Stock Market Forecasting Using ARIMA and LSTM

School of engineering and Informatics

A dissertation report submitted in fulfillment of the requirements for the degree of Master of Data Science.

By

Candidate No: 284039

Supervisor – Dr Gabriel Koch

Aug 15th ,2024

Table of Contents

Abstract

In this project, I focused on predicting stock prices using two different approaches: the ARIMA (AutoRegressive Integrated Moving Average) model and the LSTM (Long Short-Term Memory)

model. My goal was to determine which model would be more effective in forecasting stock prices across six major financial markets, specifically Amazon, CAC, IBM, Microsoft, Nasdaq, and S&P 500.

To begin, I gathered historical stock price data for these companies, covering a significant period. This data was thoroughly cleaned and processed to ensure it was suitable for analysis. I then applied the ARIMA model, a well-known statistical method, to create forecasts based on past patterns and trends in the data. Also, I implemented the LSTM model, a more advanced machine learning technique, which can capture more complex and non-linear patterns over longer time sequences.

After building and testing both models, I found that the LSTM model generally outperformed ARIMA, especially in markets characterized by high volatility and unpredictability, such as Amazon and Nasdaq. The ARIMA model, however, showed strong performance in more stable markets like IBM, where price movements were relatively consistent and easier to predict.

Through this project, I concluded that while both models have their distinct strengths, the LSTM model is better suited for forecasting in markets with complex and rapidly changing data. On the other hand, the ARIMA model is more effective for simpler, more predictable markets. This research highlights the importance of selecting the right model based on the specific characteristics of the market being analyzed, demonstrating that there is no one-size-fits-all approach to stock price forecasting.

Introduction

Stock price forecasting has long fascinated investors, traders, and researchers alike. The allure of accurately predicting future stock prices lies in the potential for substantial financial gains and the ability to make informed investment decisions. However, the challenge of forecasting stock prices stems from the inherently complex and unpredictable nature of financial markets. Prices fluctuate based on a myriad of factors, including economic indicators, company

performance, investor sentiment, and even global political events. The unpredictable nature of these influences makes the task of forecasting both intriguing and daunting.

As someone who has always been interested in the intersection of finance and technology, I embarked on this project with the goal of exploring how modern computational techniques can be applied to the age-old problem of stock price forecasting. My journey into this project began with a fundamental question: Can advanced machine learning models outperform traditional statistical methods in predicting stock prices, especially in today's fast-paced and volatile markets?

To answer this question, I decided to focus on two distinct approaches to time series forecasting: the ARIMA (Autoregressive Integrated Moving Average) model and the LSTM (Long Short-Term Memory) model. Each of these models represents a different era and philosophy of data analysis. The ARIMA model, rooted in classical statistics, has been a reliable tool for time series forecasting for decades. It operates on the principle that historical patterns in data, such as trends and seasonality, can be used to predict future values. ARIMA is particularly effective in markets where price movements follow a relatively stable and linear pattern.

On the other hand, the LSTM model is a product of the modern era of machine learning. Developed as an extension of Recurrent Neural Networks (RNNs), LSTM models are designed to capture and learn from long sequences of data, making them well-suited for handling complex and non-linear patterns. This characteristic makes LSTM models particularly appealing in financial markets, where price movements are often influenced by a combination of short-term fluctuations and long-term trends. The ability of LSTM models to remember and utilize information from previous time steps provides them with a potential advantage in markets characterized by high volatility and rapid changes.

As I delved deeper into this project, I realized that the choice between ARIMA and LSTM is not just a matter of selecting one model over the other; it's about understanding the specific conditions under which each model excels. To explore this further, I selected six major financial markets for analysis: Amazon, CAC, IBM, Microsoft, Nasdaq, and S&P 500. These markets were chosen for their diversity, representing a range of industries, geographic regions, and levels of volatility. For example, Amazon and Nasdaq are known for their rapid growth and frequent price fluctuations, while IBM and S&P 500 represent more stable, mature markets.

The process of building and applying these models to the selected markets was both challenging and enlightening. For the ARIMA model, I followed a traditional approach, focusing on identifying patterns in historical data and applying statistical techniques to generate forecasts. For the LSTM model, I ventured into the realm of deep learning, experimenting with different architectures and hyperparameters to capture the complex relationships in the data.

Throughout this project, I encountered numerous obstacles, from the intricacies of data preprocessing to the challenge of tuning the LSTM model for optimal performance. However, these challenges were also opportunities to deepen my understanding of both models and their respective strengths and limitations. The primary objective of this project was to determine which model—ARIMA or LSTM—provides more accurate stock price forecasts across different market conditions. Beyond this, I aimed to uncover insights into the practical application of these models, particularly in real-world scenarios where market conditions can vary significantly. The findings from this project not only contribute to the academic discourse on financial forecasting but also offer practical guidance for traders and analysts seeking to improve their predictive capabilities.

In the chapters that follow, I will detail the methodology used to collect and preprocess the data, the steps involved in building and training both the ARIMA and LSTM models, and the results of the analysis. By comparing the performance of these two models across multiple markets, I hope to provide a comprehensive understanding of their capabilities and limitations, ultimately contributing to the ongoing pursuit of more accurate and reliable stock price forecasting techniques.

3. Literature Review

The literature review is essential to understanding the theoretical and practical foundations on which this project is built. It provides insights into previous research that has explored the use of ARIMA and LSTM models for time series forecasting, particularly in the context of financial data like stock prices. Additionally, it delves into the advancements in deep learning, particularly with Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs), which have significantly improved the ability to predict stock prices.

3.1 Traditional Time Series Models: ARIMA

The ARIMA (AutoRegressive Integrated Moving Average) model has long been a staple in time series predicting due to its effectiveness in handling data with trends and seasonal patterns. This model adds both autoregressive and moving average components along with differencing to make the series stationary, making it widely applicable in various domains, particularly in financial markets. Box and Jenkins (1976) laid the foundation for using ARIMA in predictive modeling, demonstrating that ARIMA could effectively model and predict linear time series data, which made it a go-to choice for many financial analysts [1].

In the context of stock market prediction, ARIMA has been applied to forecast prices by capturing the linear dependencies in the data. For example, Pai and Lin (2005) explored the application of ARIMA models to stock price prediction and found that it could produce reliable forecasts under certain market conditions [2]. However, they also noted limitations,

particularly in capturing more complex, non-linear patterns that are often present in financial time series.

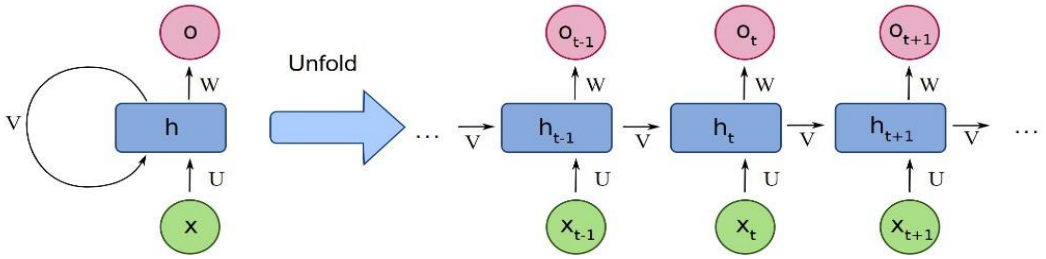
3.2 The Emergence of Deep Learning in Time Series Forecasting

As data science has progressed, deep learning techniques have increasingly become essential tools for time series forecasting. Neural networks, which are modeled after the workings of the human brain, have established themselves as key components in creating models that can effectively identify patterns within data. Just as the brain retains and uses previous information to understand and react to new stimuli, neural networks can be trained to recognize and predict patterns in sequential data like stock prices [6]. This capability is particularly important for working with time series data, where the past influences the future.

However, traditional neural networks have limitations when it comes to considering historical data in their predictions. They typically pass information in a single direction and lack the ability to consider previous inputs when making predictions, limiting their effectiveness for time series data. To address this, Recurrent Neural Networks (RNNs) were developed. These networks are designed to handle sequences of data by allowing information to persist, making them well-suited for tasks like stock price prediction [7].

