily.
- *MAPE* offers a percentage-based view of model accuracy, which is easy to interpret and helpful in comparing across datasets.
- R^2 Quantifies how well the model explains the variance in the data, indicating its overall fit.

The capstone project used these metrics to ensure a comprehensive evaluation of the models, allowing for meaningful comparisons of their predictive powers. This multi-faceted evaluation method, in addition to highlighting each model's advantages and disadvantages, helped to inform judgments regarding the models' applicability for stock market forecasting.

3.7. Integration with External Data

The financial industry represents a major advancement of traditional time series forecasting methods through its use of external data sources. The current forecasting application relies on historical stock prices but incorporating external indicators would provide more detailed contextual data that better represents stock price influencing factors.

3.7.1. Macroeconomic and Financial Indicators

Stock market cycles experience significant influence from macroeconomic factors which include GDP growth rates together with inflation and interest rates and unemployment statistics (Joseph, 2022). The market trends tend to become bullish when GDP grows but rising interest rates signal tighter monetary policy which leads to decreased stock prices. Forecasting models become more effective at predicting market movements by incorporating these indicators which represent both external cyclical forces and internal stock price dynamics.

The forecasting models can incorporate these macroeconomic indicators through exogenous variables (or external regressors). For example:

- In the ARIMA framework, the **ARIMAX** variant (AutoRegressive Integrated Moving Average with eXogenous inputs) allows the inclusion of external regressors, extending the model's ability to account for outside influences (Hyndman & Athanasopoulos, 2018).

- In machine learning models such as Random Forest or Gradient Boosting, macroeconomic indicators can be added as additional features alongside lagged stock price data.

- Deep learning models can incorporate these indicators as additional input channels or concatenated features to learn joint representations of stock prices and economic context.

3.7.2. Sentiment and Alternative Data

The analysis of investor sentiment and market expectations can be enhanced through alternative data sources which include social media feeds and news sentiment analysis and ESG scores in addition to macroeconomic indicators (López de Prado, 2018). The sentiment scores derived from financial news and Twitter data can detect short-term investor sentiment changes before actual price movements occur.

The integration of these data sources requires:

- The analysis of sentiment data through natural language processing (NLP) methods enables the creation of numerical sentiment scores.
- The scores obtained from sentiment analysis can be used as additional features to enhance the training of machine learning or deep learning models with traditional stock price data.

- Nielsen conducted research in 2019 which showed financial models produced better predictions through market data integration with sentiment analysis during volatile market periods.

3.7.3. Technical and Practical Challenges

While the benefits of integrating external data are substantial, several practical and technical challenges must be considered:

- The frequency of external data varies between indicators such as quarterly GDP figures and daily stock prices. The indicators need proper resampling or interpolation to match the daily or weekly time frequency of stock market data.
- The models need external data transformation through percentage change calculations and lag operations and rolling average computations to become effective.
- Standard ARIMA models lack built-in support for external variables so users need to implement workarounds or select different modeling approaches.
- Sentiment data and macroeconomic indicators face challenges from revisions and noise and inconsistencies. The forecasting pipeline requires complete data cleaning and validation procedures to prevent errors from entering the system.

3.7.4. Potential Impact and Future Directions

Financial forecasting models can achieve better predictive power through the implementation of external data integration. The historical stock price data served as the basis for this capstone project but future research should combine macroeconomic indicators with sentiment data for better results.

- Improve accuracy in detecting cyclical market reversals (e.g., bull and bear market transitions).
- Provide early warning signals for market corrections or rallies based on external economic forces.
- Increase the interpretability of forecasts by linking predictions to broader economic narratives (e.g., monetary policy shifts, geopolitical events).

External data types	Examples	Potential benefits
Macroeconomic Indicators	GDP growth, inflation rates, and interest rates	Capture cyclical trends and economic context affecting stock prices
Sentiment Data	Social media sentiment, financial news sentiment	Gauge investor mood and detect market turning points earlier
ESG Scores	Environmental, Social, Governance ratings	Enhance understanding of long-term risk factors and trends
Event-based Data	Commodity prices, sector indexes (e.g., oil prices, tech index)	Identify sector-specific drivers that can affect market performance

Figure 3.7.1. A table summarising possible external data types and their potential benefits for integration into forecasting models.

