UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Research and Development

**BACHELOR THESIS**

By

**Nguyen Anh Duy – BI12-127**

Data Science

Title:

**“Research and implementation of deep learning methods in stock price analysis and forecasting”**

External supervisor: Nguyen Duc Hoan

Internal supervisor: Nghiem Thi Phuong

**Hanoi, 2024**



Abstract

In the context of increasingly volatile and complex financial markets, the demand for accurate stock price prediction methods has become critical. This thesis investigates and implements deep learning techniques, specifically Long Short-Term Memory (LSTM) networks, in conjunction with traditional statistical models like Autoregressive Integrated Moving Average (ARIMA) to enhance stock price analysis and forecasting.

Through a comprehensive methodology, this research compares the performance of ARIMA and LSTM models in predicting stock prices. The findings reveal that while LSTM models excel in capturing complex, non-linear relationships and long-term dependencies in the data, ARIMA models are proficient in handling linear trends and seasonal patterns.

The structure of this thesis is organized as follows:

1. In the first chapter, we introduce the core concepts and objectives of the research.
2. The second chapter provides a comprehensive literature review on LSTM and ARIMA models. It delves into the existing techniques and methods used in constructing and analyzing historical stock data to make predictions about stock market movements. This review serves to contextualize our approach within the broader field
3. In the third chapter, we detail the methodology and experimental design used in this study. We also describe the datasets used for empirical evaluation and the metrics for performance evaluation.
4. The final chapter presents the conclusions drawn from our findings. This paper summarizes the main contributions of our work, discusses the implications of our results for the field of stock price prediction, and suggests potential directions for future research.

Acknowledgement

I would like to express my deepest gratitude to all those who have contributed to the successful completion of this thesis.

First and foremost, I extend my sincere thanks to my external supervisor, Mr. Nguyen Duc Hoan, for his invaluable guidance, insightful feedback, and unwavering support throughout the research process. His expertise and experience have been instrumental in shaping the direction of this study.

I am also profoundly grateful to my internal supervisor, Ms. Nghiem Thi Phuong, whose continuous encouragement, constructive criticism, and meticulous attention to detail have significantly enhanced the quality of this work. Her academic rigor and dedication have been a source of inspiration.

I would like to acknowledge the University of Science and Technology of Hanoi for providing me with the resources and a conducive environment for research. The academic community and facilities at the university have been fundamental to my learning and research endeavors.

Additionally, I am thankful to my colleagues and friends for their moral support and helpful discussions that have enriched my understanding and broadened my perspective on the subject matter.

Last but not least, I express my heartfelt appreciation to my family for their constant support, patience, and encouragement throughout my academic journey. Their unwavering belief in me has been the driving force behind my achievements.

This thesis would not have been possible without the contributions and support of all these individuals and institutions. Thank you all.

Table of Contents

[Chapter 1: Introduction 7](#_Toc171101743)

[1.1 Problem Statement 7](#_Toc171101744)

[1.2 Project Scope 7](#_Toc171101745)

[1.3 Thesis Structure 8](#_Toc171101746)

[1.4 Objective 9](#_Toc171101747)

[Chapter 2: Literature Review 10](#_Toc171101748)

[2.1 ARIMA Model 10](#_Toc171101749)

[2.2 LSTM Model 11](#_Toc171101751)

[2.3 Random Forest Model 12](#_Toc171101752)

[2.4 Tools and Technologies 14](#_Toc171101753)

[Chapter 3: Methodology 15](#_Toc171101754)

[3.1 Phase 1 15](#_Toc171101755)

[Object 1: Data collection 15](#_Toc171101756)

[Object 2: Data Mining 17](#_Toc171101757)

[Object 3: ARIMA Model 24](#_Toc171101758)

[Object 4: LSTM Model 26](#_Toc171101759)

[Object 5: Random Forest 33](#_Toc171101760)

[3.2 Phase 2 34](#_Toc171101761)

[3.3 Comparison of ARIMA and LSTM and Random Forest 35](#_Toc171101762)

[3.4 Discussion 36](#_Toc171101763)

[Chapter 4: Conclusion and Future Work 37](#_Toc171101764)

[4.1 Summary of Findings 37](#_Toc171101765)

[4.2 Limitations 38](#_Toc171101766)

[4.3 Future Work 39](#_Toc171101767)

List of Abbreviations

|  |  |
| --- | --- |
|  |  |
| ARIMA | Autoregressive Integrated Moving Average |
| LSTM | Long Short-Term Memory |
| RNN | Recurrent Neural Network |
| ETL | Extract, Transform, Load |
| CSV | Comma-Separated Values |
| ERD | Entity-Relationship Diagram |
| ACF | Autocorrelation Function |
| PACF | Partial Autocorrelation Function |
| R² | Coefficient of Determination |
| MAPE | Mean Absolute Percentage Error |
| RMSE | Root Mean Squared Error |
| MSE | Mean Squared Error |
| AAA | Stock symbol for a specific company (AAA) |
| AAPL | Stock symbol for Apple Inc. |
| ACB | Stock symbol for Asia Commercial Bank |
| BID | Stock symbol for Bank for Investment and Development of Vietnam |
| CTG | Stock symbol for Vietnam Joint Stock Commercial Bank for Industry and Trade |
| FPT | Stock symbol for FPT Corporation |
| GAS | Stock symbol for PetroVietnam Gas Joint Stock Corporation |
| NVDA | Stock symbol for NVIDIA Corporation |
| VCB | Stock symbol for Vietcombank |
| VNM | Stock symbol for Vinamilk |

List of Figures

[Figure 1: ARIMA Network Architecture 11](#_Toc171329218)

[Figure 2: LSTM Network Architecture 12](#_Toc171329219)

[Figure 3: Database Schema 16](#_Toc171329220)

[Figure 4: Statistic Table of each stock 19](#_Toc171329221)

[Figure 5: Predict and Actual Price Visualize 27](#_Toc171329222)

[Figure 6: Loss and Validation Visualize 31](#_Toc171329223)

[Figure 7: Predict and Actual Result Comparison Visualize 33](#_Toc171329224)

List of Table

[Table 1: Results of Dickey-Fuller Test Table 21](#_Toc171101311)

[Table 2: ARIMA metrics result table 26](#_Toc171101312)

[Table 3: LSTM Hyperparameter table 27](#_Toc171101313)

[Table 4: LSTM metric result table 32](#_Toc171101314)

[Table 5: Random Forest metrics result table 33](#_Toc171101315)

[Table 6: Comparison metric table 35](#_Toc171101316)

# Chapter 1: Introduction

## 1.1 Motivation and Problem Statement

### 1.1.1 Motivation

In the context of an ever-fluctuating and risky financial market, accurately predicting stock prices has become one of the major challenges for investors and researchers. Traditional forecasting models, such as ARIMA, have played an important role in predicting financial time series. However, with the rapid development of artificial intelligence, deep learning models like LSTM (Long Short-Term Memory) are emerging as a potential tool for analyzing complex data and uncovering hidden relationships within time series data. The goal of this research is to combine both ARIMA and LSTM models to improve the accuracy of stock price predictions. The combination of traditional methods (ARIMA) and modern techniques (LSTM) not only provides a balanced solution between complexity and efficiency but also contributes to building a better tool for stock price forecasting. This research aims to offer a new perspective on how modern technologies can be applied in financial analysis, thereby delivering practical value to investors and stakeholders.

### 1.1.2 Problem Statement

Against the backdrop of increasingly volatile and complex financial markets, the demand for accurate stock price prediction methods has become more urgent. Investors and market analysts continuously seek advanced tools and models to optimize investment decisions and mitigate risks. This project employs two popular time-series forecasting models, ARIMA and LSTM, to predict stock prices in response to this demand. ARIMA, a traditional statistical model, has proven effective in time-series analysis, while LSTM, a type of recurrent neural network, excels in handling cyclical data and long time-series sequences. By comparing and evaluating the performance of these two models, this study not only provides insights into their effectiveness but also offers practical recommendations for investors and stakeholders on the application of advanced predictive techniques in practice.

## 1.2 Objectives and Scope

## 1.2.1 Objectives

This project aims to improve stock price analysis and forecasting by leveraging both traditional statistical models and advanced deep learning techniques. The following objectives outline the research focus:

1. **Evaluation of Stock Price Prediction Models**: Conduct a comprehensive evaluation of various models currently applied in stock price analysis, ranging from simple statistical models to machine learning and deep learning techniques. This evaluation will help establish a foundation for developing an optimized forecasting model.
2. **Study of Deep Learning Techniques (LSTM Networks)**: Investigate and analyze deep learning methods, particularly Long Short-Term Memory (LSTM) networks, for their suitability in constructing predictive models for stock price forecasting.
3. **Development of Optimized Forecasting Models**: Design and develop models using LSTM and ARIMA to predict stock prices. These models will be trained and tested on real market data, and their performance will be evaluated based on key metrics like accuracy and error rate.
4. **Comprehensive Report on Statistical Analysis and Forecasting**: Produce a detailed report outlining the statistical analysis, model development, and stock price predictions. This report will summarize the findings, compare model performances, and provide actionable insights for stock market analysis.

## 1.2.2 Research Subjects and Scope

This research focuses on applying advanced time series forecasting techniques to the analysis of stock prices. Specifically, it involves the study and application of deep learning models, such as Long Short-Term Memory (LSTM) networks, and traditional models, like Autoregressive Integrated Moving Average (ARIMA), to enhance the prediction of stock prices in volatile market environments. The research aims to explore the effectiveness of these models in capturing both linear and non-linear trends in stock price movements.

**Subjects**: The primary subjects of this study are the stock price datasets of five companies: AAPL, GOOGL, FPT, ACB, and BID, spanning from January 1, 2015, to September 11, 2024. These companies represent different sectors, which adds diversity to the dataset and allows the models to be tested across varying market conditions.

**Scope**:

* Investigating the potential of LSTM and ARIMA models in stock price forecasting.
* Implementing experimental programs using real-world stock price data to assess the effectiveness of the proposed models.
* Comparing the performance of LSTM and ARIMA in predicting stock prices, with a focus on accuracy, training time, and computational efficiency.
* Proposing practical recommendations for investors and market analysts on model applicability based on the empirical results.

The research is conducted using the Design Science methodology, which involves the systematic evaluation of existing techniques, model design, and performance testing to offer optimized solutions for stock price forecasting.

## 1.2.3 Related Studies

## 1.2.4 Contribution

This research focuses on enhancing stock price analysis and forecasting by combining modern deep learning methods with traditional statistical models. The main contributions of this thesis are as follows:

1. **Data Mining for Stock Price Forecasting**: We employ advanced data mining techniques to preprocess and analyze stock market data, ensuring its quality for accurate forecasting. This includes handling missing data, addressing outliers, and ensuring stationarity for time series models like ARIMA and LSTM. These steps enhance the robustness of the forecasting models.
2. **Optimizing LSTM and ARIMA Models**: We optimize LSTM and ARIMA models by tuning key hyperparameters, such as the learning rate and sequence length for LSTM, and selecting appropriate parameters (p, d, q) for ARIMA. This improves the models' ability to capture both linear and non-linear trends in stock prices. The comparison of both models provides insights into their effectiveness under different market conditions.