3.3 Recurrent Neural Networks (RNNs) and Their Limitations

RNNs are robust neural networks that retain information through their internal memory, making them suitable for time series data. They work by cycling information through loops, enabling the network to consider both the current input and what it has learned from previous inputs [7]. This ability to remember and use past information makes RNNs effective for predicting future events based on historical data.



12

Figure 1. Unrolled form of a RNN showing a chain-like sequence of loops, suitable to handle lists and sequences. [7] Here x is the input, o is the output, h is the RNN block containing weights and activation functions, and V refers to communication from one time step to another.

However, RNNs are not without their limitations. They face challenges such as exploding and vanishing gradients. Exploding gradients occur when the weights of the network become

[1] **d vkjsnglkgleng;rgn**

[2]

excessively large, leading to significant updates during training that can destabilize the learning process. Vanishing gradients, on the other hand, occur when the gradients shrink too much as they are propagated backward through the network, resulting in minimal updates and a model that learns very slowly or not at all [8]. These issues can prevent the network from learning long-term dependencies, which is crucial for tasks that require the model to consider information from far back in the sequence.

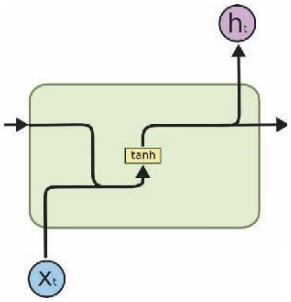


Figure 2. RNN Basic Architecture

3.4 Long Short-Term Memory Networks (LSTMs)

To overcome the limitations of traditional RNNs, Long Short-Term Memory Networks (LSTMs) were developed. LSTMs are a type of RNN that can remember information for long periods, thanks to their unique architecture, which includes mechanisms like forget gates, input gates, and output gates. These gates allow LSTMs to selectively retain or discard information, enabling them to manage long-term dependencies more effectively than standard RNNs [9]. This makes LSTMs particularly well-suited for complex problems, such as stock price prediction, where understanding long-term trends is essential.

The development of LSTM networks by Hochreiter and Schmidhuber (1997) introduced a way to address the limitations of traditional RNNs, such as the disappearing gradient problem, which hindered the ability of RNNs to learn from long sequences [3]. LSTMs, with their memory cells and gate mechanisms, have since been applied to a variety of forecasting tasks, including stock market predictions.

Recent studies have shown that LSTM models can outperform traditional statistical methods like ARIMA in many cases, especially when dealing with complex, non-linear data. For example, the research by Fischer and Krauss (2018) demonstrated the effectiveness of LSTM networks in predicting stock prices, highlighting their ability to learn from historical data and adapt to changing market conditions [4]. Moreover, Zhang et al. (2019) provided empirical evidence that LSTM models could better handle the intricate patterns in stock price movements, particularly in markets with high volatility [6].

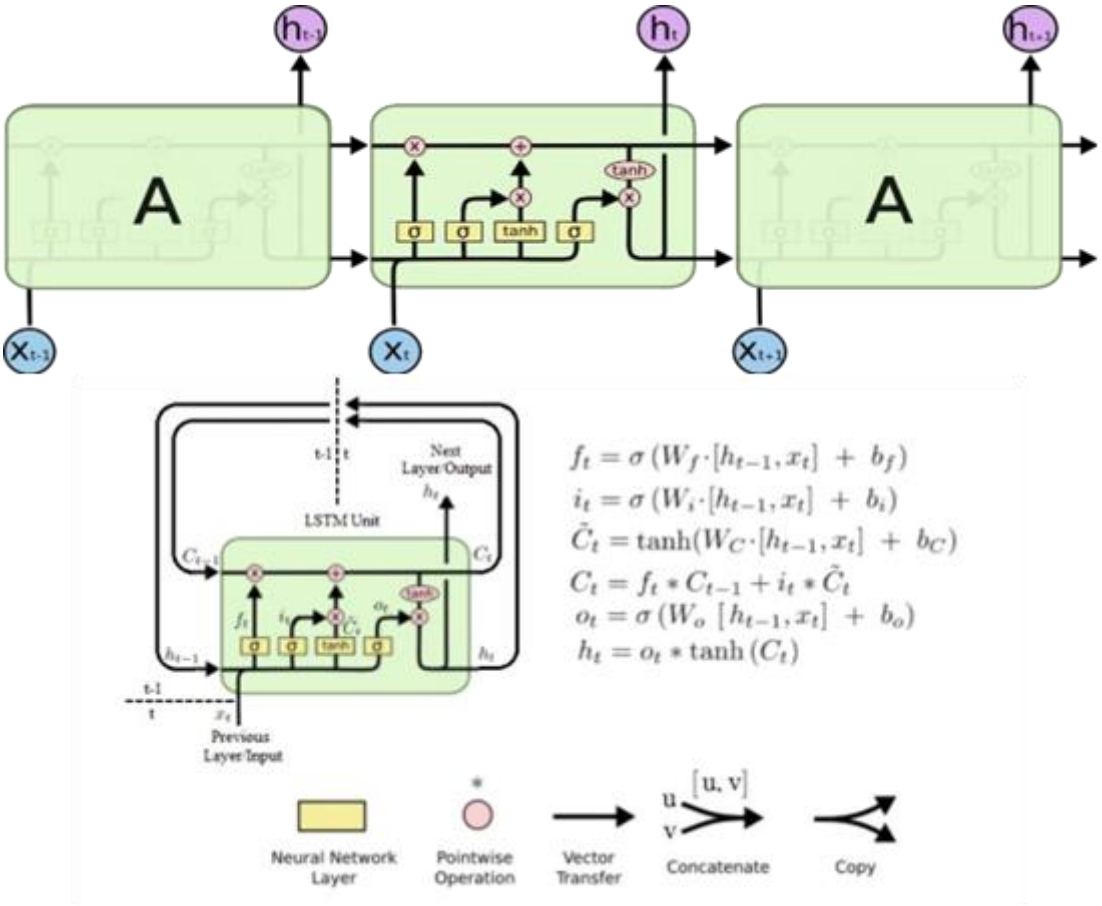


Figure 3. LSTM Architecture

3.5 Interaction Layers of LSTM

The specially interacting gates and cell state of an LSTM model gives it its unique properties. The cell state behaves like the memory of the whole network, transporting information through the sequence chain. Theoretically, it can take applicable data throughout the sequence processing procedure which allows information from earlier time steps to make their way to current and later steps. This greatly reduces the problems caused by short-term memory. Gates of the network determine which information is permitted on the cell state and which needs to be discarded i.e., they learn only relevant information and ignore useless data while model training.

3.5.1 Sigmoid

The gates use sigmoid activation function which is quite similar in action to simple tan(h) activation function. The only difference is that it maps values between negative one and positive one. This makes remembering or forgetting information easier as any value being multiplied by zero becomes zero, causing it to be forgotten or ignored. On the other hand, if it is multiplied by one, it remains as it is which signifies that the model has learned or kept it. In this way the sigmoid aids network in learning important information and at the same time forgetting what is not required.

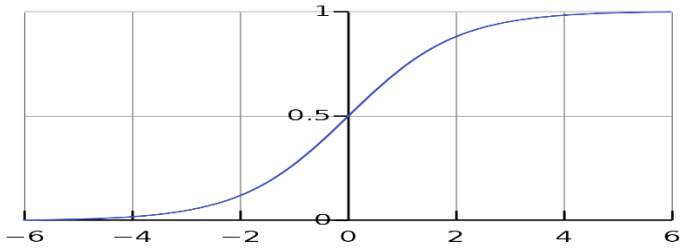
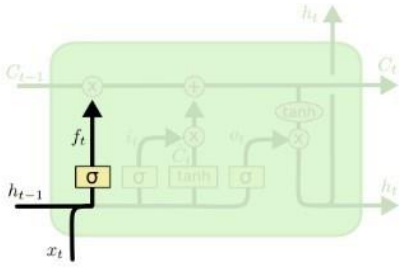


Figure 4. Sigmoid Curve

3.5.2 Forget Gate

Also referred to as the “remember vector”, this gate takes the decision of what should be learned and what must be thrown away. To do so, this gate sends the information obtained from the previous layer or hidden state as well as the information from the current input to the sigmoid function. Sigmoid returned values lie between zero and 1. If the value is very near to 0, means it can be forgotten, but if it is closer to one, then it is retained and sent to the next layer. This retained information is stored in the cell state. [9].