The application requires external data integration as its essential transformation step to evolve from a stock-price-only model into a complete market analysis tool. The integration of external data complicates data management and model engineering but enables the development of more precise and contextually aware forecasts that match the intricate elements affecting stock market cycles.

3.8. User Interface And Deployment

The Python framework Streamlit enabled developers to build interactive data-driven web applications which formed the basis of the forecasting application. The application interface operates at a basic level which allows non-technical users to follow the forecasting process. Users can access historical stock data from Yahoo Finance through the application by choosing a stock ticker and defining a date range (see Appendix A, Figure A1). After data retrieval the system displays the information in a table format which provides a brief summary of the dataset (refer to Appendix A, Figure A2).

The application provides seven forecasting models which include ARIMA, SARIMA, Random Forest, Gradient Boosting, LSTM, GRU and Prophet for users to select from (see Appendix A, Figure A1). Users can execute the forecasting process immediately after choosing a model through the interface. The backend trains the selected model after processing input data before displaying predictions to the application users. The visualizations include an overlay of predicted and actual values for comparison and a chart showing training and testing subset divisions of the dataset (see Appendix A, Figures A4,

A5). The results present information in a concise manner which makes them easy to understand thus creating a better user experience.

The application generates PDF reports through its analysis overview feature which stands as a crucial element. The reports generated by the system contain essential performance metrics RMSE and MAPE together with detailed graphical representations and tabular information (Appendix A, Figures A6 and A7). The functionality enables users to efficiently distribute their analyses and store them for future reference.

The application received Docker containerization to ensure deployment stability and platform compatibility. The entire application with dependencies and runtime environment exists as a portable Docker image after containerization. The application runs identically across Windows, macOS and Linux systems because this method removes platform-related dependency mismatches. Docker Compose simplifies application deployment by automating setup and execution so users can start the application without much effort.

The containerized design enables scalability and facilitates local deployment of the application. The Docker image can be deployed to cloud platforms such as AWS or Google Cloud to provide remote accessibility which enables multi-user environments and expands possible use cases. The application demonstrates technical excellence and user-friendly functionality through the combination of Docker reliability with Streamlit simplicity which makes it a useful tool for stock market trend prediction.

3.9. Software Architecture and Design

The forecasting application software architecture follows a modular design which provides scalability and user-friendly features. The system uses a layered modular design which divides data processing from model training and user interface components to improve maintainability and extensibility.

At a high level, the system consists of three primary components:

1. **Data Collection and Preprocessing Layer**

The first layer functions to acquire historical stock market data which it then transforms into a format suitable for modeling. The yfinance library retrieves data from the Yahoo Finance API through user-defined stock tickers and time periods.

The preprocessing module handles:

- **Imputation**: Filling missing values to maintain data integrity.
- **Lag Feature Engineering**: Creating lagged features for machine learning models to capture temporal dependencies.
- **Normalisation**: Scaling data for deep learning models using MinMaxScaler, ensuring numerical stability and efficient training.

2. **Model Training and Forecasting Layer**

The application supports a diverse set of forecasting models, each encapsulated within dedicated modules in the /models directory. This separation allows for easy addition or modification of models in the future. Key models include:

- **Statistical models** (/models/arima/). ARIMA and SARIMA are implemented using the pmdarima library for detecting linear trends and seasonality.
- **Machine learning models** (/models/random_forest/ and /models/gbr/). Random Forest and Gradient Boosting, leveraging scikit-learn for regression-based forecasting.
- **Deep learning models** (/models/lstm/ and /models/gru/). LSTM and GRU models built with *tensorflow*, ideal for learning complex non-linear patterns in sequential data.
- **Specialised framework** (/models/prophet/). Prophet by Meta is known for its interpretability and handling of holiday effects.

Each model module contains two main scripts:

- train.py: Handles the training process using the preprocessed data.
- prediction.py: Generates forecasts based on trained models.

3. **User Interface and Visualisation Layer**

The user interface is implemented using *Streamlit*, a lightweight Python framework that simplifies the development of interactive web applications. The interface provides:

- Stock ticker and date range selection.

- Model selection dropdown for choosing from the available forecasting algorithms.
- Data visualisation, including plots of training and testing data, and forecast overlays using *Plotly* for interactive graphs.
- Metrics display (RMSE, MAPE, R^2) to assess model performance.
- A feature to generate PDF reports (*fpdf* library) summarising the analysis with key visualisations and tabular data.