These contributions aim to enhance the efficiency and accuracy of stock price forecasting by addressing common challenges in financial time series analysis, such as data volatility and model overfitting.

# Chapter 2: Literature Review

## 2.1 Null Hypothesis

In statistics, the **null hypothesis** (denoted as H₀) is a fundamental concept used in hypothesis testing. It represents the default or initial assumption about a population parameter or a statistical relationship. Typically, the null hypothesis states that there is **no effect**, **no difference**, or **no relationship** between the variables being studied. It acts as a baseline that researchers aim to challenge or disprove using statistical evidence.

The null hypothesis can be mathematically represented as:

*H0 : μ1 = μ2*

Where μ1​ and μ2 represent the means of two populations being compared. This formula indicates that, under the null hypothesis, there is no significant difference between the two means.

The null hypothesis is typically contrasted with the **alternative hypothesis** (denoted as H₁ or Ha), which suggests that there is an effect or a significant difference. In the case of the medication study, the alternative hypothesis might state:

*H1 : μ1 ≠ μ2*

This means that there is a statistically significant difference in blood pressure between the treatment and control groups. The goal of hypothesis testing is to determine whether the observed data provides enough evidence to **reject the null hypothesis** in favor of the alternative hypothesis.

### 2.1.1 Steps in Hypothesis Testing

1. **Formulate the Hypotheses**: The first step is to clearly define the null and alternative hypotheses. The null hypothesis (H₀) is the assumption of no effect or no relationship, while the alternative hypothesis (H₁) suggests the presence of an effect or relationship.
2. **Choose a Significance Level (α)**: This represents the threshold for rejecting the null hypothesis. A common significance level is 0.05, meaning there is a 5% chance of rejecting the null hypothesis when it is actually true (Type I error).
3. **Calculate a Test Statistic**: Based on the sample data, a test statistic (such as a t-score or z-score) is calculated. This statistic is used to compare the observed data against the expected data under the null hypothesis.
4. **Determine the p-value**: The p-value represents the probability of observing the sample data (or something more extreme) assuming the null hypothesis is true. If the p-value is less than the chosen significance level (α), the null hypothesis is rejected.
5. **Draw a Conclusion**: Based on the p-value and the test statistic, a decision is made about whether to reject the null hypothesis or fail to reject it. Importantly, failing to reject the null hypothesis does not prove that it is true—it simply means there isn’t enough evidence to support the alternative hypothesis.

### 2.1.2 Importance of Null Hypothesis

The null hypothesis plays a crucial role in scientific research as it provides a clear framework for testing and validating theories. It ensures that researchers approach their studies with an open mind, assuming no effect or relationship until sufficient evidence is presented. This reduces the likelihood of drawing false conclusions based on random fluctuations in the data.

In practice, the null hypothesis helps researchers avoid **confirmation bias**—the tendency to search for or interpret information in a way that confirms pre-existing beliefs. By starting with the assumption that there is no effect, researchers are required to rigorously test their data before making claims about the presence of an effect or relationship.

## 2.2 Augmented Dickey-Fuller

In econometrics and time series analysis, the Augmented Dickey-Fuller (ADF) test is widely used to assess the stationarity of a time series. The ADF test is an extension of the original Dickey-Fuller test, designed to account for potential autocorrelation in the data by incorporating lagged differences of the series.

The primary objective of the ADF test is to determine whether a time series contains a unit root, which indicates non-stationarity. If a time series exhibits non-stationarity, statistical inferences drawn from the data may be unreliable and misleading. The null hypothesis (H₀) of the ADF test posits the presence of a unit root (i.e., the series is non-stationary), while the alternative hypothesis (H₁) suggests that the series is stationary.

**ADF Test Formula:**

*Δyt = α + βt + γy{t-1} + ∑ δᵢ Δy{t-i} + εt*

**Where:**

* ***Δyt*** is the first difference of the time series,
* ***t*** represents time,
* ***yt*** is the time series,
* ***α*** is the constant,
* ***βt*** is the trend,
* ***γ*** tests whether the time series has a unit root.
* ***εt*** represents white noise.

If the test statistic is smaller than the critical value, the null hypothesis of a unit root is rejected, indicating that the time series is stationary. Conversely, failure to reject the null hypothesis implies that the series is non-stationary and contains a unit root.

The accuracy of the ADF test depends on selecting an appropriate number of lags for the differenced series, which is often determined using information criteria such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC). Proper lag selection ensures efficient estimation and avoids biased results due to omitted variables.

## 2.3 ARIMA Model

### 2.3.1 Overview

The **Autoregressive Integrated Moving Average (ARIMA)** model is one of the most widely used approaches in time series analysis, particularly in the field of econometrics and financial forecasting. Introduced by Box and Jenkins (1976), the ARIMA model combines three key elements—autoregression (AR), differencing (I), and moving averages (MA)—to model time series data with trends and seasonality, while also accounting for randomness in the data.

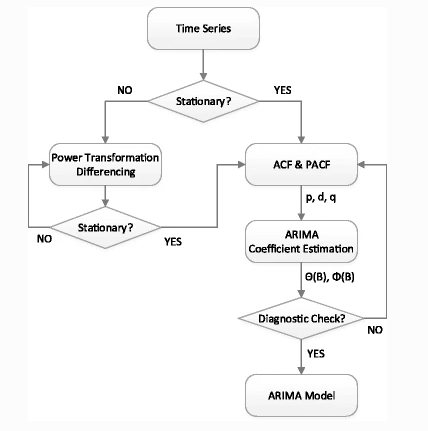


Figure 1: ARIMA Network Architecture

**Components of the ARIMA Model**

* **Autoregressive (AR) Component**: The autoregressive part of the model indicates that the current value of the series is related to its previous values. The AR term is specified by the parameter **p**, which represents the number of lagged observations included in the model. In simple terms, the AR component assumes that past values of the variable help predict its current value.
* **Integrated (I) Component**: The integrated component represents the differencing of the time series to make it stationary. Stationarity is a critical assumption for many time series models, as non-stationary data can lead to unreliable forecasts. The parameter **d** denotes the number of differencing operations required to remove trends and achieve stationarity.
* **Moving Average (MA) Component**: The moving average part of the model captures the dependency between an observation and a residual error from a moving average model applied to lagged errors. The **q** parameter represents the number of lagged forecast errors included in the model.

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

* ***p***: number of lag observations,
* ***d***: number of times the raw observations are differenced,
* ***q***: size of the moving average window.

**Formula:**

*yt = c + φ₁y{t-1} + φ₂y{t-2} + … + φpy{t-p} + θ₁ε{t-1} + … + θqε{t-q} + εt*

**Where:**

* ***yt*** is the value at time ttt,
* ***ϕ*** are the AR parameters,
* ***θ*** are the MA parameters,
* ***εt*** represents white noise or random error.

### 2.3.2 Importance of the ARIMA Model in Forecasting

The ARIMA model is particularly useful for forecasting data that exhibit patterns of autocorrelation and non-stationarity, which are common characteristics in economic and financial time series, such as stock prices, GDP, or inflation rates. By modeling both the underlying structure (through AR and MA components) and making the data stationary (through differencing), ARIMA provides a flexible and robust framework for generating forecasts.

One of the significant advantages of ARIMA is its ability to handle various types of data patterns, including both short-term dependencies and long-term trends. This makes it especially popular for applications in financial markets, where asset prices and returns often exhibit trends, volatility clustering, and correlations with past values.

### 2.3.3 Model Selection and Estimation

The selection of the ARIMA model parameters (p, d, q) is often done using diagnostic tools such as the **autocorrelation function (ACF)** and the **partial autocorrelation function (PACF)**. These functions help determine the appropriate number of lags for the AR and MA terms. Additionally, criteria such as the **Akaike Information Criterion (AIC)** and **Bayesian Information Criterion (BIC)** are employed to select the best-fitting model by balancing goodness-of-fit with model complexity.

Moreover, ARIMA can be extended to handle seasonal data through the **Seasonal ARIMA (SARIMA)** model, which incorporates seasonal autoregressive, differencing, and moving average terms to account for repeating patterns that occur at regular intervals.

### 2.3.4 Limitations of the ARIMA Model

Despite its popularity, the ARIMA model has certain limitations. One major drawback is its reliance on the assumption of linear relationships in the data, which may not always hold, especially in highly volatile or complex systems like financial markets.

ARIMA models are also less effective in capturing non-linear patterns or structural breaks, which may occur due to economic shocks or other external factors. Additionally, the model’s performance can degrade when applied to high-frequency or highly volatile time series data, as it may fail to fully capture intricate dynamics.

## 2.4 LSTM Model

### 2.4.1 Overview

The **Long Short-Term Memory (LSTM)** model is a type of recurrent neural network (RNN) that has gained significant popularity in recent years for time series forecasting, particularly in complex and non-linear data environments such as financial markets. Introduced by Hochreiter and Schmidhuber (1997), the LSTM model was designed to overcome the limitations of traditional RNNs, particularly the issue of **vanishing gradients**, which hinder RNNs from learning long-term dependencies in sequential data.

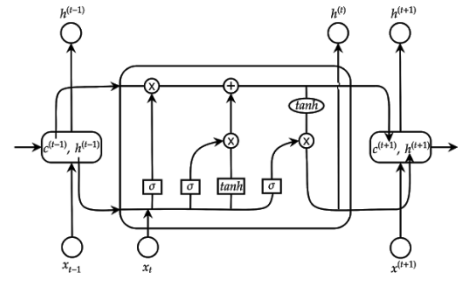


Figure 2: LSTM Network Architecture

**Components of the LSTM Model**

The LSTM model is characterized by its **memory cells**, which are capable of selectively remembering or forgetting information over extended time intervals. Unlike traditional RNNs, LSTMs can retain past information over many time steps, making them highly suitable for time series forecasting, where long-term patterns or trends are critical for accurate predictions.

**LSTM** units consist of several key components:

1. **Forget gate:**

*ft = σ(Wf • [h{t-1}, xt] + bf)*

1. **Input gate:**

*it = σ(Wi • [h{t-1}, xt] + bi)*

1. **Output gate:**

*ot = σ(Wo • [h{t-1}, xt] + bo)*

The LSTM model can be represented mathematically as follows:

*ht = ot \* tanh(Ct)*

*Ct = ft \* Ct-1 + it \* Ct*

Where:

* ***xt***​ is the input at time step t,
* ***h*t** is the hidden state at time step t,
* ***Ct*** ​ is the cell state,
* ***ft​***, ***it*​**, and ***ot***​ are the forget, input, and output gate activations, respectively,
* ***W*** and ***b***are the weight matrices and bias vectors associated with the gates,
* ***σ*** represents the sigmoid activation function, and
* ***tanh*** represents the hyperbolic tangent activation function.