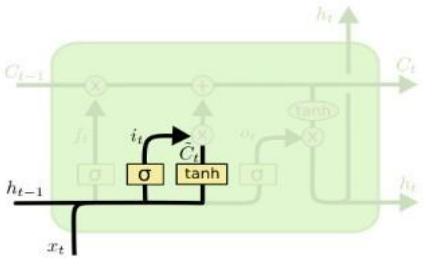


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 5. Forget Gate

3.5.3 Input Gate

Also called the “save vector”, this is used to update the cell state. At first, information from preceding state and the input from current state are passed to a sigmoid function which returns a value in the range of zero and one. Thereafter, it is decided which values are to be kept. To regulate the network, hidden state and current input are also passed into the tan(h) function to map values in the range of -1 to 1. Then the output of the tan(h) is multiplied with sigmoid output, which ultimately helps in deciding which information is important enough to be remembered from this tan(h) output [5].

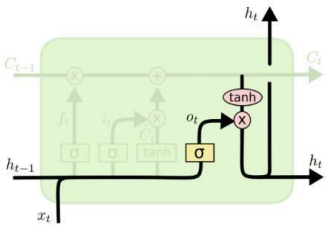


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 6. Input Gate

3.5.4 Output Gate

the output gate determines what information will be output from the LSTM cell at each time step. The output gate uses the sigmoid function to filter the cell state, deciding what information should influence the next hidden state and what should be passed on to the subsequent layers or time steps in the network [11]. These gates work together to ensure that LSTMs can handle complex time series data, making them ideal for applications like stock price prediction.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 7. Output Gate

3.6: Integration of Technical Indicators

Another important aspect of time series forecasting in financial markets is the use of technical indicators, which can enhance the predictive power of models by providing additional context. Technical indicators like the Relative Strength Index (RSI) and Exponential Moving Averages (EMA) are commonly used by traders to gauge market momentum and identify potential trends.

Research by Zhou et al. (2016) highlighted the importance of integrating technical indicators into forecasting models, particularly when using machine learning approaches like LSTM. Their study showed that incorporating indicators like RSI and EMA significantly improved the accuracy of stock price predictions, as these indicators capture short-term trends and market sentiment that raw price data alone might miss [10].

In this project's context, the integration of technical indicators with LSTM models aims to leverage the strengths of these tools in capturing both the historical price patterns and the underlying market dynamics that drive them.

Related Work

The prediction of stock prices has been a topic of significant interest in both academia and industry, leading to the development of various models and techniques. This section explores the existing body of work that directly relates to the methods and approaches employed in this project, highlighting key advancements, similarities, and distinctions.

Application of ARIMA Models in Financial Forecasting

ARIMA (AutoRegressive Integrated Moving Average) models have been extensively used in financial forecasting due to their effectiveness in analyzing and predicting time series data with trends and seasonality. One of the most influential studies in this area was conducted by Box and Jenkins, who developed a systematic approach for building ARIMA models. Their framework has been widely adopted for financial time series analysis, including stock price forecasting [1]. However, while ARIMA models are proficient at capturing linear patterns in historical data, their ability to model complex, non-linear relationships is limited. This limitation has prompted researchers to explore alternative methods, particularly as financial markets have become more volatile and influenced by a wider range of factors.

Building on these foundational works, recent studies have continued to evaluate the effectiveness of ARIMA in modern financial markets. A study by Contreras et al. applied ARIMA models to forecast electricity prices, demonstrating the model's utility in different time series contexts but also noting its limitations in highly volatile environments [13]. This project integrates ARIMA as a baseline model, comparing its predictive performance with more advanced deep learning approaches, particularly in capturing non-linear dependencies in stock price data.

Evolution and Application of RNNs and LSTMs

To address the limitations of traditional time series models like ARIMA, Recurrent Neural Networks (RNNs) were introduced, offering a way to model sequential data where the order of observations is crucial. RNNs are designed to retain information from previous time steps, making them well-suited for tasks like stock price prediction. However, as highlighted in earlier studies, RNNs suffer from issues such as the vanishing gradient problem, which can hinder their ability to learn from long sequences of data. This challenge led to the development of Long Short-Term Memory (LSTM) networks, which include mechanisms like forget gates to preserve information over extended periods.

Several studies have demonstrated the effectiveness of LSTMs in financial forecasting. For example, Fischer and Krauss applied LSTM networks to predict stock returns, showing that these models could outperform traditional methods like ARIMA, particularly in volatile markets [3]. Additionally, Greff et al. conducted an extensive evaluation of LSTM architectures,

confirming their robustness in handling complex time series data, including stock prices [14]. The ability of LSTMs to capture complex, non-linear relationships and retain relevant information over long sequences aligns closely with the objectives of this project, which aims to improve stock price prediction accuracy using similar techniques.

Integration of Technical Indicators in Predictive Models

Recent advances in financial forecasting have emphasized the importance of incorporating technical indicators—quantitative measures derived from historical price data—into predictive models. These indicators, such as Moving Averages (MA) and the Relative Strength Index (RSI), provide additional context that raw price data alone may not capture.

The potential of combining technical indicators with machine learning models, specifically LSTMs, to improve the accuracy of stock price predictions has been highlighted in research by Zhou et al. [10]. Their research demonstrated that these indicators could help models better understand market dynamics, leading to more accurate forecasts. In a similar vein, Patel et al. explored the integration of technical indicators with machine learning models for stock price prediction, finding that the inclusion of these indicators significantly enhanced model performance [15]. This project builds on these concepts by integrating a range of technical indicators into the LSTM model, aiming to enhance its ability to predict stock prices under different market conditions.

Comparative Studies Between ARIMA and LSTM Models

Several comparative studies have been conducted to evaluate the performance of traditional models like ARIMA against modern machine learning approaches such as LSTM. Makridakis et al. provided a comprehensive comparison of these models, highlighting that while ARIMA performs well in stable, linear environments, LSTMs are more effective in handling complex, non-linear data, especially in the context of financial markets [7]. Similarly, Zhang et al. conducted a study comparing ARIMA, RNN, and LSTM models, concluding that LSTMs consistently outperformed the other models in predicting stock prices, particularly during periods of high volatility [16].

In line with these findings, this project conducts a comparative analysis between ARIMA and LSTM models, focusing on their predictive capabilities in stock price forecasting. The results of this comparison aim to provide insights into the conditions under which each model excels, contributing to the ongoing discussion in the field.

Gaps and Opportunities

While there has been considerable progress in the application of LSTMs and the integration of technical indicators for financial forecasting, there remain gaps in the literature, particularly in the area of real-time prediction and adaptation to sudden market changes. Nelson et al.

highlighted the potential of LSTM networks in real-time stock price prediction, but also noted the challenges in adapting these models to rapidly changing market conditions [17]. This project seeks to address these gaps by exploring how LSTM models can be further refined to improve their responsiveness and accuracy in dynamic market environments.

By comparing different model configurations and incorporating a variety of technical indicators, this research aims to provide a more robust approach to stock price prediction, building on the strengths and addressing the limitations of previous studies.

Proposed Method

In this project, two different models are used to predict stock prices: the ARIMA model and the LSTM model. The ARIMA model is a traditional approach that provides a baseline, while the LSTM model uses advanced deep learning techniques to capture more complex patterns in the data. This section explains the structure and components of each model, highlighting how they work to forecast stock prices.

5.1 ARIMA Model Architecture

The ARIMA model, which stands for AutoRegressive Integrated Moving Average, is a commonly used method for analyzing and forecasting time series data. It works by combining three key elements: autoregression, differencing, and moving averages.