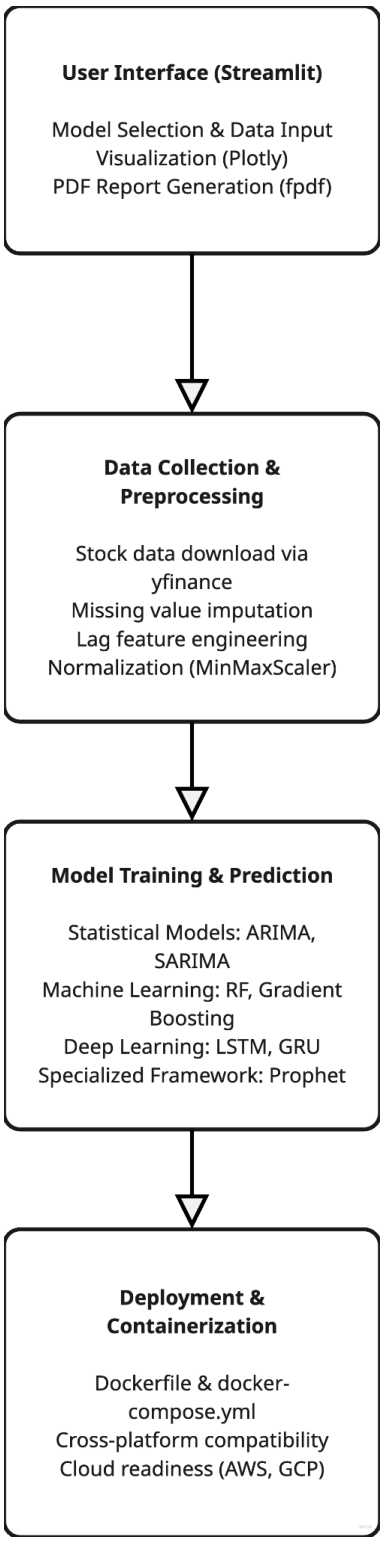

Figure 3.7.1. Software architecture of the time series forecasting application. The architecture consists of four layers: user interface, data collection and preprocessing, model training and prediction, and deployment.

3.10. Interim Conclusions

The methodology employed in this capstone project provided a solid foundation for developing a practical and versatile forecasting application. The project successfully addressed the challenges inherent in stock market prediction by combining robust data preprocessing techniques, diverse forecasting models, and a user-centric interface.

Including multiple forecasting models—ranging from statistical methods like ARIMA and SARIMA to machine learning and deep learning approaches—ensured the application could cater to various data characteristics and user needs. Each model was carefully trained and evaluated using industry-standard metrics such as RMSE, MAPE, and R^2, providing a comprehensive understanding of their strengths and limitations. Statistical models excelled in capturing linear trends and seasonality, while machine learning models leveraged feature engineering to address non-linear patterns. Although computationally intensive, deep learning models offered advanced capabilities for modelling temporal dependencies.

In order to get the input ready for analysis, data preprocessing was essential. The models were given clean and consistent inputs thanks to methods like imputation for missing values, lagged feature generation for machine learning models, and normalization for deep learning. The reliability of the model evaluations was further improved by the deliberate separation of data into training and testing subsets.

The application provides an easy-to-use interface that enables smooth Docker deployment. The interface enables users to easily load data and select models while viewing results and creating detailed reports which simplifies the forecasting process. The Dockerized design provides a foundation for future scalability and expansion because it ensures consistent platform performance.

The methodology integrates data processing with model implementation and deployment to create an intuitive application that offers extensive forecasting capabilities. The comprehensive approach creates a solid foundation to achieve project objectives while enabling future development opportunities.

CHAPTER 4. RESULTS AND ANALYSIS

This chapter presents the results of forecasting stock prices using multiple models. It evaluates their performance using RMSE, MAPE, and R^2 metrics. The discussion highlights the models' strengths, limitations, and insights from the visualisations.

4.1. Input Data and Parameters

The input data for this project consisted of historical stock prices for Apple Inc. (AAPL), collected from Yahoo Finance using the *yfinance* library. The dataset included daily prices from January 1, 2010, to the end of 2023. The variables considered were:

- **open** – the opening price for each trading day.
- **close** – the closing price for each trading day.
- **high** – the highest price was recorded during the day.
- **low** – the lowest price was recorded during the day.
- **volume** – the total number of shares traded during the day.