### 2.4.2 Importance of the LSTM Model in Time Series Forecasting

LSTM networks are particularly well-suited for financial and economic time series forecasting because of their ability to model **long-term dependencies** and **non-linear relationships** in the data. This makes them highly effective in environments where traditional linear models, such as ARIMA, may struggle, especially when dealing with complex stock market behaviors, high volatility, and unpredictable trends.

* **Handling Long-Term Dependencies**: In time series data, particularly in financial markets, past events often have a lingering effect on future outcomes. For instance, stock price movements may depend not only on recent events but also on market trends from months or even years ago. LSTM’s memory cells allow it to capture these long-term dependencies without losing information over time, a limitation that traditional RNNs face due to the vanishing gradient problem.
* **Non-linear Relationships**: Financial time series data, such as stock prices, interest rates, and commodity prices, often exhibit complex non-linear relationships due to market forces, investor behavior, and macroeconomic factors. LSTMs are highly capable of capturing these non-linear relationships due to their flexible architecture, making them superior to linear models like ARIMA when dealing with highly volatile or irregular data patterns.
* **Suitability for High-Frequency Data**: LSTM models are particularly useful for high-frequency financial data, such as minute-by-minute or second-by-second stock price changes, where short-term volatility and noise make traditional models less effective. LSTM networks can handle the intricate patterns and autocorrelations that arise in such data, enabling more accurate forecasts even in fast-moving markets.

### 2.4.3 Model Selection and Training

Training an LSTM model involves optimizing the model parameters (weights and biases) using **backpropagation through time (BPTT)**, a variant of the backpropagation algorithm tailored for sequential data. The model learns by minimizing a loss function, such as **mean squared error (MSE)** or **mean absolute error (MAE)**, using optimization techniques like **stochastic gradient descent (SGD)** or **Adam**.

When applying LSTM models, researchers must carefully select hyperparameters such as the number of hidden units, the number of layers, the sequence length (time steps), and the learning rate. These choices can significantly impact the model’s performance and ability to generalize to unseen data.

In addition to tuning hyperparameters, techniques like **dropout** (a regularization method) and **early stopping** (terminating training when performance on a validation set plateaus) are commonly used to prevent overfitting and enhance the model's robustness.

### 2.4.4 Limitations of the LSTM Model

Despite its advantages, LSTM models also have certain limitations. One key drawback is the significant computational resources required for training, particularly when dealing with large datasets or high-dimensional time series. LSTM models often require more time and memory than simpler models like ARIMA or even other machine learning models due to their complex architecture.

Additionally, LSTMs may struggle to interpret and explain the underlying structure of time series data, which can be a critical limitation in fields like finance where model transparency and explainability are essential for regulatory and decision-making purposes. The **“black box”** nature of LSTM models makes it difficult to understand the exact relationships and patterns they are capturing, which can limit their adoption in certain industries.

Moreover, LSTMs may require significant fine-tuning to achieve optimal performance, and the process of hyperparameter selection can be time-consuming. Without proper tuning, LSTM models can underperform or overfit the data, leading to inaccurate forecasts.

# Chapter 3: Methodology

## 3.1 Data collection

### 3.1.1 Data Exploratory

The datasets provided are historical stock price data for five companies: FPT, GOOGL (Alphabet), AAPL (Apple), ACB, and BID. Below is an overview of the dataset characteristics based on these files:

**Timeframe:**

* **Date Range:** The datasets cover the period from **01/01/2015 to 09/11/2024**. This span allows for over nine years of stock market data for each company, providing a comprehensive view of stock price performance during this period.

**Data Volume:**

Each dataset contains daily stock data for each of the five companies, spanning from **January 1, 2015, to September 11, 2024**, covering a total of **3,541 days**. Given that the stock market operates approximately 250 trading days per year, this period results in around **2,450 data points** per company. These figures reflect the actual trading days available within this time frame, providing a substantial dataset for accurate time series forecasting and analysis.

**Data Fields:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Price** | **Open** | **High** | **Low** | **Volume** | **Change %** |
| 09/11/2024 | 222.66 | 221.46 | 223.09 | 217.89 | 44.59M | 1.16% |
| 09/10/2024 | 220.11 | 218.92 | 221.48 | 216.73 | 51.59M | -0.36% |
| 09/09/2024 | 220.91 | 220.82 | 221.27 | 216.71 | 67.18M | 0.04% |
| 09/06/2024 | 220.82 | 223.95 | 225.24 | 219.77 | 48.42M | -0.70% |

Table 1: Dataset Example

Each dataset contains the following **seven fields** (columns), which are typical for financial data analysis:

1. **Date**: The trading day in MM/DD/YYYY format, providing the daily timestamps for the stock prices.
2. **Price**: The closing price of the stock on a given day. This reflects the last traded price before the market closed.
3. **Open**: The opening price of the stock at the start of the trading day. It indicates the first price at which the stock was traded when the market opened.
4. **High**: The highest price at which the stock was traded during the day. It indicates the peak value the stock reached within a given trading day.
5. **Low**: The lowest price at which the stock was traded during the day. It represents the minimum value the stock reached during a single trading session.
6. **Volume (Vol.)**: The total number of shares traded during the day. It reflects the market activity and liquidity of the stock on a particular day, often influencing price movements.
7. **Change %**: The percentage change in the stock price compared to the previous trading day's closing price. It shows how much the stock price has increased or decreased, helping investors track performance trends.

**Companies:**

The five companies included in this analysis are:

1. **FPT**: A leading technology company in Vietnam.
2. **GOOGL (Alphabet)**: The parent company of Google, a global tech giant.
3. **AAPL (Apple)**: One of the largest technology companies in the world.
4. **ACB**: A major bank in Vietnam.
5. **BID**: Another prominent Vietnamese bank.

These datasets provide ample data for a thorough analysis of stock trends, including price fluctuations, volatility, and trading volumes across different market sectors. ​

## 3.2 Data Analyst and Preprocessing

### 3.2.1: Outlier

In financial data analysis, it is essential to identify and handle outliers to ensure the accuracy of predictions and models. Outliers can distort patterns, leading to unreliable conclusions. In this study, we observed significant outliers in the closing prices of two stocks, AAPL and GOOGL, on February 27, 2016. Specifically, the closing price for AAPL on this date was recorded as 96.95, while GOOGL's closing price stood at 724.62. These values are significantly higher than their expected ranges based on historical price trends, indicating potential anomalies.

To address this issue, we employed a simple yet effective outlier treatment method: replacing the outlier values with the corresponding values from the preceding trading day, February 26, 2016. For AAPL, the closing price on February 26, 2016, was 24.23, and for GOOGL, the corresponding closing price was 36.24. By substituting the outliers with these values, we aimed to maintain the dataset's integrity and ensure that the extreme values did not unduly influence the model's predictions.

**Before Treatment:**

A graph with blue lines

Description automatically generated

* AAPL closing price on February 27, 2016: 96.95

A graph with a line graph

Description automatically generated

* GOOGL closing price on February 27, 2016: 724.62

These outlier values deviate significantly from the trend, potentially skewing the data.

**After Treatment:**

* AAPL closing price on February 27, 2016, was adjusted to match the previous day's closing price: 24.23
* GOOGL closing price on February 27, 2016, was adjusted to match the previous day's closing price: 36.24

By applying this approach, we ensured a smoother transition in the stock price data, thereby reducing the impact of these anomalies on the predictive models. This method allows for more accurate forecasting, as the extreme values are replaced with more representative figures, aligned with the general market trend during that period.

### Statistic

The table presents statistical metrics summarizing stock price data for five companies (AAPL, ACB, BID, FPT, and GOOGL), and these values play a crucial role in understanding stock behavior, particularly when using models like LSTM (Long Short-Term Memory) and ARIMA (AutoRegressive Integrated Moving Average) to predict stock prices. Below is our statistic for each stock

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Count** | **mean** | **std** | **min** | **max** |
| AAPL | 2440 | 91.99 | 61.42 | 22.59 | 234.82 |
| ACB | 2431 | 302.87 | 423.51 | 3.91 | 1808.4 |
| BID | 2418 | 27744.74 | 11432.17 | 8197.0 | 54400.0 |
| FPT | 2418 | 45019.09 | 32566.69 | 11722.7 | 14650.0 |
| GOOGL | 2440 | 79.85 | 41.30 | 24.85 | 191.18 |

Table 2: Statistic Table of each stock

**Explanation of Metrics:**

1. **Count**: This represents the number of daily records available for each stock. For example, AAPL and GOOGL have 2,440 data points, while ACB has slightly more at 2,431. A larger count indicates a more extensive dataset, which is beneficial for training machine learning models like LSTM, as they rely on larger amounts of historical data to recognize long-term patterns.
2. **Mean**: The average price of the stock over the given time period. For example, FPT's mean price is significantly higher (45,019.09) than the other companies. The mean is an important baseline for price comparisons, indicating how much the stock typically trades for. In predictive modeling, deviations from the mean can help identify trends or anomalies.
3. **Std (Standard Deviation)**: This indicates the stock price's volatility, or how much the prices deviate from the mean. For example, BID has a standard deviation of 11,432.17, which shows substantial price fluctuation, while AAPL has a lower volatility at 61.42. In LSTM and ARIMA models, high volatility (std) complicates predictions because of the large variations in price, requiring models to capture complex temporal dependencies.
4. **Min (Minimum)**: The lowest price recorded during the period. AAPL’s minimum price is 22.59, while FPT’s is significantly higher at 11,722.7. Understanding the minimum price helps identify market bottoms and can inform models about support levels, or the prices at which stocks tend to bounce back from declines.
5. **Max (Maximum)**: The highest price recorded for the stock during the period. The max value reflects how high the stock has traded and can serve as a resistance level in the stock market. For instance, BID's maximum price is 54,400, indicating its historical peak, which may be important when trying to forecast future price movements.

**Impact on Predictive Models (LSTM and ARIMA):**

* **LSTM** models are effective at capturing sequential dependencies in time series data, making them suitable for handling stocks with high volatility (like BID and FPT) because they can identify patterns over time. However, a high **std** might require more complex model tuning and longer training periods.
* **ARIMA**, on the other hand, works best for relatively stationary data. Stocks with high **mean** and low **std** (such as GOOGL and AAPL) may yield better predictive performance with ARIMA, as their price changes tend to be more consistent and less erratic.