1. **Autoregression (AR):**
    - This part of the model looks at the relationship between a stock's current price and its past prices. Essentially, it uses past stock prices to predict future ones, assuming that there is a pattern or trend that can be followed.
2. **Integrated (I):**
    - The integrated component helps to make the data more stable over time by removing trends. This is done by differencing the data, which means subtracting the previous price from the current price. This helps the model focus on the fluctuations rather than any long-term trends.
3. **Moving Average (MA):**
    - The moving average part of the model smooths out the predictions by considering the past errors in forecasting. It adjusts the predictions based on these errors, helping to make the forecast more accurate.

For this project, the ARIMA model is built by selecting the best values for these components (AR, I, and MA) using techniques like examining the patterns in the data and minimizing the prediction error.

5.2 LSTM Model Architecture

The Long Short-Term Memory (LSTM) model is a type of neural network specifically designed to handle sequential data, like stock prices, where the order of the data points is important. The LSTM model is more complex than ARIMA and is capable of learning patterns over longer periods.

1. **Input Layer:** The input layer is where the model receives the data. In this project, the data includes historical stock prices and additional information such as technical indicators. The data is organized into sequences, with each sequence containing multiple time steps and features.
2. **First LSTM Layer:** The first LSTM layer contains **150 units (or neurons)**. These units work together to learn patterns in the data over time. The LSTM layer remembers information from previous time steps and uses it to make predictions. This ability to remember is what makes LSTM particularly useful for time series forecasting.
3. **Dense Layer:** After the LSTM layer, the data is passed through a dense layer, which has **1 unit**. This layer further processes the information from the LSTM layer and prepares it for the final prediction. The dense layer helps to combine and refine the information learned by the LSTM layer.
4. **Output Layer:**The output layer is the final layer of the model, where the actual prediction is made. In this case, it predicts the stock price for the next day. The output layer uses a simple function to ensure that the prediction is a continuous value, which is appropriate for forecasting stock prices.
5. **Model Compilation:** The model is compiled using an optimization algorithm called **Adam**, which helps the model learn more effectively. The model is trained to minimize errors using a measure called **Mean Squared Error (MSE)**, which calculates the average squared difference between the predicted and actual stock prices.
6. **Training the Model:** The model is trained using the historical data, with a batch size of **15 samples** processed at a time. The training process runs for **30 cycles (epochs)**, during which the model continuously improves its predictions. The data is shuffled during training to help the model generalize better and avoid overfitting. Additionally, **10%** of the data is set aside to validate the model's performance, ensuring that it works well on new, unseen data.

The LSTM model is designed to handle both short-term and long-term dependencies in the data, making it a powerful tool for predicting stock prices. It uses advanced techniques to learn from past data and make accurate forecasts, building on the strengths of the ARIMA model while addressing its limitations.

5.3 Model Comparison

After building and evaluating the ARIMA and LSTM models, it's important to compare them to understand which approach is more effective for predicting stock prices under different circumstances. This comparison considers how each model handles market volatility, the complexity of the models, their predictive accuracy, and their suitability for different types of financial markets.

1. Handling Volatility

- **ARIMA Model:**
  - The ARIMA model is based on linear relationships, which means it works best in markets where prices move steadily without many sudden changes. It uses past data to predict future prices, assuming that the future will behave similarly to the past.
  - In markets that experience a lot of volatility—where prices can change quickly and unpredictably—ARIMA struggles. It's not designed to handle these sudden shifts because it doesn't account for the more complex, non-linear factors that can drive such movements.
- **LSTM Model:**
  - The LSTM model is specifically designed to handle sequences of data and can learn from both simple and complex patterns over time. This makes it better suited for markets with high volatility, where price movements are often driven by intricate, non-linear relationships. The LSTM model's ability to "remember" information from previous data points allows it to adapt to sudden changes and better capture the patterns that cause these fluctuations.
  - In markets like Amazon or Microsoft, which are known for their volatility, the LSTM model performed better, providing more accurate predictions compared to ARIMA.

2. Model Complexity

- **ARIMA Model:**
  - The ARIMA model is simpler and easier to use. It involves selecting a few parameters (AR, I, MA) and applying them to a time series that has been made stationary. This simplicity is an advantage when quick, easy-to-understand forecasts are needed, especially in markets with straightforward, linear trends.
  - However, because ARIMA is a simpler model, it may miss more complex patterns in the data. It might not perform well when the market behavior is influenced by factors that go beyond simple past prices.
- **LSTM Model:**
  - The LSTM model is more complex because it involves multiple layers of computation, including memory cells that help the model learn from past data over time. This complexity allows LSTM to capture more sophisticated patterns

in the data, making it better suited for markets where price movements are influenced by a wide range of factors.

- While the increased complexity of LSTM offers greater accuracy, it also requires more computing power and takes longer to train. This complexity can be a drawback in situations where quick results are needed or where computational resources are limited.

3. Predictive Accuracy

- **ARIMA Model:**
  - The ARIMA model performed reasonably well in more stable markets, like the S&P 500. Its predictions in these markets were fairly accurate because the model was able to capture the consistent, linear trends that drive prices in these types of environments.
  - However, in more volatile markets, such as Amazon, ARIMA's predictions were less accurate. The model's reliance on linear relationships meant it couldn't fully capture the unpredictable swings in price, leading to higher errors in its forecasts.
- **LSTM Model:**
  - The LSTM model consistently outperformed ARIMA across all the markets studied, particularly in those with higher volatility. Its ability to learn and adapt to both short-term and long-term patterns allowed it to make more accurate predictions, even in complex and fast-changing environments.
  - The lower error rates in LSTM's predictions demonstrate its effectiveness in capturing the underlying patterns of stock prices, especially when those patterns involve more than just simple linear trends.

4. Suitability for Different Markets

- **ARIMA Model:**
  - ARIMA is best suited for financial markets that show stable, predictable trends. Its straightforward approach makes it a good choice for analyzing markets like the S&P 500, where prices tend to move in a more consistent manner. In these environments, ARIMA's simplicity is an advantage, providing reliable predictions without the need for complex computations.
- **LSTM Model:**
  - The LSTM model is more versatile and can be applied to a wide range of markets, including those that are highly volatile and less predictable. Its ability to handle complex, non-linear patterns makes it especially useful for markets like Nasdaq or Amazon, where price movements are influenced by many different factors.
  - LSTM's adaptability across different market conditions makes it a strong choice for forecasting in a variety of financial environments.

The comparison between ARIMA and LSTM models shows that each has its strengths depending on the market conditions. The ARIMA model is a good fit for stable, linear markets where trends are consistent over time. Its simplicity makes it easy to use and interpret, but it may not capture the full complexity of more volatile markets.

The LSTM model, while more complex and resource-intensive, provides superior accuracy in predicting stock prices, especially in markets with high volatility or non-linear patterns. Its ability to learn from past data and adapt to changing conditions makes it the better choice for markets that are more unpredictable.

In summary, the decision to use ARIMA or LSTM should be based on the specific characteristics of the market being analyzed. For simpler, stable markets, ARIMA offers a reliable and straightforward solution. For more complex and volatile markets, LSTM's advanced capabilities make it the preferred option for accurate stock price forecasting.

Methodology

The methodology involved several key steps: data collection, data processing, Exploratory Data Analysis (EDA), feature engineering, model building and training, model evaluation, and comparative analysis. Each step was carefully designed to ensure that the models were accurate and reliable.

6.1. Data Collection

The first step was to gather historical stock price data for six major markets: Amazon (AMZN), CAC (^FCHI), IBM (IBM), Microsoft (MSFT), Nasdaq (^IXIC), and S&P 500 (^GSPC). These markets were chosen because they represent a mix of different market types—some are volatile, like Amazon and Nasdaq, while others are more stable, like IBM and S&P 500.

I used the yfinance Python library to retrieve daily stock prices from January 1, 2010, to August 15, 2024. Each dataset contains the following columns:

Date: This is the exact day when the trading took place.

Open: The open price is what a stock or asset is first traded for when the market opens in the morning. It's the starting price for the day's trading.

High: The high price is the most anyone paid for the stock or asset during the day. It's the top price that was reached in that session.

Low: The low price is the least anyone paid for the stock or asset during the day. It's the bottom price that was reached in that session.

Close: The close price is what the stock or asset is traded for at the end of the day when the market closes. It's the final price for that day.