For this analysis, the **closing** price was used as the primary target variable for forecasting, a key indicator of daily market sentiment. The data was divided into training (95%) and testing (5%) subsets, with training data spanning most of the dataset and the testing subset covering the final segment of the time series.

4.2. Model Performance Summary

The performance of each forecasting model was evaluated using RMSE, MAPE, and R^2, as shown in Table 3.1. These metrics provide a comprehensive assessment of prediction accuracy and model reliability.

Table 4.1. Model Performance Metrics

Model	RMSE	MAPE	R2
ARIMA	3.11	0.01	0.98
SARIMA	3.11	0.01	0.98
LSTM	4.35	0.02	0.96
GRU	4.88	0.02	0.95
Random Forest	30.04	0.11	-0.84
Gradient Boosting	30.05	0.11	-0.84
Prophet	37.02	0.15	-1.79

4.3. Analysis of Results

4.3.1 Statistical Models

ARIMA and SARIMA outperformed other models, achieving RMSE values of 3.11 and R^2 scores of 0.98. These models effectively captured the data's linear and seasonal patterns. Their low MAPE (1%) highlights their predictive accuracy.

4.3.2. Deep Learning Models

LSTM and GRU provided competitive results but underperformed relative to the statistical models. LSTM achieved an RMSE of 4.35 and an R^2 of 0.96, while GRU recorded an RMSE of 4.88 and an R^2 of 0.95. Despite their strength in handling sequential data, the models faced challenges related to the limited dataset size, high noise in stock prices, and the lack of additional input features to provide context for their predictions.

4.3.3. Machine Learning Models

Random Forest and Gradient Boosting were the weakest performers among the models. With RMSE values of around 30 and negative R^2 scores, these models struggled to capture temporal dependencies effectively. Their reliance on lagged features was insufficient for modelling the complex behaviour of stock prices.

4.3.4. Prophet

Prophet performed the poorest overall, with an RMSE of 37.02 and an R^2 of -1.79. The model's assumptions of strong seasonality and consistent trends were misaligned with the non-linear, volatile nature of stock market data, leading to significant forecasting errors.

4.4. Visual Analysis

The visualisations in Figures 4.1 to 4.7 illustrate the alignment between predicted and actual stock prices for each model. While ARIMA and SARIMA closely align with actual values, Prophet and the machine learning models show significant deviations.

Figure 4.1. ARIMA Forecast

Figure 4.2. SARIMA Forecast

Figure 4.3. LSTM Forecast

Figure 4.4. GRU Forecast

Figure 4.5. Random Forest Forecast

Figure 4.6. Gradient Boosting Forecast

Figure 4.7. Prophet Forecasting

4.5. Challenges and Future Directions

The performance of LSTM, GRU, and Prophet was constrained by a number of factors. The small dataset size caused problems for deep learning models, making generalization challenging. Furthermore, significant patterns were obscured by the high level of noise in stock prices. The Prophet's efficacy was hampered by its reliance on seasonality assumptions, which were inconsistent with the erratic and non-linear character of stock market data.

Several improvements can be investigated to enhance model performance:

1. Deeper forecasting insights may be obtained by including extra features like technical indicators (like moving averages and RSI) or macroeconomic variables (like GDP and interest rates).
2. Complex temporal relationships may be better captured by utilizing transformer-based models, such as Temporal Fusion Transformers.
3. The advantages of both approaches could be combined by combining deep learning models with statistical techniques.
4. Accuracy could be increased by optimizing LSTM and GRU hyperparameters using strategies like grid search.
5. Expanding the dataset to include intraday or minute-level data could offer more prosperous training signals for deep learning models.

4.6. Interim Conclusions

The variable accuracy of forecasting models in predicting stock prices is highlighted in this chapter. The most dependable models were statistical ones like ARIMA and SARIMA, whereas deep learning models produced results that were competitive but marginally less accurate. Prophet and machine learning models found it difficult to adjust to the intricacies of stock market data. In order to provide more precise and reliable forecasting solutions, this project can be expanded by tackling the issues that have been identified and investigating the suggested future paths.