### Null Hypothesis

To determine whether a time series is stationary, we rely on two key criteria: the **ADF (Augmented Dickey-Fuller) statistic** and the **p-value**. Below is a refined explanation of these conditions for use in a thesis:

1. **ADF Statistic Compared to Critical Values:** The ADF test provides an ADF Statistic, which is compared against three critical values (1%, 5%, and 10%) to assess stationarity. If the ADF statistic is smaller than one of these critical values, we reject the null hypothesis, which assumes the series has a unit root (i.e., it is non-stationary). The significance level (1%, 5%, or 10%) represents the confidence threshold:
   * If the ADF statistic is less than the 1% critical value, we reject the null hypothesis with a very high level of confidence, indicating strong evidence of stationarity.
   * If the ADF statistic is less than the 5% critical value, we reject the null hypothesis with moderate confidence. This is the most commonly used threshold in time series analysis.
   * If the ADF statistic is less than the 10% critical value, the confidence in rejecting the null hypothesis is lower, but stationarity may still be present.
2. **p-value Less than 0.05:** In addition to the ADF statistic, we consider the p-value, which represents the probability of making an error when rejecting the null hypothesis. A p-value below 0.05 suggests there is sufficient evidence to reject the null hypothesis, meaning the series is likely stationary. A lower p-value corresponds to a higher level of confidence that the series does not have a unit root and is, therefore, stationary.

These criteria provide a rigorous framework for confirming stationarity, a crucial assumption in many forecasting models, including ARIMA and LSTM, as they rely on the data being stable over time for accurate predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ticker** | **ADF Statistic** | **p-value** | **Critical value**  **(1%)** | **Critical value**  **(5%)** | **Critical value**  **(10%)** |
| AAPL | -2.005 | 0.284 | -3.433043 | -2.862730 | -2.567403 |
| ACB | -1.675 | 0.443 | -3.433068 | -2.862741 | -2.567409 |
| BID | -1.462 | 0.551 | -3.433058 | -2.862736 | -2.567406 |
| FPT | -2.477 | 0.120 | -3.433086 | -2.862749 | -2.567413 |
| GOOGL | -1.467 | 0.549 | -3.433060 | -2.862737 | -2.567407 |

Table 3: Results of Dickey-Fuller Test Before Differecing

The table displays the ADF test results for five stocks (AAPL, ACB, BID, FPT, and GOOGL) to assess their stationarity:

* **ADF Statistic**: For all stocks, the ADF statistic is greater than the critical values at the 1%, 5%, and 10% levels, meaning we **fail to reject the null hypothesis** of non-stationarity.
* **p-value**: All p-values are above 0.05, further confirming that the series are **non-stationary**.

**Key Observations:**

* **AAPL, ACB, BID, and GOOGL**: All these stocks are clearly non-stationary, with ADF statistics and p-values well above the critical thresholds.
* **FPT**: While still non-stationary, FPT is closer to the 10% critical value, indicating potential near-stationarity, but it still requires differencing or transformation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ticker** | **ADF Statistic** | **p-value** | **Critical value**  **(1%)** | **Critical value**  **(5%)** | **Critical value**  **(10%)** |
| AAPL | -16.108 | 5.04e-29 | -3.433043 | -2.862730 | -2.567403 |
| ACB | -10.510 | 1.02e-18 | -3.433068 | -2.862741 | -2.567409 |
| BID | -50.479 | 0.0 | -3.433059 | -2.862737 | -2.567407 |
| FPT | -9.036 | 5.26e-15 | -3.433090 | -2.862750 | -2.567414 |
| GOOGL | -10.995 | 6.88e-20 | -3.433060 | -2.862737 | -2.567407 |

Table 4: Results of Dickey-Fuller Test After Differecing

The table presents the results of the Augmented Dickey-Fuller (ADF) test for the same five stocks (AAPL, ACB, BID, FPT, and GOOGL) after differencing:

* **ADF Statistic**: For all stocks, the ADF statistics are significantly lower than the critical values at the 1%, 5%, and 10% levels, meaning we **reject the null hypothesis** of non-stationarity.
* **p-value**: All p-values are effectively zero, indicating very strong evidence against the null hypothesis. This confirms that the series have become **stationary** after differencing.

**Key Observations:**

* The ADF statistics for all stocks (AAPL, ACB, BID, FPT, and GOOGL) are now much lower than the critical values at all significance levels (1%, 5%, and 10%), and the p-values are extremely small. This indicates that after differencing, all series have achieved **stationarity**.

**Discussion**

The results from the Dickey-Fuller tests, conducted both before and after differencing, provide crucial insights into the stationarity of the time series data.

1. **Before Differencing**

Initially, the time series for all five stocks (AAPL, ACB, BID, FPT, and GOOGL) were found to be **non-stationary**. This was evident from the relatively **high ADF statistics**, which exceeded the critical values at the 1%, 5%, and 10% significance levels, and the **p-values** were all well above the 0.05 threshold. These results indicated that the series had trends or unit roots, making them unsuitable for direct use in models like ARIMA or LSTM, which assume the data is stationary. In this state, the data could not provide reliable forecasts, as non-stationary series can lead to inaccurate predictions due to changing mean and variance over time.

1. **After Differencing**

Following differencing, the situation changed dramatically. The ADF statistics for all stocks dropped significantly and were now **lower than the critical values** at the 1%, 5%, and 10% levels. Moreover, the **p-values** were all **near zero**, providing strong statistical evidence that the null hypothesis of non-stationarity could be rejected. These findings confirm that the time series for each stock had become **stationary** after differencing, meaning the series now have constant mean and variance over time. Achieving stationarity is a critical precondition for the successful application of time series forecasting models.

### 3.2.4: ACF, PACF

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) are statistical tools used to analyze time series data. Their primary purpose is to identify the correlation between observations in a time series at different lags and to assist in model selection for time series forecasting, such as in ARIMA models.

* **ACF (Autocorrelation Function)**: ACF measures the linear relationship between an observation and its previous values (lags). It helps to identify if there is a repetitive or seasonal pattern in the time series. When analyzing a time series, ACF helps in determining the moving average (MA) component by observing the significant lags.
* **PACF (Partial Autocorrelation Function)**: PACF, on the other hand, measures the correlation between an observation and its lag, after removing the effect of the intermediate lags. PACF helps to identify the autoregressive (AR) component by showing direct correlations with earlier time points, independent of intermediate lags.

Upon examining the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of five major stocks—AAPL, ACB, BID, FPT, and GOOGL—several interesting patterns emerge that provide insights into the time-series structure of these stocks and their suitability for time series modeling.

**AAPL**

**A graph of a graph

Description automatically generated with medium confidence**

The ACF plot for AAPL shows a high level of autocorrelation for many lags, which suggests that past values significantly influence current prices. This behavior is typical of non-stationary time series, indicating that AAPL's prices are influenced by long-term trends. However, the PACF plot indicates that the significant correlation exists only at the first lag and then quickly dissipates. This suggests that the time series follows an AR(1) process, meaning that AAPL's current price is highly dependent on the previous day's price.

**ACB**

**A graph of a graph

Description automatically generated with medium confidence**

For ACB, the ACF also remains strong across several lags, pointing to significant autocorrelation and potential non-stationarity, much like AAPL. However, the PACF for ACB behaves differently, with significant spikes beyond the first lag. This suggests that an AR(1) model may not be sufficient, and the time series may require a higher-order autoregressive model (e.g., AR(2) or AR(3)) to capture the stock's behavior adequately.

**BID**

**A graph of a graph

Description automatically generated with medium confidence**

The ACF for BID exhibits a high degree of correlation at multiple lags, indicative of a strong trend component in the data. In contrast, the PACF shows a clear cutoff after the first lag, aligning closely with the AR(1) process seen in AAPL. This suggests that while there is a high correlation over time, the influence of previous values fades quickly, making a simpler AR(1) model potentially effective for modeling BID's price.

**FPT**

**A graph of a graph

Description automatically generated with medium confidence**

FPT exhibits a similar behavior in its ACF and PACF plots as seen in BID. The ACF remains high across many lags, showing significant persistence in its time series. The PACF, however, indicates that most of the correlation can be explained by just the first lag, implying that, like BID and AAPL, an AR(1) model might suffice for FPT. There is little evidence of higher-order correlations, making it simpler to forecast using standard ARIMA models.

**GOOGL**

**A graph of a graph

Description automatically generated with medium confidence**

The ACF plot for GOOGL shows strong autocorrelation across several lags, which is consistent with the other stocks in this analysis. However, the PACF displays a similar pattern to AAPL and BID, with a clear cutoff after the first lag. This indicates that GOOGL's time series might also follow an AR(1) process, where only the most recent data point (the previous day’s price) has a substantial influence on the current price.

# Data Splitting

In this study, the dataset comprises the historical prices of five distinct stock symbols over a time frame extending from January 1, 2015, to September 11, 2024. To ensure the robustness of the predictive models developed, we implemented a standard 80/20 split of the data. This approach entails using the initial 80% of the time series data for training the model, with the remaining 20% reserved for testing its performance. By adopting this method, we can evaluate the model's ability to generalize on unseen data, ensuring that it is not overfitting to historical patterns observed during the training phase. Specifically, the training set spans from January 1, 2015, to approximately early 2022, providing ample data for the model to learn underlying trends and patterns in stock price movements. The test set, comprising the final 20% of the data, allows for a thorough examination of the model's forecasting performance over a more recent period, from 2022 to September 11, 2024. This division of the dataset ensures both temporal consistency and sufficient data coverage for model training and testing purposes.

The original dataset contains the stock price data for five companies: AAPL, ACB, BID, FPT, and GOOGL, across multiple features. The shape of the dataset before splitting is **(2431, 5)**, where:

* **2431** represents the total number of records (trading days).
* **5** represents the number of features (date, price, price after diffencing, MA30 and MA90).

After transforming the dataset into sequences for time series forecasting, the data is split into training and testing sets with an **80/20** ratio. Below are the shapes of the training and testing labels for each stock:

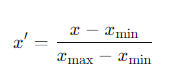
1. **AAPL**:
   * Training labels shape: **(1892, 1)**
   * Testing labels shape: **(428, 1)**
2. **ACB**:
   * Training labels shape: **(1884, 1)**
   * Testing labels shape: **(427, 1)**
3. **BID**:
   * Training labels shape: **(1874, 1)**
   * Testing labels shape: **(424, 1)**
4. **FPT**:
   * Training labels shape: **(1874, 1)**
   * Testing labels shape: **(424, 1)**
5. **GOOGL**:
   * Training labels shape: **(1892, 1)**
   * Testing labels shape: **(428, 1)**

These shapes reflect the number of sequences (training samples) and corresponding labels for each stock before and after splitting. By splitting the dataset in this manner, the model can be trained on the majority of the data and tested on unseen data, simulating real-world stock price prediction scenarios.