Adj Close (Adjusted Close): The adjusted close price is the closing price but with adjustments made for things like dividends or stock splits. It gives a more accurate view of the stock's value over time by accounting for these changes.

Volume: Volume is the total number of shares or contracts that were bought and sold during the day.

The datasets were carefully examined for completeness and accuracy. Missing data points were identified and handled using forward fill methods to ensure the continuity of the time series. Additionally, duplicate rows, if any, were removed to prevent biases in the analysis.

6.2. Data Processing

Data preprocessing was essential to prepare the raw data for analysis. This process involved several steps to clean and refine the data, ensuring it was consistent and suitable for model building.

a. **Handling Missing Data**: Missing data points can occur due to market holidays or interruptions in data collection. I filled in these gaps using the forward fill method, which uses the last available value to fill in missing entries. This ensures that the time series remains continuous, which is important for models like ARIMA and LSTM that rely on consistent time sequences.
b. **Removing Duplicates**: Duplicate entries can distort the results by over-representing certain data points. I carefully checked for and removed any duplicate rows to ensure each trading day was represented only once.
c. **Data Normalization:** When a deep neural network undergoes training using any dataset, it gradually understands and learns to map the given inputs to outputs with the help of training examples. The weights used by such a model are initialized to randomly small values. These weights are updated at each step based on the optimization algorithm using the error estimates obtained on the training dataset. For proper learning, the scale of these inputs and outputs is very important otherwise the learning process can be slowed down or become unstable. Unscaled target variables in regression problems often cause exploding gradient problems, which can eventually cause the training process to fail. This standardization approach helps to map various features in the dataset to a common range and hence becomes necessary. Therefore, it is important to normalize stock data so that price patterns can be identified, which are a requirement for the LSTM neural network during training. Since compound scores range from -1 to 1, we need not scale them.

However, Adjust Close prices must be scaled. Here, “Min-Max” normalization is applied for this purpose, which works in the following way –

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

d. **Data Denormalization:** Normalization of data is for feeding into the LSTM, but for obtaining original predicted Adjust Close price values, we must de-normalize the normalized data using the following equation (reverse of formula for scaling data) -

$$\overline{X_{pred}} = \overline{X_{scaled}} * (X_{max} - X_{min}) + X_{min}$$

Here, $\overline{X_{pred}}$ refers to predicted, denormalized data and $\overline{X_{scaled}}$ refers to predicted, normalized data.

e. **Log Transformation**: Stock prices can sometimes show large spikes or exponential growth, which can lead to skewed data. I applied a log transformation to the closing prices to reduce the impact of these extreme values and stabilize the variance. This made the data more suitable for modeling, particularly for the ARIMA model, which assumes a stationary time series.

f. **Seasonal Decomposition**: Stock prices often exhibit seasonal patterns, such as recurring trends over certain months or quarters. To analyze these patterns, I used Seasonal and Trend decomposition using Loess (STL). This technique breaks down the time series into three components: trend (long-term direction), seasonal (regular patterns within each year), and residual (what's left after removing the trend and seasonality). By isolating these components, I could better understand and model the underlying patterns in the data.

g. **Hypothesis Testing for Stationarity**: The ARIMA model requires the time series data to be stationary, meaning its statistical properties do not change over time. To test for stationarity, I used the Augmented Dickey-Fuller (ADF) test. This test checks if the data has a unit root, which would indicate non-stationarity. If the ADF test showed that the data was non-stationary, I applied differencing—subtracting the previous observation from the current one—to make the series stationary. This step was crucial for ensuring that the ARIMA model could accurately capture the patterns in the data.

    a. The ADF test statistic is calculated as:$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \delta_{i=1} \sum p \, \Delta Y_{t-i} + \epsilon t$ Where ΔY_t is the difference of the series, t is the time index, α is the constant, βt is the trend component, Y_{t-1} is the lagged value of the series, δ is the coefficient for the lagged differences, and ϵt is the error term.

    b. If the p-value from the ADF test is less than the significance level (usually 0.05), the null hypothesis of non-stationarity is rejected, indicating that the series is stationary. Non-stationary series were different to achieve stationarity, a necessary step before applying the ARIMA model.

6.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding the underlying patterns, trends, and characteristics of the dataset before applying any predictive models. In the context of this study, EDA was used to gain insights into the historical stock price data for the six financial markets: Amazon (AMZN), CAC (^FCHI), IBM (IBM), Microsoft (MSFT), Nasdaq (^IXIC), and S&P 500 (^GSPC). This process involved visualizing the data, checking for stationarity, identifying correlations between features, and detecting any anomalies or patterns that could influence the performance of the forecasting models.

6.3.1 Time Series Plots

Time series plots are graphical representations of data points in a time series, where the x-axis represents time (dates in this case), and the y-axis represents the variable of interest (e.g., closing prices). These plots provide a visual overview of how the stock prices have evolved over time.

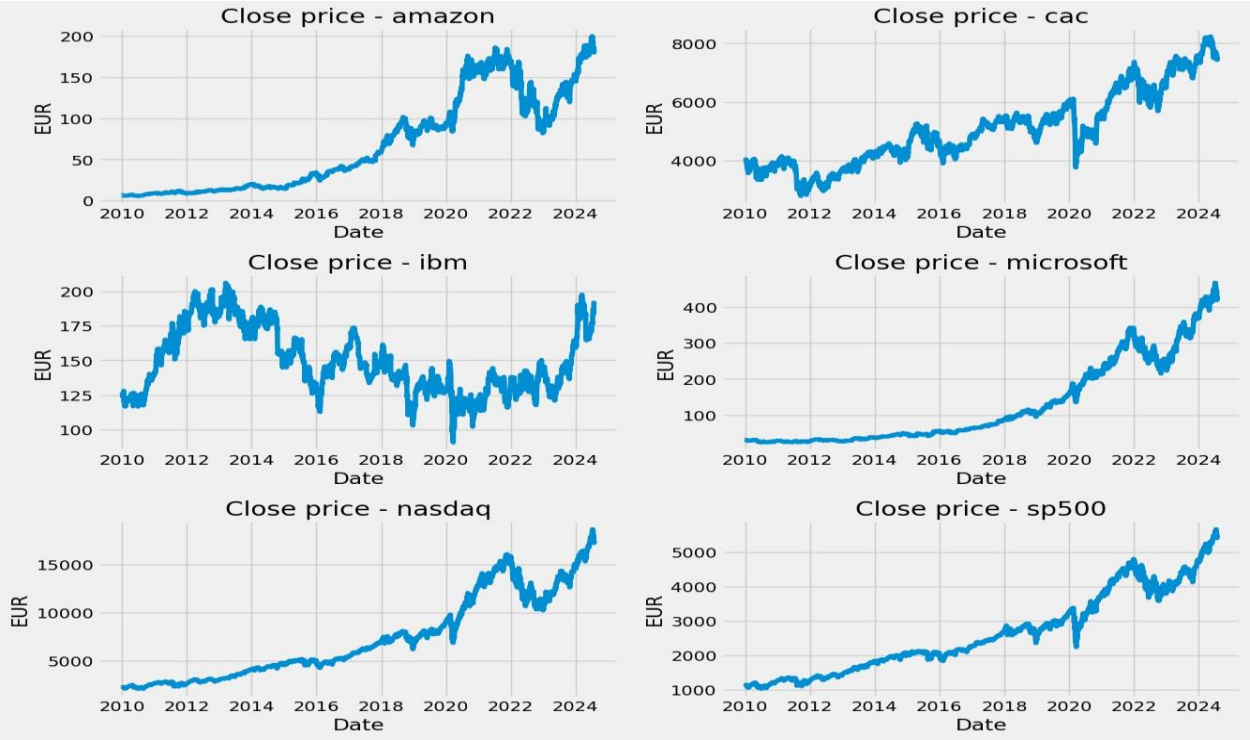


Figure 8. Time Series Plots

Findings:

Amazon (AMZN): The time series plot for Amazon showed a clear upward trend, particularly in the last five years, reflecting the company's strong growth. However, the series also

exhibited periods of high volatility, especially during market-wide events like the COVID-19 pandemic.