CHAPTER 5. IMPLEMENTATION AND DEPLOYMENT

The chapter describes the technical implementation and deployment process of the time series forecasting application. The chapter describes the user interface design and forecasting model integration and software architecture and deployment procedure. The system design aimed to develop an efficient and effective forecasting solution.

5.1. Software Architecture

The application uses a monolithic architecture which is built in Python. All key components (data retrieval, preprocessing, model execution, and evaluation) are encapsulated in a single cohesive service.

5.1.1. Back-End Logic

The backend is structured using a modular approach to separate core functionalities. This design provides flexibility, scalability, and ease of maintenance. It is containerised using Docker to ensure consistent deployment across environments.

The responsibilities of the backend are the following:

- Retrieve historical stock market data from Yahoo Finance.
- Preprocess the raw data into a clean and usable format.
- Train and generate predictions using various forecasting models, including statistical and machine learning approaches.
- Evaluate model performance using predefined metrics RMSE, MAPE, and R^2.
- Respond to user requests via the Streamlit frontend, providing interactive visualisations and downloadable reports.

The main components of the backend are the following:

1. **API Layer**:

Acts as the entry point for the backend, handling user interactions from the Streamlit frontend. Routes user requests (e.g., selecting a stock or forecasting model) to other backend components.

2. **Data Retrieval**:

Interfaces with the Yahoo Finance API using the ***yfinance*** library to fetch historical stock market data (e.g., opening/closing prices, volume, etc.) based on user inputs like stock symbols and date ranges.

3. **Data Preprocessing**:

Cleans and transforms the raw stock market data to make it suitable for forecasting models. It fills in missing values, normalises data using ***MinMaxScaler*** from ***scikit-learn***, and engineers lagged features for time-series forecasting.

4. **Model Engine**:

Implements and executes multiple forecasting models:

- **ARIMA/SARIMA**. Statistical forecasting using the ***pmdarima*** library.
- **Prophet**. Intuitive statistical forecasting for seasonal trends and missing data.
- **Deep Learning Models (LSTM/GRU)**. Neural networks implemented using ***tensorflow*** to capture temporal dependencies in data.

The component handles training and prediction workflows, depending on the user's input. Generates future forecasts based on trained models and user inputs (e.g., time horizon).

5. **Evaluation Module**:

Calculates predefined metrics such as RMSE (Root Mean Square Error), MAPE (Mean Absolute Percentage Error), and R^2 (coefficient of determination) to assess model performance. Returns these metrics along with the predictions to the user interface.

6. **Visualisation and Reporting**:

Generates interactive visualisations of raw data, model outputs, and evaluation results using ***plotly*** and Streamlit-native charts. Produces downloadable PDF reports using ***fpdf***, summarising the forecast results and performance metrics.

The following steps summarise the backend workflow:

1. **User Request**:
    - A user interacts with the frontend (e.g., selects a stock ticker, date range, and forecasting model).
    - The frontend sends a request to the backend API.
2. **Data Retrieval**:
    - The backend uses ***yfinance*** to fetch historical stock data from Yahoo Finance.
    - The raw data (e.g., open/close prices, volumes) is returned to the backend.
3. **Data Preprocessing**:
    - The raw data is cleaned and transformed:
        - Missing values are filled or removed.
        - Features are engineered (e.g., lagged variables for time-series context).
        - Data is normalised for compatibility with deep learning models.
4. **Model Training and Prediction**:
    - The backend's **Model Engine** selects and executes the chosen model:
        - Statistical models (e.g., ARIMA, Prophet) are trained and generate predictions.
        - Deep learning models (e.g., LSTM, GRU) use preprocessed time-series data to produce forecasts.
5. **Evaluation**:
    - The backend evaluates the forecasted results using RMSE, MAPE, and R^2 metrics.
    - These metrics are returned to the frontend and included in reports.
6. **Response**:
    - The backend sends the forecasted results, performance metrics, and visualisations back to the frontend.
    - If requested, a downloadable PDF report is generated.

5.1.2. Front-End Interface

The frontend of the time-series forecasting application serves as the **user interface** for interacting with the backend. It allows users to:

1. Select stock symbols and time ranges for analysis.
2. Choose forecasting models to generate predictions.
3. Visualise historical data, predictions, and evaluation metrics interactively.
4. Download reports summarising results, metrics, and visualisations.