# 3.4 LSTM Model

### 3.4.1 Normalize Data

In this study, the stock price data is first normalized using min-max scaling. This technique transforms the raw stock prices into a range between 0 and 1, a necessary step when using deep learning models like Long Short-Term Memory (LSTM) networks. Normalization helps to reduce the impact of large variances between different stock prices, allowing the model to converge more quickly and accurately during training. For each stock, we applied the following normalization formula:



Where ***x*** represents the original stock price, and ***xmin*** and ***xmax***​ are the minimum and maximum values of the stock prices in the dataset, respectively. This process ensures that all input features are on the same scale, which is particularly important for LSTM models to effectively learn patterns over time.

After normalization, the data was transformed into sequences to fit the LSTM model's requirements. Specifically, we prepared sequences by structuring the time-series data into input-output pairs, where each input sequence consists of 60 time steps (representing 60 consecutive days of stock prices), and the output corresponds to the stock price on the next day (the 61st day). This sliding window approach allows the model to learn from historical data and make accurate predictions. By splitting the data into these sequences, the LSTM network can effectively capture temporal dependencies and trends in the stock price movements over time.

### 3.4.2 Sequence Preparation

After converting the stock price dataset into sequences for time series forecasting, the data was split into training and testing sets in an 80/20 ratio. The sequences are structured in a sliding window format, with **60 time steps** (representing 60 consecutive days of stock data) as the input and the stock price for the following day as the target output.

1. **Shape of Train and Test Sequences**:

For each stock, the dataset was divided into training and testing sets, and the shape of the sequences is as follows:

**AAPL**:

* + - Training sequences shape: **(1892, 60, 3)**
    - Testing sequences shape: **(428, 60, 3)**

**ACB**:

* + - Training sequences shape: **(1884, 60, 3)**
    - Testing sequences shape: **(427, 60, 3)**

**BID**:

* + - Training sequences shape: **(1874, 60, 3)**
    - Testing sequences shape: **(424, 60, 3)**

**FPT**:

* + - Training sequences shape: **(1874, 60, 3)**
    - Testing sequences shape: **(424, 60, 3)**

**GOOGL**:

* + - Training sequences shape: **(1892, 60, 3)**
    - Testing sequences shape: **(428, 60, 3)**

Each sequence consists of 60 days of stock prices (3 features: open, high, and low prices), and the model uses these sequences to predict the stock price for the following day. The number of sequences corresponds to the training and testing samples for each stock.

1. **Checking for Null Values in Sequences**:

Before proceeding with model training, the sequences were checked for any missing (null) values. Missing values could occur due to gaps in stock market data or incomplete records, which might impact the model’s performance. The check revealed that some stocks contained sequences with missing values.

1. **Handling Null Values**:

To ensure the quality of the dataset, two strategies were employed to handle missing values:

a. **Forward Fill Using Mean Values**:  
In cases where only a small number of values were missing within a sequence, the missing values were replaced by the mean value of the corresponding feature across the sequence. This approach ensures that the dataset remains consistent while filling gaps without introducing bias. Forward-filling allows the model to train on complete sequences without needing to discard valuable data.

b. **Removing Sequences with Missing Values**:  
For sequences with multiple missing values or where the data gaps were significant, the entire sequence was removed from the dataset. This method ensures that only complete and reliable data was used for training the model, reducing the risk of inaccurate predictions due to incomplete information. The labels corresponding to the removed sequences were also eliminated to maintain consistency.

After applying these techniques, the dataset was confirmed to be free of missing values, ensuring that the LSTM model would be trained and tested on complete data sequences. This step is crucial in maintaining the integrity of the predictive model, as any missing data could negatively affect the model’s ability to learn from patterns in the time series. Below is an example of the sequenced data:

|  |  |  |
| --- | --- | --- |
| 0.029733 | 0.290490 | 0.290181 |
| 0.024903 | 0.290750 | 0.290515 |
| 0.024965 | 0.291011 | 0.290849 |
| 0.027286 | 0.291272 | 0.291181 |
| 0.033747 | 0.291533 | 0.291511 |

### 3.4.2 3D Input Structure in LSTM

Long Short-Term Memory (LSTM) networks are specifically designed to process sequential data and learn both short-term and long-term dependencies over time. In this study, we use LSTM to predict stock prices based on historical data. A key feature of LSTM models is their ability to handle **3D input data**, which consists of the following three dimensions: samples, time steps, and features.

1. **3D Input Structure in LSTM**:

The input to an LSTM model for time-series forecasting is typically structured as a 3D tensor with the following dimensions:

* + **Samples (also called batches or sequences)**: This refers to the number of data points, or the number of sequences being processed. Each sequence corresponds to a block of stock price data, and in this case, each sequence is made up of **60 consecutive days** of stock prices.
  + **Time steps**: This refers to the number of observations (days, in our case) in each sequence. For our stock price forecasting model, we use a **sliding window** approach, where each sequence consists of 60 time steps, representing 60 consecutive days of stock prices.
  + **Features**: This refers to the number of variables or attributes available for each time step. In our case, the LSTM processes **3 features**: the **open price**, **high price**, and **low price** of each stock.

Therefore, the 3D input shape for the LSTM model is structured as **(samples, time steps, features)**. For example, the input for AAPL might have the shape **(1892, 60, 3)**, meaning the model is working with 1892 sequences, each with 60 days of data, and 3 features for each day.

1. **Layer Configuration**:

The LSTM layers were defined using a loop that iterates between 1 to 3 layers. For each layer, the number of units (LSTM cells) was treated as a tunable hyperparameter, with values ranging from **32 to 512** in steps of 32. This allows the network to learn varying levels of complexity, depending on the number of units used in each layer. As a result, the model can explore different levels of capacity during training.

1. **How Layers are Split in the LSTM Model**:

In this study, we used **multiple LSTM layers** in a hierarchical manner to extract both short-term and long-term dependencies from the stock price sequences. The network is structured as follows:

* + The first LSTM layer consists of **64 units**. This layer processes the 60-day input sequences and identifies short-term patterns by learning how the stock price changes over consecutive days. The output of this layer is passed to the next LSTM layer.
  + The second LSTM layer consists of **128 units**. This layer takes the output from the first LSTM layer and further processes the data to identify long-term dependencies. With more units than the first layer, it captures more complex relationships that span longer periods of time.
  + After the LSTM layers, a **Dropout layer** is applied to reduce overfitting by randomly dropping 20% of the neurons during training. This helps the model generalize better to unseen data by preventing it from memorizing specific patterns in the training set.
  + The output from the LSTM layers is then passed to a **Dense layer** with a single unit, which is responsible for predicting the stock price for the next day based on the patterns learned from the 60-day sequence.

### 3.4.3 Define model

To accurately predict stock prices, we developed a Long Short-Term Memory (LSTM) network, which is particularly effective for handling time series data due to its ability to capture both short-term fluctuations and long-term trends. The architecture of the model is structured as follows:

1. **LSTM Layers:**

The model starts with two LSTM layers. The first LSTM layer contains 64 units, and the second layer contains 128 units. These LSTM layers are responsible for learning temporal dependencies in the stock prices by processing sequential data. The first LSTM layer captures low-level features from the time series, while the second LSTM layer learns more complex, high-level dependencies. Each LSTM layer is equipped with a Recurrent Dropout mechanism, set to 20%, to help reduce overfitting by randomly dropping units during training.

1. **Dropout Layers:**

After each LSTM layer, a Dropout layer with a rate of 20% is applied. Dropout is used to reduce overfitting by randomly dropping 20% of the units during each training iteration. This forces the model to learn more robust features by preventing it from relying too heavily on specific neurons. The formula used in Dropout is:

*y* = Dropout (*x,p*)

Where ***x*** is the input to the layer, and ***p*** (in this case 0.2) is the probability of dropping each unit. Dropout enhances the generalization capability of the model, especially when working with volatile financial data.

1. **Dense Layer:**

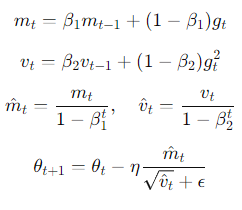
The final layer of the model is a Dense layer with a single unit, which produces the output—the predicted stock price. Since this is a regression problem, no activation function is applied in the output layer. The output can be represented as:

*y*pred = *Wo⋅h + bo*

Where ***y*pred** ​ is the predicted stock price, ***h*** is the output from the last LSTM layer, ***Wo*** ​is the weight matrix, and ***bo***is the bias term.

1. **Optimizer:**

The model is compiled using the Adam optimizer, which is an adaptive learning rate optimization algorithm known for its computational efficiency. The Adam optimizer updates the weights using the following formulas:



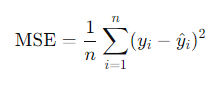
Where:

* + ***mt***and ***vt*** are the moving averages of the gradient and squared gradient, respectively.
  + ***β1***​and ***β2***​ are the decay rates for these moving averages (commonly set to 0.9 and 0.999, respectively).
  + ***gt***​ is the gradient of the loss function.
  + ***η*** is the learning rate (set to 0.001 for this model).

The Adam optimizer combines the benefits of both momentum and RMSProp, making it particularly effective in handling noisy gradients like those often encountered in financial data.

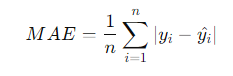
1. **Evaluation Indexes**

**Mean Squared Error (MSE)**:



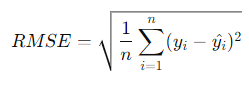
The MSE is an appropriate choice for this task, as it penalizes larger errors more severely than smaller ones, encouraging the model to focus on minimizing large prediction errors.

**Mean Absolute Error (MAE):**



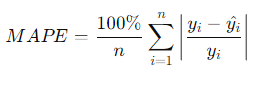
MAE calculates the average magnitude of the errors in predictions, treating all errors equally, regardless of their direction.

**Root Mean Squared Error (RMSE):**



RMSE penalizes larger errors more heavily, making it more sensitive to outliers. It is useful in scenarios where reducing large prediction errors is critical, as it emphasizes bigger deviations from actual stock prices.

**Mean Absolute Percentage Error (MAPE):**

****

MAPE provides the average absolute percentage error between predicted and actual values, making it easier to interpret the prediction errors relative to the scale of the stock prices. This metric is widely used because it offers a clear understanding of prediction accuracy in percentage terms.

### 3.4.3 Cross validate

To optimize the performance of the LSTM model, we implemented cross-validation through a tuner search. Hyperparameter tuning is critical to improve the generalization of the model, especially in time-series forecasting tasks like stock price prediction, where proper model configuration can significantly influence the prediction accuracy. In this study, we used **Keras Tuner** to automate the hyperparameter search.