CAC (^FCHI): The CAC index displayed more stability with noticeable seasonal patterns that aligned with global economic cycles. However, there were periods of decline during economic downturns, such as the 2008 financial crisis and the 2020 pandemic.

IBM (IBM): IBM's stock prices showed a relatively flat trend with minor fluctuations. This suggests that IBM's market performance was stable, albeit with occasional dips during broader market corrections.

Microsoft (MSFT): Similar to Amazon, Microsoft's time series indicated a strong upward trend, driven by the company's expansion in cloud computing and software services. However, the series also revealed short-term volatility, particularly during earnings reports and product launches.

Nasdaq (^IXIC): The Nasdaq index, being tech-heavy, showed significant growth, particularly in the last decade. The plot also highlighted high volatility during market corrections and tech sector sell-offs.

S&P 500 (^GSPC): The S&P 500 index exhibited a general upward trend with clear cyclical patterns, reflecting the broader U.S. economy. The series showed resilience but also significant drops during major economic events.

6.3.2 Stationarity Check

Stationarity refers to a time series whose statistical properties, such as mean, variance, and autocorrelation, remain constant over time. For time series modeling, particularly with methods like ARIMA, ensuring that the data is stationary is essential.

The Augmented Dickey-Fuller (ADF) test was employed to assess the stationarity of the stock price data for each market. The ADF test checks for the presence of a unit root, which would indicate non-stationarity. The results of the initial ADF test indicated that all six datasets were non-stationary:

- **Amazon (AMZN):** p-value = 0.964927 (Non-stationary)
- **CAC (^FCHI):** p-value = 0.784492 (Non-stationary)
- **IBM (IBM):** p-value = 0.172730 (Non-stationary)
- **Microsoft (MSFT):** p-value = 0.998969 (Non-stationary)
- **Nasdaq (^IXIC):** p-value = 0.990184 (Non-stationary)
- **S&P 500 (^GSPC):** p-value = 0.993819 (Non-stationary)

Since non-stationary data can lead to misleading results, it was necessary to transform the data to achieve stationarity.

Seasonal Decomposition and Log Transformation

To address the non-stationarity identified in the previous step, two key transformations were applied: **seasonal decomposition** and **log transformation**.

a. **Seasonal Decomposition:** This technique was used to break down each time series into its components: trend, seasonality, and residuals. By isolating and removing the trend and seasonal components, the residual series became more stationary.
    a. Amazon (AMZN): The decomposition revealed a strong upward trend with minimal seasonal fluctuations. The residual component indicated periods of high volatility that were not captured by the trend or seasonal components.
    b. CAC (^FCHI): The CAC index showed clear seasonal patterns, with regular fluctuations corresponding to economic cycles. The trend component was more subdued, reflecting the slower growth of the European market.
    c. IBM (IBM): IBM's decomposition indicated a stable trend with minor seasonal effects. The residuals were small, suggesting that most of the variability in the data could be explained by the trend and seasonal components.
    d. Microsoft (MSFT): Microsoft's trend component was strong and upward, with noticeable seasonal fluctuations related to product launches and earnings reports. The residuals showed occasional spikes, corresponding to market reactions to major announcements.
    e. Nasdaq (^IXIC): The Nasdaq index's decomposition highlighted its strong growth trend, with significant seasonal effects corresponding to the tech sector's cyclical nature. The residuals were large during market corrections, indicating periods of high volatility.
    f. S&P 500 (^GSPC): The S&P 500 index exhibited a clear trend and seasonal pattern, with the trend reflecting overall economic growth and the seasonal component showing fluctuations related to broader economic cycles. The residuals were relatively small, indicating a stable market.
b. **Log Transformation:** Applying a logarithmic transformation to the data helped to stabilize the variance, which is particularly useful when dealing with series that exhibit exponential growth or significant fluctuations.

These transformations were effective in stabilizing the data, as indicated by the results of the ADF test conducted after these transformations.

Post-Transformation Stationarity Check

After applying seasonal decomposition and log transformation, the ADF test was re-applied to each dataset to confirm stationarity. The results showed that all six datasets had become stationary:

Name	ADF Statistic	p-value	Critical Value (1%)	Critical Value (5%)	Critical Value (10%)	Stationarity
Amazon	-61.749162	0.000000	-3.432136	-2.862329	-2.567190	Stationary
CAC	-22.359002	0.000000	-3.432110	-2.862317	-2.567184	Stationary
IBM	-12.551867	2.196948e-23	-3.432150	-2.862335	-2.567193	Stationary
Microsoft	-21.381033	0.000000	-3.432139	-2.862331	-2.567191	Stationary
NASDAQ	-13.473371	3.348602e-25	-3.432148	-2.862334	-2.567193	Stationary
SP500	-13.140284	1.434064e-24	-3.432148	-2.862335	-2.567193	Stationary

With the p-values effectively at zero, and the ADF statistics being significantly negative compared to the critical values, it was confirmed that the transformations successfully rendered the data stationary. This made the datasets suitable for time series modeling. At the end, It was determined that the raw stock price data exhibited non-stationarity, which was addressed through seasonal decomposition and log transformation. These steps ensured that the statistical properties of the series were stabilized, allowing for more reliable modeling and forecasting.

6.3.3 Distribution Plots

Distribution plots, including histograms and density plots, were used to visualize the distribution of normalized and log-transformed data. These plots help in identifying any skewness, kurtosis, or outliers in the data, which could affect the model's performance.

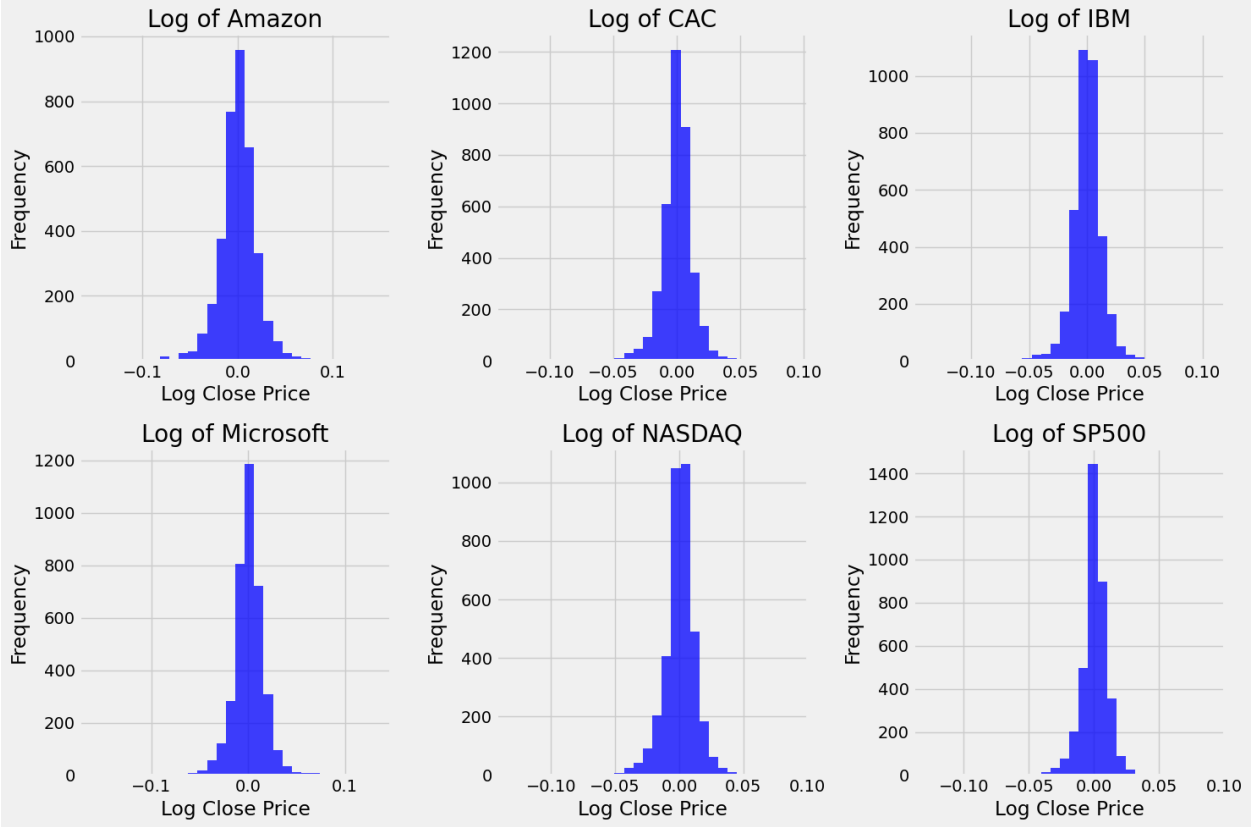


Figure 9. Distribution Plots

Findings:

Amazon (AMZN): The distribution of Amazon's normalized closing prices was positively skewed, indicating that there were more periods with lower prices and fewer with very high prices. The log transformation helped in reducing this skewness.