The frontend is built using ***Streamlit***, a Python-based framework for building interactive web applications.

The frontend operates with a declarative architecture pattern because it generates the user interface through user input and backend response processing. The application maintains a lightweight design which provides real-time integration with backend services to retrieve data and process requests and display results.

The main components of the frontend are the following:

User Interaction Layer:

- Provides an intuitive and interactive interface using **Streamlit**.
- Allows users to:
    - Input stock tickers and date ranges.
    - Select forecasting models (e.g., ARIMA, Prophet, LSTM).
    - Configure model parameters (e.g., training duration, prediction horizons).
- Includes dynamic widgets like dropdown menus, sliders, buttons, and text input fields to capture user preferences.

Visualisation Layer:

- Handles the rendering of interactive visualisations to display:
    - Historical stock data.
    - Model predictions and their confidence intervals.
    - Model evaluation metrics (e.g., RMSE, MAPE).
- Uses **Plotly** for interactive, zoomable charts and **Streamlit-native visualisations** for lightweight static plots.

API Integration Layer:

- Communicates with the backend via HTTP requests to fetch data, run models, and retrieve results.
- Sends user-selected parameters (e.g., stock symbol, date range, and model type) to the backend and receives processed results (e.g., predictions and metrics) for display.

Report Generation Layer:

- Provides a feature for generating downloadable **PDF reports** summarising:
    - Forecasted results.
    - Model performance metrics.
    - Visualisations of historical and predicted data.
- Uses backend-generated reports, which are served to the frontend as downloadable files.

Error Handling and Notifications:

- Displays error messages in case of:
    - Invalid stock symbols or date ranges.
    - Backend connectivity issues (e.g., API timeouts).
    - Model-specific errors (e.g., unsupported configurations).
- Ensures a seamless user experience by notifying users of invalid inputs or retrying failed operations.

5.2. Integration of Forecasting Models

The application supports a range of forecasting models, enabling users to select the most suitable model for their specific needs. The supported models include ARIMA, SARIMA, Prophet, Random Forest, Gradient Boosting, LSTM, and GRU. Each model follows a standardised workflow:

1. The raw stock data is preprocessed to fit the requirements of the selected model. For example:
    - Lagged features are created for machine learning models, such as Random Forest and Gradient Boosting.
    - Deep learning models, such as LSTM and GRU, use normalised inputs for sequential processing.
2. The selected model is executed for the dataset.

- Using automated methods, statistical models (e.g., ARIMA and SARIMA) are configured with optimal parameters.
- Machine learning models are trained on lagged features to capture historical dependencies.
- Deep learning models process sequences of data to learn temporal patterns.
- Prophet provides interpretable trend-based predictions with minimal configuration.

3. The results of each model are evaluated using RMSE, MAPE, and R^2 to assess its accuracy and reliability.

This standardised approach simplifies the integration of models into the system, enabling users to switch seamlessly between them.

The following steps describe how the frontend interacts with users and the backend:

1. **User Input**:
   - The user selects a stock ticker, date range, and forecasting model using interactive widgets in the Streamlit interface.
2. **Backend Request**:
   - The frontend sends the user's input to the backend API, which processes the request (e.g., fetching data, running models) and returns results.
3. **Data Rendering**:
   - The frontend receives processed results (e.g., predictions, metrics) from the backend and:
     - Displays interactive time-series plots (e.g., stock prices and forecasts).
     - Renders model performance metrics in a tabular format.
4. **Report Download**:
   - The user can click a "Download Report" button to generate a **PDF report** containing forecasts, metrics, and visualisations.

5.3. User Interface Development

The user interface (UI) is a critical part of the system, enabling users to interact with the application effortlessly. Built with *Streamlit*, the UI offers the following features:

- **Data Loading**

Users can select a stock ticker (e.g., AAPL) and a start date. The system retrieves historical stock data using the *yfinance* library and displays it in a table for review.

- **Model Selection**

A dropdown menu allows users to choose from the available forecasting models. Once a model is selected, it can be executed directly from the UI.

- **Visualisation**

The application generates dynamic line charts to visualise historical stock data, training and testing splits, and predicted versus actual values. These visualisations provide an intuitive way to evaluate forecasting results.

- **Result Presentation**

Key performance metrics, including RMSE, MAPE, and R^2, are displayed alongside a table of predicted versus actual values. This ensures that users can quickly understand the model's accuracy.