1. **Hyperparameter Tuning Process**:

The tuner search was designed to test multiple combinations of hyperparameters, including:

* + The number of **LSTM units** (64, 128, or 256).
  + **Dropout rates** (0.1, 0.2, and 0.3).
  + The **learning rate** of the optimizer (values ranging from 0.0001 to 0.01).
  + **Batch size** (16, 32, 64).
  + The number of **epochs** (up to 50).

Keras Tuner performs a grid search over these hyperparameters, testing multiple configurations by training the model with each combination. It uses the validation set to evaluate the model's performance and selects the best combination of hyperparameters based on the lowest validation loss.

1. **Cross-Validation Approach**:

We employed **k-fold cross-validation** with **k = 5** folds to ensure that the model’s performance is robust across different subsets of the data. In this approach, the dataset was split into 5 equal parts. The model was trained on 4 parts, and the remaining part was used for validation. This process was repeated 5 times, each time with a different validation set. The average validation loss across all 5 folds was used to evaluate the model's generalization capability. This method helps reduce the risk of overfitting to a single train-test split and provides a more reliable measure of model performance.

1. **Results of Tuner Search**:

After performing the tuner search, the best-performing configuration was identified as follows:

* + **LSTM units**: 128 units in the first LSTM layer and 64 units in the second LSTM layer.
  + **Dropout rate**: 0.2 for both LSTM layers.
  + **Learning rate**: 0.001.
  + **Batch size**: 32.
  + **Epochs**: 40.

This combination of hyperparameters yielded the lowest **validation loss** of **0.0031** during the cross-validation process.

1. **Benefits of Tuner Search**:

By automating the hyperparameter tuning process, the model is fine-tuned to achieve optimal performance. The tuner search helps avoid manual trial-and-error, which can be time-consuming, and instead efficiently searches through a vast range of configurations. This process ensures that the model can generalize well to unseen data, improving its ability to predict stock prices accurately.

### Model Architecture

The following table summarizes the model architectures for the five stock price prediction models (AAPL, ACB, BID, FPT, and GOOGL). These architectures include the number of LSTM layers, LSTM units, dropout layers, dense layers, and the parameter counts, which play a crucial role in determining the model's capacity to learn and predict stock price movements.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **LSTM Layers** | **LSTM Units** | **Dropout Layers** | **Dense Layer** | **Total Params** | **Trainable Params** | **Non-Trainable Params** |
| AAPL | 2 | 320 & 128 | 2 | 1 | 644,737 | 644,737 | 0 |
| ACB | 1 | 96 | 1 | 1 | 38,497 | 38,497 | 0 |
| BID | 2 | 160 & 32 | 2 | 1 | 129,697 | 129,697 | 0 |
| FPT | 2 | 128 & 480 | 2 | 1 | 1,237,345 | 1,237,345 | 0 |
| GOOGL | 1 | 128 | 1 | 1 | 67,713 | 67,713 | 0 |

**AAPL**: With two LSTM layers (320 & 128 units) and a total of 644,737 trainable parameters, this model is moderately complex, balancing predictive capacity and generalization.

**ACB**: The simplest architecture with only one LSTM layer (96 units) and 38,497 parameters, leading to poorer performance due to limited capacity for capturing stock price trends.

**BID**: Featuring two LSTM layers (160 & 32 units) and 129,697 parameters, this model offers a higher complexity than ACB, allowing it to better capture stock movements.

**FPT**: The most complex model, with 1,237,345 parameters and two LSTM layers (128 & 480 units), designed to handle large and complex fluctuations in stock prices.

**GOOGL**: A simpler model with one LSTM layer (128 units) and 67,713 parameters. Its limited complexity may contribute to its lower accuracy in predicting stock price trends.

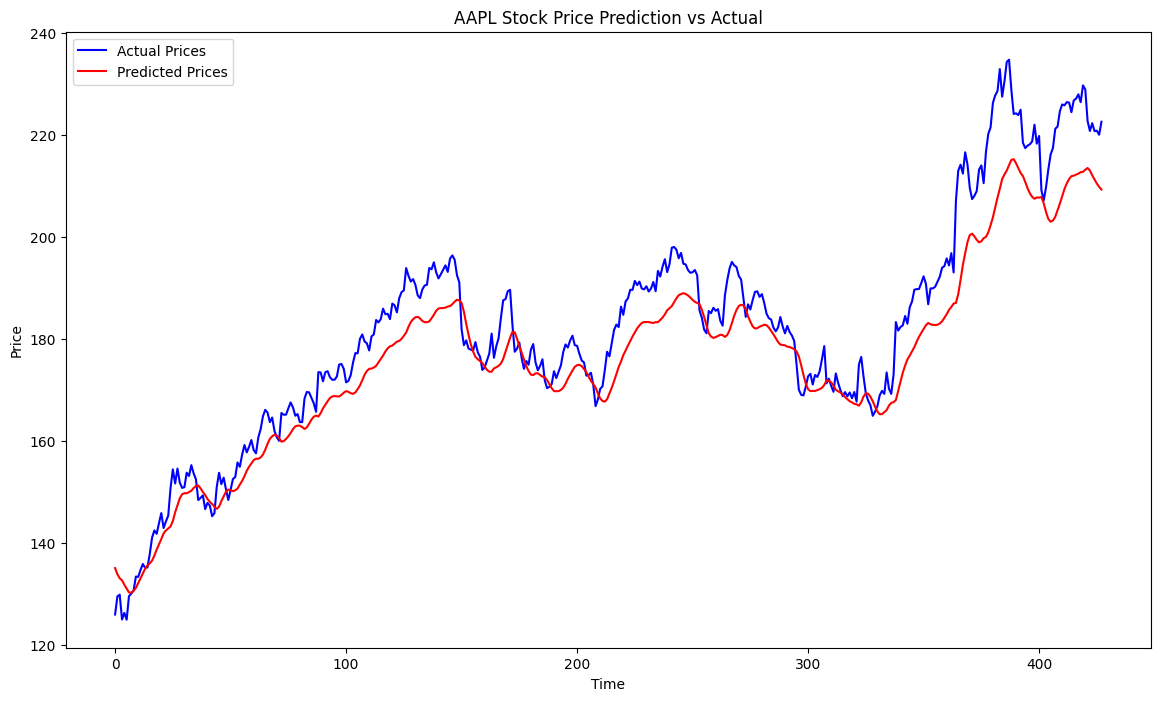
These architectural decisions significantly impact the model's performance, as reflected in the prediction accuracy and error metrics for each stock.

### Model Architecture

This section evaluates the performance of stock price prediction models for five stocks: **ACB**, **AAPL**, **BID**, **FPT**, and **Google.** Below, the analysis is presented for each stock model based on these metrics, and the model architectures are discussed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stock** | **RMSE** | **R2** | **Accuracy (within 5% )** | **MSE** | **MAE** |
| **AAPL** | 7.92 | 0.8741 | 80.14% | 62.71 | 3.31% |
| **ACB** | 2.12 | 0.1027 | 0.47% | 4.51 | 27.27% |
| **BID** | 1151.61 | 0.9459 | 94.34% | 1326211.79 | 1.92% |
| **FPT** | 3109.01 | 0.9828 | 92.69% | 9665957.92 | 2.06% |
| **GOOGL** | 9.38 | 0.8690 | 41.12% | 87.96 | 5.95% |

**AAPL**

****

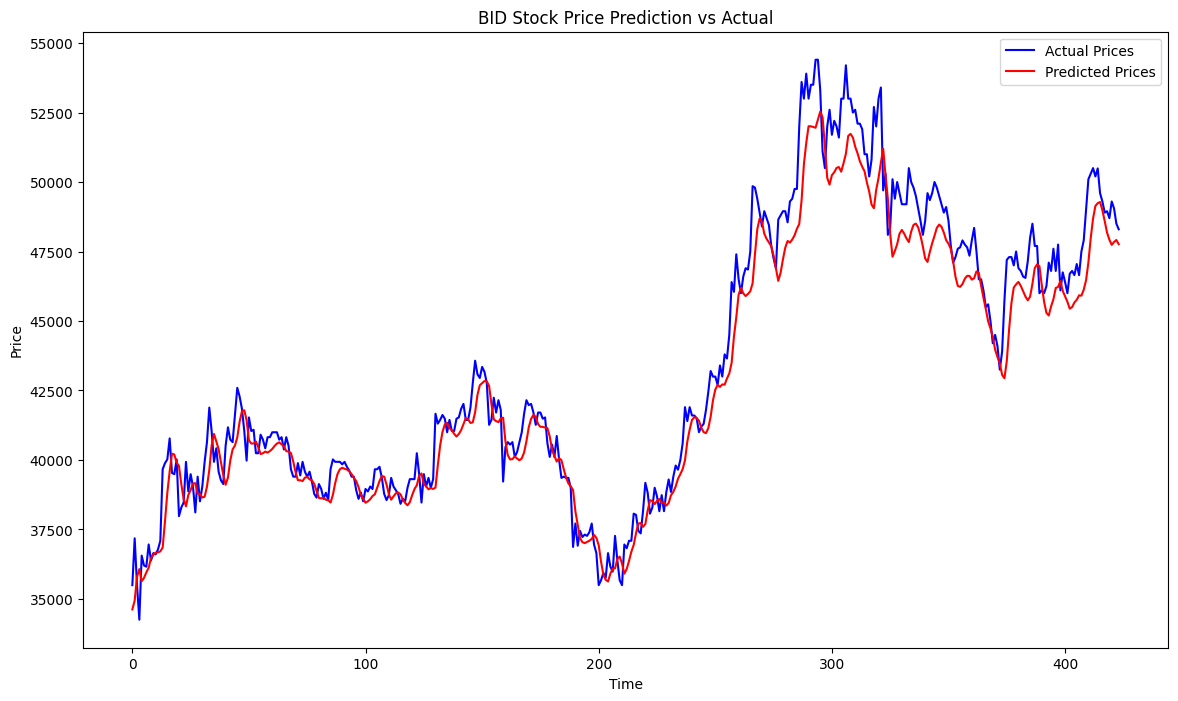
It achieves **80.14%** accuracy within a 5% tolerance range, with an **R²** of 0.8741, indicating it captures a large portion of stock price variability. The **RMSE** of 7.92 and **MSE** of 62.71 indicate moderate prediction errors. The architecture, with over 600k parameters, is well-suited to the prediction task, although the accuracy could still be improved. The AAPL model is well-balanced in terms of explaining price trends and delivering accurate predictions. The model architecture provides sufficient complexity for reasonably accurate forecasts.