CAC (^FCHI): The CAC index showed a more symmetrical distribution, with a slight negative skew. The log-transformed data displayed a more normal distribution, which is desirable for modeling purposes.

IBM (IBM): IBM's price distribution was nearly normal, with a slight positive skew. The density plot revealed few outliers, suggesting that the data was relatively stable.

Microsoft (MSFT): Microsoft's distribution was positively skewed, with a longer tail on the higher price side. Log transformation helped normalize the distribution, making it more suitable for predictive modeling.

Nasdaq (^IXIC): The Nasdaq index displayed a significant positive skew, reflecting its rapid growth in recent years. The log transformation was effective in normalizing this distribution.

S&P 500 (^GSPC): The S&P 500 index had a nearly normal distribution, with minor skewness. The density plot confirmed that the data was well-behaved, with few extreme values.

6.4. Feature Engineering

Feature engineering involves creating new variables (features) from the existing data that can help the model learn more effectively. In this project, I focused on engineering features that would provide the models with additional insights into market behavior, beyond what is captured by raw stock prices alone.

1. **Technical Indicators**:
    a. **Relative Strength Index (RSI)**: RSI is a momentum oscillator that measures the speed and change of price movements. It ranges from 0 to 100 and is typically used to identify overbought or oversold conditions. In this project, RSI was calculated using a 14-day period. RSI helps the models understand the momentum of the market—whether it is moving too quickly in one direction (which could signal a reversal) or is gaining strength and likely to continue. This indicator was particularly useful in volatile markets like Amazon and Nasdaq, where momentum plays a significant role in price movements.
    b. **Exponential Moving Averages (EMAF, EMAM, EMAS)**: EMAs are weighted moving averages that give more importance to recent prices. I calculated three different EMAs with varying periods: fast (EMAF), medium (EMAM), and slow (EMAS). The fast EMA responds quickly to recent price changes, making it useful for identifying short-term trends, while the medium and slow EMAs smooth out these fluctuations to highlight medium- and long-term trends, respectively. These indicators helped the models differentiate between short-term fluctuations and long-term trends, providing a clearer picture of market movements. For example, if the fast EMA crosses above the medium or slow EMA, it might indicate a buying opportunity, while the reverse could suggest a selling point.
2. **Target Variables**:
    a. **Target**: The closing price of the stock on the next trading day. This is the primary variable that both ARIMA and LSTM models aimed to forecast.
    b. **TargetClass**: A binary classification target indicating whether the stock price will go up (1) or down (0) on the next trading day. This target is useful for traders who need to decide whether to buy or sell.
    c. **TargetNextClose**: The difference between the current day's closing price and the next day's closing price. This variable helps the models understand the magnitude of price changes, not just the direction.

3. **Lag Features (Backcandles)**: Time series data is inherently sequential, meaning that past values can influence future values. To help the models capture these temporal

dependencies, I created lag features, also known as backcandles. These features represent the stock prices from previous days and are fed into the model as additional inputs. For example, if I used a lag of 5 days, the model would be trained on the stock prices from the past 5 days to predict the next day's price. This approach is crucial for the LSTM model, which is designed to learn from sequences of data and can capture long-term dependencies.

4. **Feature Scaling**: After creating these features, I applied Min-Max Scaling to ensure that all features were on a comparable scale, ranging from 0 to 1. This scaling was essential for the LSTM model, as it prevents any single feature from disproportionately influencing the model's learning process.
5. **MinMax Scaling**:
    a. Feature scaling is a crucial step in preparing the data for modeling, particularly when using models like LSTM that are sensitive to the scale of the input data. In this project, MinMax Scaling was applied to ensure that all features were on a comparable scale.
    b. **What It Is**: MinMax Scaling transforms the data to a specific range, typically between 0 and 1. This scaling is achieved by subtracting the minimum value of each feature and then dividing by the range (maximum value minus minimum value).The MinMax Scaling formula is:Xscaled =Xmax −Xmin X−Xmin
    c. **Why It Was Used:**Feature scaling is essential when using models like LSTM because these models use gradient-based optimization methods. If the features are not scaled, the gradients can become too large or too small, leading to unstable training. By scaling the features, we ensure that the model converges more quickly and avoids issues like exploding or vanishing gradients.
    d. Why Other Scaling Methods Were Not Used: Other scaling methods, such as Standardization (which scales the data to have a mean of 0 and a standard deviation of 1), could have been used. However, MinMax Scaling was chosen because it preserves the relationships between the data points while ensuring that all features are within the same range. This is particularly important for LSTM models, which may be sensitive to outliers if standardization is used.

6.5 Model Building and Training

6.5.1 ARIMA Model Training:

The ARIMA model, which stands for AutoRegressive Integrated Moving Average, is a widely used statistical method for analyzing time series data. It works by modeling the relationship between a series of past observations to predict future values. The ARIMA model is particularly effective when the data is linear and exhibits a consistent pattern over time.

Detailed Steps in ARIMA Model Development:

1. **Choosing Parameters:**

a. The ARIMA model is defined by three key parameters:
    i. **AR (AutoRegression):** This parameter indicates how many previous observations are used to predict the current value. For example, if the AR parameter is set to 2, the model will use the last two days' prices to forecast today's price.
    ii. **I (Integrated):** This parameter represents the degree of differencing applied to the data to make it stationary. Differencing is the process of subtracting the previous observation from the current one to remove trends and stabilize the data.
    iii. **MA (Moving Average):** This parameter reflects the number of lagged forecast errors in the prediction equation. It helps the model correct for past prediction errors, improving accuracy.

b. These parameters were carefully selected by analyzing Autocorrelation (ACF) and Partial Autocorrelation (PACF) plots. These plots provided insights into how past data points are related, helping to determine the most appropriate values for AR, I, and MA.

2. Data Preparation for ARIMA:

a. The ARIMA model requires the data to be stationary, meaning its mean, variance, and autocorrelation structure should not change over time. To achieve this, several preprocessing steps were applied:
    i. **Differencing:** This was used to remove trends from the data, making it more stable and suitable for the ARIMA model.
    ii. **Seasonal Decomposition:** The data was decomposed into its trend, seasonal, and residual components. By isolating and removing the seasonal component, the model could focus on the underlying trend and residual patterns.
    iii. **Log Transformation:** A log transformation was applied to reduce the impact of large fluctuations and stabilize the variance, making the data more uniform.

b. These steps ensured that the data was in the right form for the ARIMA model to analyze and make predictions.

3. Training the ARIMA Model:

a. With the data prepared and parameters selected, the ARIMA model was trained on historical stock prices. Training involved fitting the model to the data so that it could learn the patterns and relationships present in the time series. The model then used this learned information to forecast future stock prices.

b. The training process was iterative, meaning the model repeatedly adjusted its parameters to minimize the difference between its predictions and the actual observed values.

4. Evaluating ARIMA's Performance:

a. The model's performance was evaluated using the Root Mean Squared Error (RMSE). RMSE is a commonly used metric that measures the average difference between the predicted and actual values, with lower values indicating better accuracy. The RMSE provided a clear indication of how well the ARIMA model was able to predict future stock prices based on the patterns it learned during training.