- **Report Generation**

A "Create Report" button generates a comprehensive PDF report with metrics, visualisations, and data tables. This feature makes it easy to archive or share the analysis.

The UI's minimalistic design ensures a smooth and user-friendly experience for individuals with varying levels of technical expertise.

5.4. Deployment

The deployment process leverages *Docker* to ensure consistency, scalability, and ease of use. The application is containerised to ensure that all dependencies and configurations are packaged together, simplifying deployment on any compatible environment.

This deployment strategy ensures that the application is portable, can be easily shared with other users, and can be deployed in cloud environments for broader accessibility.

5.5. Interim Conclusions

The chapter focuses on deploying and implementing the time series forecasting application. The application becomes accessible and user-friendly through the Streamlit-powered user interface while the

forecasting models integrate smoothly and the modular architecture ensures reliable functionality. The system becomes more portable and scalable through Docker deployment which makes it a dependable stock market forecasting tool.

CHAPTER 6. CONCLUSIONS

The project's fundamental objective involved developing a time series forecasting application which utilizes historical data to predict stock market trends. The application delivers a performance comparison between statistical models and machine learning models and deep learning models. Users with minimal technical expertise can forecast stock market trends because the system features a modular structure alongside an intuitive interface and robust evaluation framework.

The results demonstrated that statistical models including ARIMA and SARIMA achieved superior accuracy through their RMSE values of 3.11 and R2 scores of 0.98. The models work well for structured time series analysis because they effectively detect seasonal and linear patterns in the data. The deep learning models LSTM and GRU delivered competitive results yet their full potential required additional data and processing power to achieve. The stock market data proved challenging for Prophet and other machine learning models including Random Forest and Gradient Boosting because they struggled to adapt to its complex non-linear and volatile nature.

Through its user interface the application enables users to perform three main functions which include data loading and model selection and result viewing. The Docker deployment of the application enhances its accessibility together with scalability and portability features. The integration of PDF reporting enables users to generate detailed summaries of their forecasting outcomes.

The project achieved its objectives by delivering a practical solution for stock market trend prediction.

6.1. Practical Applications

The time series forecasting application has several practical applications:

1. Investors and financial analysts can utilise the application to forecast stock price trends, thereby enhancing portfolio management and informed trading strategies.

2. The application can be an educational resource for teaching time series forecasting techniques and model evaluation.
3. The application framework can be extended to other domains, such as sales forecasting, supply chain management, energy demand prediction, and stock prices.

APPENDIX

Appendix A: Screenshots of the Application

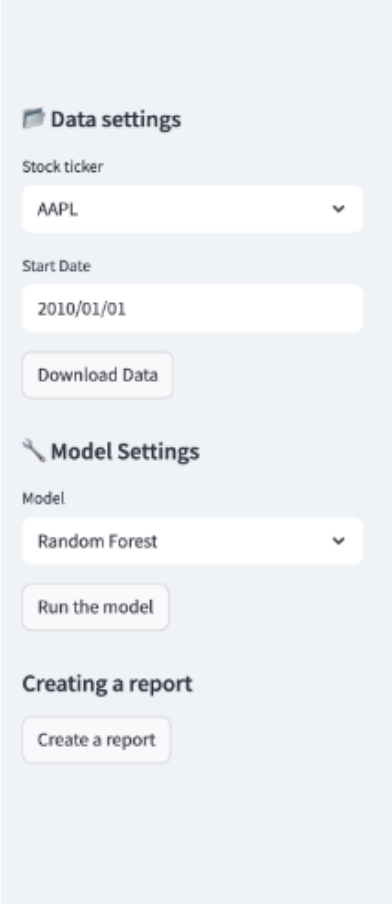

Figure A1. Sidebar panel of the forecasting application, showing options for selecting stock tickers, start dates, forecasting models, and report creation.