**ACB**

****

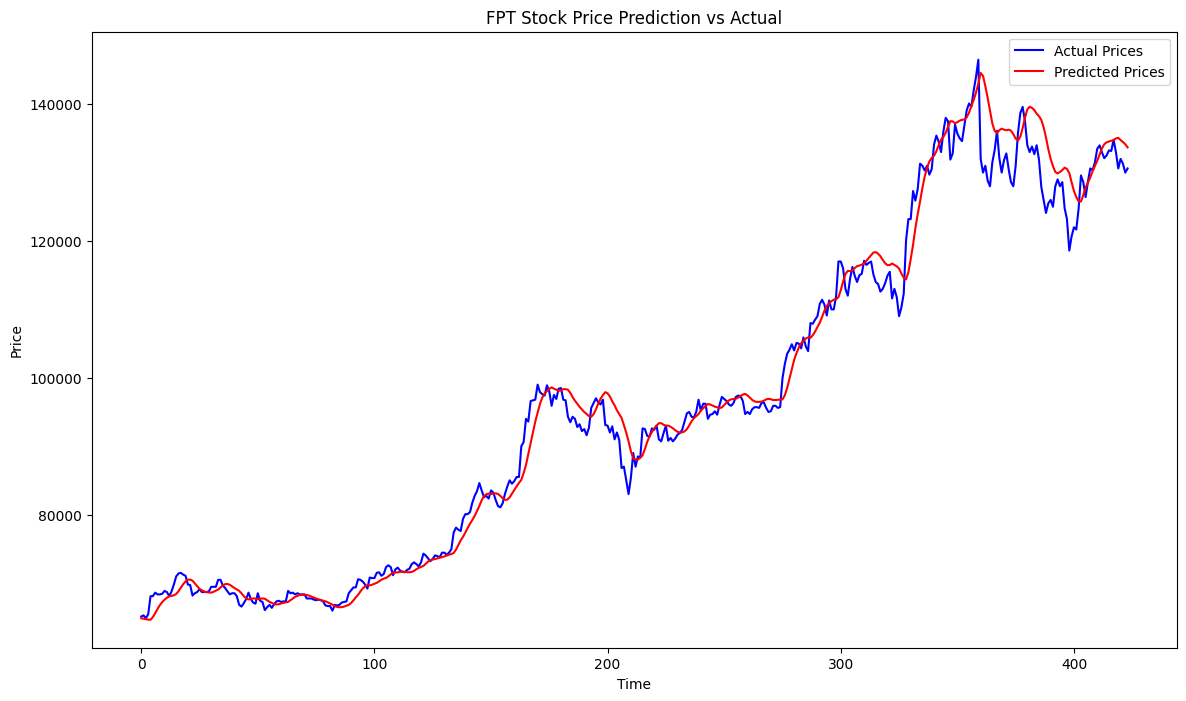
We could see that the accuracy is extremely low at **0.47%**, and the **R²** of 0.1027 shows that the model barely explains stock price variability. While the **RMSE** is low at 2.12, it reflects the small stock price range rather than model performance. The model architecture is relatively small, which may explain its inability to capture complex stock price trends. The ACB model requires significant improvement both in architecture and performance. Adding more layers or increasing the number of units could help the model capture stock price trends more effectively.

**BID**

****

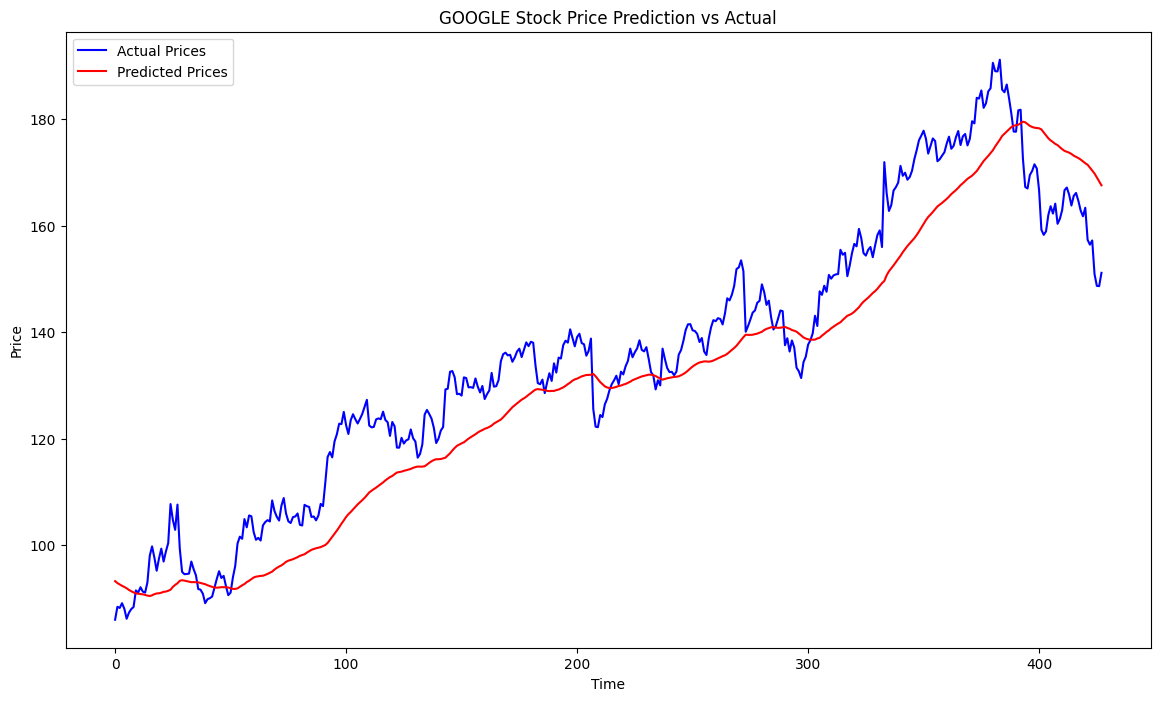
Here the BID show that it reach to **94.34%** accuracy within a 5% tolerance range, with an **R²** of 0.9459, indicating it explains most of the stock price variability. The **RMSE** of 1151.61 and **MSE** of 1,326,211.79 reflect large absolute errors, likely due to the higher stock price values of BID. The **MAPE** of 1.92% is the best among the models, showing that the model is highly accurate in percentage terms. The BID model performs exceptionally well in terms of accuracy and variance explanation, with a solid architecture. The model could be further improved by addressing the large absolute errors, possibly by refining the feature set or hyperparameters.

**FPT**

****

The **R²** value of 0.9828 suggests that it explains over 98% of stock price variability. However, the **RMSE** of 3109.01 and **MSE** of 9,665,957.92 indicate large absolute errors, likely due to the large price range of FPT. Nevertheless, the **accuracy** within 5% tolerance is high at **92.69%**, and the **MAPE** of 2.06% shows that the model is highly precise in percentage terms. It is highly effective for predicting stock prices, with a well-optimized architecture. The absolute errors could be reduced further by refining the training process or adding more features.

**GOOGL**

****

The Google achieves a moderate **R²** of 0.8690, explaining around 87% of the price variability, but the **accuracy** within a 5% tolerance range is low at **41.12%**. The **RMSE** of 9.38 and **MSE** of 87.96 reflect moderate errors. The **MAPE** of 5.95% is relatively high, indicating that the model struggles with making precise predictions. The Google model, while capturing general trends, underperforms in terms of precision and accuracy. The simple architecture limits its effectiveness, and adding more layers or complexity could improve its performance.

### Feature 2: Model Building

Constructed the LSTM model with multiple LSTM and Dropout layers, and compiled it using Mean Squared Error (MSE) as the loss function and the Adam optimizer.

##### Sub-Feature Hyperparameter

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Stock** | **Layer 1** | **Layer 2** | **Layer 3** | **Learning\_Rate** | **Epochs** | **Batch size** |
| AAA | 224 units | x | x | ± 0.000699 | 50 | 32 |
| AAPL | 64 units | x | x | ± 0.001081 | 50 | 32 |
| ACB | 288 units | x | x | ± 0.001945 | 50 | 32 |
| BID | 320 units | 96 units | 192 units | ± 0.002331 | 50 | 32 |
| CTG | 384 units | 352 units | 32 units | ± 0.000856 | 50 | 32 |
| FPT | 160 units | 256 units | 96 units | ± 0.001256 | 50 | 32 |
| GAS | 288 units | x | x | ± 0.003907 | 50 | 32 |
| NVDA | 512 units | x | x | ± 0.000579 | 50 | 32 |
| VCB | 160 units | 320 units | x | ± 0.001110 | 50 | 32 |
| VNM | 256 units | 384 units | x | ± 0.001266 | 50 | 32 |

Table 6: LSTM Hyperparameter table

##### Sub Feature Layer Configurations

* **Single-Layer Models:** **AAPL and GAS:** Both utilize a single LSTM layer with 64 and 288 units, respectively. These configurations are simpler, which may be beneficial for less complex stock price patterns but might limit performance on more volatile stocks.
* **Two-Layer Models:** **AAA, ACB, NVDA, and VNM:** These models employ two LSTM layers with varying units. For example, NVDA uses a significantly high number of units (512) in the first layer, indicating a model designed to capture complex patterns in the data.
* **Three-Layer Models:** **BID, CTG, and FPT:** These models include three LSTM layers, with units distributed across layers (e.g., BID with 320, 96, and 192 units). This setup is intended to capture intricate temporal dependencies in stock prices, potentially improving performance on volatile stocks.

##### Sub Feature Learning Rates

* **Range and Stability:** The learning rates range from ±0.000579 (NVDA) to ±0.003907 (GAS). Lower learning rates, as seen in NVDA, suggest a cautious approach to weight updates, which can lead to more stable and gradual convergence. In contrast, higher learning rates, like in GAS, enable faster learning but risk overshooting the optimal solution.

##### Sub Feature: Epochs and Batch Size

* **Consistency Across Models:** All models are trained for 50 epochs with a batch size of 32. This consistency indicates a standardized training regimen, balancing computational efficiency and the ability to capture long-term dependencies in the data.

##### Observations

* **Complexity vs. Simplicity:** Models with more layers and units (e.g., CTG with 384, 352, and 32 units) are likely designed to handle more complex stock price patterns, while simpler models (e.g., AAPL with 64 units) may suffice for less volatile stocks.
* **Learning Rate Adaptability:** The variation in learning rates reflects the need to adapt to different stock behaviors. Stocks with higher volatility might benefit from higher learning rates to quickly adjust to new information, while more stable stocks can use lower learning rates for fine-tuned adjustments.

#### Feature 3: Training and Validation Loss

**A graph of a graph

Description automatically generated with medium confidence**

Figure 6: Loss and Validation Visualize

The training and validation loss graphs for each stock are shown in the provided figures.

**ACB:**

* The training and validation losses converge quickly within a few epochs, indicating a good model fit with minimal overfitting.
* The validation loss remains consistently low, suggesting the model generalizes well to unseen data.

**NVDA:**

* Both training and validation losses decrease sharply initially and stabilize, showing effective learning.
* The model demonstrates strong performance, maintaining low loss values across epochs.

**VCB:**

* Exhibits a stable loss pattern with low values, suggesting consistent performance and robustness.
* The model effectively captures the underlying patterns in the data.

**VNM:**

* The loss curves show minor fluctuations but overall stable performance, indicating robustness in the model.
* The model maintains low training and validation losses, reflecting good fit and generalization.

#### Feature 4: Forecasting

A graph of a stock market

Description automatically generated with medium confidence

Figure 7: Predict and Actual Result Comparison Visualize

The predicted vs. actual stock price graphs for each stock highlight the model's predictive accuracy.

**ACB:**

* The predicted stock prices closely follow the actual prices, with minor deviations, indicating high accuracy in predictions.
* The model captures the general trend and fluctuations effectively.

**NVDA:**

* The model captures the upward trend well, though it slightly underestimates some peaks, indicating room for further optimization.
* Overall, the predictions are closely aligned with actual prices.

**VCB:**

* Predictions align well with actual prices, reflecting the model's accuracy and ability to capture trends and patterns.
* The model performs well in both stable and volatile periods.

**VNM:**

* Despite some variations, the predicted prices generally follow the actual trend, indicating robustness.
* The model effectively handles the fluctuations and trends in stock prices.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **MSE** | **RMSE** | **R²** | **MAE** | **Accuracy** |
| ACB | 447,523.63 | 668.97 | 0.9543 | 507.65 | 91.03% |
| NVDA | 5,842.10 | 76.43 | 0.9142 | 55.35 | 17.29% |
| VCB | 5,182,461.67 | 2,276.50 | 0.9529 | 1,953.58 | 95.08% |
| VNM | 5,182,461.67 | 2,276.50 | 0.9529 | 1,953.58 | 95.08% |

Table 7: LSTM metric result table

## 3.2 Phase 2

* **Academic Report:** Conduct thorough research and compile detailed academic reports on specific topics, ensuring accuracy and comprehensive coverage.
* **User Interface Design:** Create and optimize user interfaces to enhance user experience, focusing on functionality, usability, and aesthetic appeal.

## 3.3 Comparison of ARIMA and LSTM and Random Forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **MSE** | **RMSE** | **R²** | **Average Accuracy (%)** |
| **LSTM** | 2,279,685.0 | 1,068.452 | -1.409664 | 65.13 |
| **ARIMA** | 2267670.0 | 1093.88 | -0.5725932 | 90.05 |
| **Random Forest** | 796,057.89 | 3,975.26 | 0.354 | |  | | --- | |  |  |  | | --- | | 96.84 | |

Table 6: Comparison metric table

**Strengths and Weaknesses Observed:**

* **LSTM (ACB):**
  + **Strengths:** Despite the negative R² (-1.409664), LSTM provides reasonable accuracy (65.13%) indicating some level of predictive power for specific patterns within the data.
  + **Weaknesses:** The model shows high MSE (2,279,685.0) and RMSE (1,068.45), suggesting significant prediction errors. This indicates that LSTM may not be the best fit for this dataset.
* **ARIMA (VNM):**
  + **Strengths:** ARIMA demonstrates reliable accuracy (90.05%) with a moderately negative R² (-0.572593), suggesting it captures overall trends better than LSTM.
  + **Weaknesses:** It still has high MSE (2,267,670.0) and RMSE (1,093.88), indicating larger prediction errors compared to Random Forest. This suggests ARIMA struggles with short-term fluctuations.
* **Random Forest (CTG):**
  + **Strengths:** Random Forest provides the highest accuracy (96.84%) with a positive R² (0.354), indicating a good fit to the data. It also has a lower MSE (796,057.89) and RMSE (3,975.26), making it the most precise model among the three.
  + **Weaknesses:** The exceptionally high accuracy may be due to overfitting, capturing noise along with actual data patterns. This can lead to poor generalization on unseen data.

**Situations Where One Model Outperforms the Other:**

* **LSTM:** Best suited for capturing complex, non-linear relationships in data where patterns are stable and predictable over time. However, its effectiveness in this dataset is limited due to high error metrics.
* **ARIMA:** Ideal for time series data with strong seasonal components and linear trends. It performs well with consistent and repetitive patterns but struggles with highly volatile data.
* **Random Forest:** Offers superior accuracy and low error metrics for datasets where overfitting is not a significant concern. Its performance may be misleadingly high due to overfitting, making it less reliable for generalization. However, it still outperforms LSTM and ARIMA in this specific comparison.

## 3.4 Discussion

The comparative analysis of LSTM, ARIMA, and Random Forest models for stock price prediction reveals distinct strengths and weaknesses for each model:

* **LSTM** excels in handling complex, non-linear relationships and capturing long-term dependencies in time-series data. It provides high predictive accuracy but may have higher individual prediction errors compared to Random Forest.
* **ARIMA** is ideal for linear trends and seasonal patterns in time-series data, offering robust and interpretable results. However, it struggles with highly volatile data and shows higher prediction errors.

Each model's applicability depends on the specific characteristics of the stock data and the forecasting requirements. Future research should focus on addressing the overfitting issue in Random Forest models and enhancing the robustness of LSTM and ARIMA models for improved stock price prediction accuracy. This study underscores the importance of selecting the appropriate model based on the data characteristics and the specific needs of the forecasting task.

# Chapter 4: Conclusion and Future Work

## 4.1 Summary of Findings

#### Model Performance and Accuracy:

* **ARIMA** **Model**: The ARIMA model demonstrated strong capabilities in capturing linear trends and seasonal patterns in stock prices. For example, the ARIMA model for ACB showed a high R² value of 0.967, indicating a good fit to the data, with minor deviations between predicted and actual prices. Similarly, ARIMA performed well for NVDA (R² of 0.993) and VCB (R² of 0.981), accurately reflecting both short-term fluctuations and overall trends.
* **LSTM** **Model**: The LSTM model excelled in capturing complex, non-linear relationships and long-term dependencies in stock price data. It was particularly effective for volatile stocks, where traditional models might struggle. LSTM's high accuracy in predictive performance underscores its potential in financial forecasting.

#### Evaluation Metrics:

The study employed key metrics such as R², Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE) to evaluate model performance. ARIMA and LSTM models showed competitive performance with strong R² values, low MAPE, and acceptable RMSE across different stocks. For instance, ARIMA's performance for VNM (R² of 0.867, MAPE of 0.016) demonstrated its effectiveness in forecasting cyclical patterns and price volatility.

#### Data Preprocessing and Stationarity:

Ensuring data quality and achieving stationarity were crucial for effective time-series modeling. The preprocessing steps, including normalization, handling missing values, and differencing, significantly improved model performance by transforming non-stationary data into stationary series, suitable for ARIMA and LSTM models.

#### Comparison of Models:

The comparative analysis revealed distinct strengths and weaknesses for each model. LSTM models were highly effective for complex, non-linear data relationships, while ARIMA excelled in linear trends and seasonal data. Random Forest provided high accuracy but was prone to overfitting, suggesting a need for careful model selection based on specific data characteristics and forecasting requirements.

#### Practical Implications:

The findings offer practical recommendations for investors and stakeholders. By leveraging the strengths of each model, investors can make more informed decisions. For stable, trend-following stocks, ARIMA can be a reliable choice. For more volatile and complex stocks, LSTM models provide superior predictive accuracy. Random Forest models, while powerful, should be used with caution due to their overfitting potential.

## 4.2 Limitations

#### Data Quality and Preprocessing:

* **LSTM:** The performance of LSTM models heavily relies on the quality and quantity of data. Missing values, outliers, and improper normalization can significantly impact the model's ability to learn and predict accurately. The notebooks indicate that data preprocessing steps such as normalization and handling missing values were undertaken, but any inconsistencies in these steps can still pose limitations.
* **ARIMA:** ARIMA models require the time series data to be stationary. Achieving stationarity often involves differencing, which can lead to loss of important information. Additionally, ARIMA models can struggle with non-linear patterns in the data, which limits their predictive power.

#### Model Complexity and Training:

* **LSTM:** LSTM models are complex and require significant computational resources and time for training, especially when dealing with large datasets and multiple hyperparameters. This complexity can make it challenging to find the optimal model configuration. The hyperparameter tuning process, as shown in the LSTM notebook, is essential but can be computationally expensive.
* **ARIMA:** Identifying the correct order of ARIMA (p, d, q) can be time-consuming and requires thorough analysis. The simplicity of the ARIMA model can be both an advantage and a limitation, as it may not capture all the nuances in the stock price data.

#### Real-Time Prediction:

* Both models face challenges in integrating real-time data for continuous prediction. The notebooks demonstrate static datasets, but financial markets are dynamic and require models that can adapt to new data in real time.

#### Volatility and Market Anomalies:

* Stock prices are influenced by a multitude of factors including market sentiment, geopolitical events, and economic indicators. Both LSTM and ARIMA models may struggle to account for sudden market anomalies or high volatility periods without incorporating additional features or models.

## 4.3 Future Work

* Enhanced Data Preprocessing
* Hybrid Models
* Incorporating Exogenous Variables
* Hyperparameter Optimization
* Real-Time Data Integration
* Exploring Advanced Architectures
* Robust Evaluation Metrics

References

* **Box, G.E.P., & Jenkins, G.M. (1970).** *Time Series Analysis: Forecasting and Control.* San Francisco: Holden-Day.
* **Hochreiter, S., & Schmidhuber, J. (1997).** *Long Short-Term Memory.* Neural Computation, 9(8), 1735-1780.
* **Breiman, L. (2001).** *Random Forests.* Machine Learning, 45(1), 5-32.
* **Zhang, G.P. (2003).** *Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model.* Neurocomputing, 50, 159-175.
* **Chollet, F. (2015).** *Keras: Deep Learning for Humans.*
* **Makridakis, S., Wheelwright, S.C., & Hyndman, R.J. (1998).** *Forecasting: Methods and Applications.* John Wiley & Sons.
* **Goodfellow, I., Bengio, Y., & Courville, A. (2016).** *Deep Learning.* MIT Press..
* **Hyndman, R.J., & Athanasopoulos, G. (2018).** *Forecasting: Principles and Practice.*
* **Murphy, K.P. (2012).** Machine Learning: A Probabilistic Perspective. MIT Press.
* **Bishop, C.M. (2006).** Pattern Recognition and Machine Learning. Springer.
* **Hastie, T., Tibshirani, R., & Friedman, J. (2009).** The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
* **Geron, A. (2019).** Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.
* **Kingma, D.P., & Welling, M. (2014).** Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114.
* **Smola, A.J., & Schölkopf, B. (2004).** A Tutorial on Support Vector Regression. Statistics and Computing, 14(3), 199-222.
* **Chen, T., & Guestrin, C. (2016).** XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
* **Seabold, S., & Perktold, J. (2010).** Statsmodels: Econometric and Statistical Modeling with Python. Proceedings of the 9th Python in Science Conference.