	Date	Close	High	Low	Open	Volume
		AAPL	AAPL	AAPL	AAPL	AAPL
0	2010-01-04 00:00:00	6.4319	6.4466	6.3829	6.4145	493729600
1	2010-01-05 00:00:00	6.443	6.4794	6.4091	6.4496	601904800
2	2010-01-06 00:00:00	6.3405	6.4686	6.3339	6.443	552160000
3	2010-01-07 00:00:00	6.3288	6.3715	6.2828	6.364	477131200
4	2010-01-08 00:00:00	6.3709	6.3715	6.2831	6.3204	447610800
5	2010-01-11 00:00:00	6.3147	6.4015	6.2648	6.3955	462229600
6	2010-01-12 00:00:00	6.2429	6.3045	6.2038	6.287	594459600
7	2010-01-13 00:00:00	6.3309	6.3393	6.1341	6.2474	605892000
8	2010-01-14 00:00:00	6.2942	6.3252	6.2819	6.3147	432894000

Figure A2. Example of downloaded stock market data (Open, High, Low, Close, and Volume) for the selected ticker (AAPL).

Figure A3. Line plot visualisation of the historical stock prices retrieved from Yahoo Finance.

Figure A4. Visualisation of the training and testing data split used in model training and evaluation.

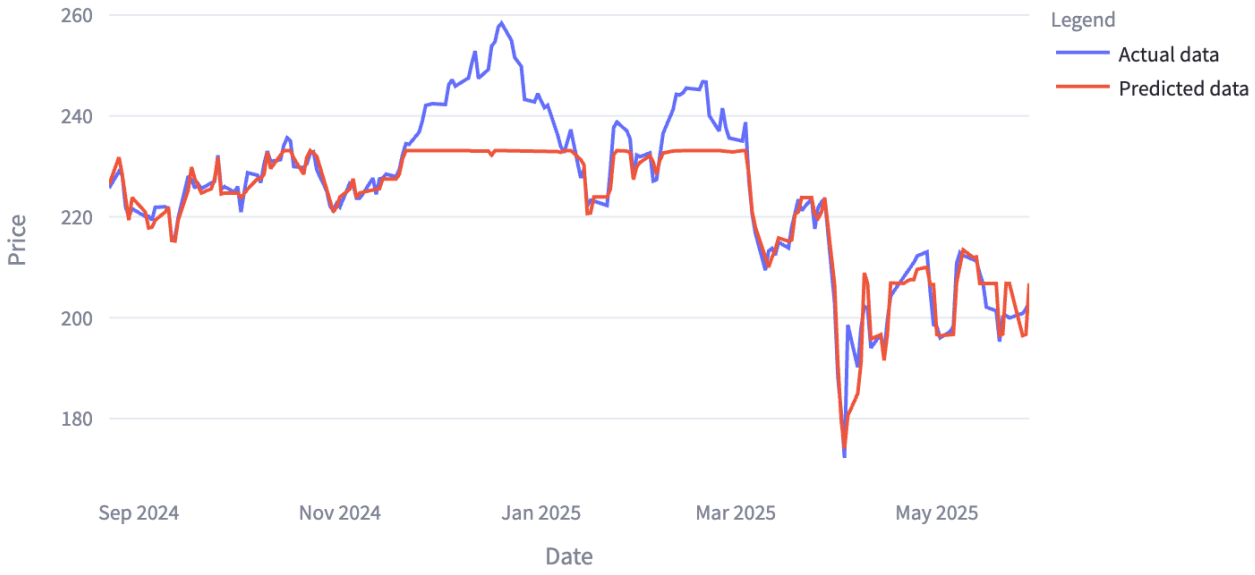

Figure A5. Overlay of actual and predicted stock prices, showing the selected model's (Random Forest) predictive performance on the test dataset.

RMSE: 6.79

MAPE: 1.82%

R-квадрат: 0.82

Figure A6. Key performance metrics of the selected forecasting model (Random Forest), including RMSE, MAPE, and R-squared.

	date	actual	predicted
3684	2024-08-23 00:00:00	225.6966	226.371
3685	2024-08-26 00:00:00	228.9851	231.7984
3686	2024-08-27 00:00:00	228.1979	228.3932
3687	2024-08-28 00:00:00	221.9897	223.7906
3688	2024-08-29 00:00:00	220.0764	219.3201
3689	2024-08-30 00:00:00	221.601	223.8201
3690	2024-09-03 00:00:00	220.0465	220.9369
3691	2024-09-04 00:00:00	220.1362	217.7806
3692	2024-09-05 00:00:00	219.339	217.9262
3693	2024-09-06 00:00:00	221.8801	219.3201

Figure A7. Comparison of predicted and actual stock prices for the selected test period.