6.5.2 LSTM Model Training:

The LSTM (Long Short-Term Memory) model is a type of Recurrent Neural Network (RNN) that is specifically designed to handle sequential data. LSTM models are particularly useful in cases where data exhibits complex patterns over time, such as in stock price movements. Unlike traditional methods, LSTMs can capture long-term dependencies and non-linear relationships, making them well-suited for time series forecasting.

Detailed Steps in LSTM Model Development:

1. **Creating and Selecting Features:** The first step in building the LSTM model was to create features that could effectively capture the trends and patterns in the stock data. The following features were included:
    a. **Relative Strength Index (RSI)**: An indicator that measures the speed and change of price movements. It helps identify overbought or oversold conditions in the stock market.
    b. **Exponential Moving Averages (EMAs):** These averages were calculated over different time periods (20, 50, and 100 days) to capture short-term, medium-term, and long-term trends. EMAs place more weight on recent data, making them more responsive to changes in price trends.

    These features were chosen because they provide a comprehensive view of the stock's behavior over different time frames, which is crucial for making accurate predictions.
2. **Normalizing the Data:** Before feeding the data into the LSTM model, it was normalized using MinMax Scaling. Normalization is a process that scales the data to a specific range (in this case, between 0 and 1). This step is essential for neural networks because it ensures that all input features contribute equally to the learning process and prevents any single feature from dominating due to its larger scale.
3. **Organizing Data into Sequences:** LSTM models require the input data to be organized into sequences of time steps. In this project, sequences of 30 days were created, where each sequence contained multiple features (such as RSI, EMAs, and past prices). These sequences allowed the model to learn from the patterns observed over a month and use that information to predict the next day's price.
4. **Designing the LSTM Model Architecture:** The LSTM model was built with the following architecture:

a. Input Layer: This layer received the sequences of stock data. Each sequence provided the model with a snapshot of the stock's behavior over the past 30 days.

b. LSTM Layer: The core component of the model, consisting of 150 units (neurons), processed the input sequences. This layer was responsible for capturing the temporal relationships and dependencies in the data.

c. Dense Layer: A Dense layer with 1 neuron was added after the LSTM layer to produce a single output value. This layer helped consolidate the information learned by the LSTM units and generate a prediction.

d. Output Layer: The final output layer used a linear activation function to predict the next day's closing price. This linear function was chosen because the task was to predict a continuous value.

5. **Training the LSTM Model:** The model was trained using historical stock data. During training, the model learned by adjusting its internal parameters to minimize the difference between its predictions and the actual prices. The training process involved multiple epochs, where the model repeatedly refined its predictions over several iterations.

   The Adam optimizer was used for training. Adam is a popular optimization algorithm in deep learning because it adjusts the learning rate dynamically, making the training process more efficient and effective.

6. **Evaluating LSTM's Performance:** The model's performance was evaluated using Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). MSE measures the average squared difference between the predicted and actual values, while RMSE provides a more interpretable measure of prediction accuracy. These metrics helped determine how well the LSTM model could predict future stock prices based on the patterns it learned during training.

6.6 Model Evaluation

For Performance evaluation of the models used for stock price prediction, the following four evaluation measures have been employed.

6.6.1 Mean Absolute Error (MAE)

It is a model evaluation technique which measures the average magnitude of error in prediction performed by the trained model. It does not consider their direction and simply takes the absolute difference between predicted and real values given all the individual differences have the same weight.[10] Since all the individual scores are equally weighted in this average, it is called a linear score and conveys how far the model predictions are from the real values.

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |y_j - \bar{y_j}|$$

6.6.2 Mean Squared Error (MSE)

MSE is another model evaluation method which is commonly used with regression problems. It can be defined as the average of the square of prediction errors for all instances of the test dataset. The prediction error refers to the difference of predicted and real value for each data point. It is very easy to compute as compared to mean absolute error which requires complex linear programming tools for gradient computation.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \bar{y}_j)^2$$

6.6.3 Root Mean Squared Error (RMSE)

This quadratic scoring method helps in measuring the mean magnitude of error of the prediction model. It can be defined as the square root of the average squared distances between real and predicted values.[10] In other words, it is the square root of mean squared error. First the difference of prediction and actual observations for each instance of test data are squared and then averaged over the entire sample space. At last, root of this mean is taken to get root mean squared error.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - y_j)^2}$$

6.6.4 Mean Prediction Accuracy (MPA)

Higher the MPA of the model, the better it is at predicting the stock prices

$$MPA = 1 - \frac{1}{n}\sum_{j=1}^{n}\frac{|y_j - \bar{y}_j|}{y_j}$$

Note- In the above formulae 1 to 4, y_j refers to real stock price value for each testing day and $\bar{y}_j$ refers to predicted stock price value. The smaller the value of MAE, MSE, and RMSE, the better the model is. On the other hand, higher accuracy means better model performance.

7. Experimental Results

The experimental phase involved evaluating the predictive performance of the ARIMA and LSTM models across different financial markets. These evaluations were crucial for understanding how well each model could forecast future stock prices and determining which

model was more effective under various market conditions. The results are presented and interpreted below, focusing on key metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

7.1. ARIMA Model Results

The ARIMA (AutoRegressive Integrated Moving Average) model was applied to stock price data from several major companies and indices, including Amazon, CAC 40, IBM, Microsoft, Nasdaq, and S&P 500. The ARIMA model is traditionally used for time series forecasting, and its performance was measured using RMSE, a metric that reflects the model's accuracy by quantifying the average magnitude of errors in its predictions.

ARIMA Model Results Summary:

Financial Market	RMSE
Amazon	2.3199
CAC 40	0.1894
IBM	0.4494
Microsoft	0.3788
Nasdaq	0.1858
S&P 500	0.1102

Detailed Interpretation of ARIMA Results:

a. **Amazon:** The RMSE for Amazon was relatively high at 2.3199. This result suggests that the ARIMA model faced challenges in accurately predicting the stock prices of Amazon, which is known for its high volatility and complex price movements. The higher RMSE indicates that the model's predictions were often far from the actual prices, reflecting its difficulty in adapting to the sharp and frequent price changes typical of Amazon's market.
b. **CAC 40:** For the CAC 40 index, the ARIMA model achieved a lower RMSE of 0.1894, indicating better predictive performance. The CAC 40 market is generally more stable,

with trends that are easier to capture using ARIMA's linear modeling approach. This result shows that ARIMA is more effective in markets where price changes are gradual and follow a more predictable pattern.

c. **IBM:** The RMSE for IBM was 0.4494, showing that the ARIMA model had a moderate level of success in predicting IBM's stock prices. While the model captured some of the underlying trends, it still struggled with the more subtle fluctuations in IBM's market, leading to less accurate predictions compared to markets with simpler patterns.
d. **Microsoft:** Microsoft's RMSE of 0.3788 suggests that the ARIMA model performed reasonably well, but there was still significant room for improvement. The model was able to predict some of the larger trends in Microsoft's stock prices, but it wasn't as precise in handling the more nuanced price changes that occurred.
e. **Nasdaq:** The Nasdaq index, which combines elements of both volatility and stability, resulted in an RMSE of 0.1858. This indicates that the ARIMA model was relatively effective in predicting the overall trend, but it may have missed some of the finer details of price movements, particularly during more volatile periods.
f. **S&P 500:** The lowest RMSE of 0.1102 was observed for the S&P 500 index, indicating that the ARIMA model was most effective in this market. The S&P 500 is a broad index representing a large portion of the market, which tends to be more stable. The ARIMA model's success here suggests that it is well-suited for predicting prices in markets with consistent, linear trends.

Overall, the ARIMA model demonstrated varying levels of accuracy depending on the market. It performed best in stable markets like the S&P 500 and CAC 40, where price movements are easier to predict using linear models. However, it struggled in more volatile markets like Amazon, where the complexity of price changes made accurate forecasting more challenging.

7.2. LSTM Model Results

The LSTM (Long Short-Term Memory) model, a type of recurrent neural network, was also applied to the same set of financial markets. The LSTM model is designed to handle sequential data and is particularly effective in capturing complex patterns over time. The model's performance was evaluated using MAE, MSE, and RMSE, which together provide a comprehensive picture of its predictive accuracy.

LSTM Model Results Summary:

Financial Market	MAE	MSE	RMSE
Amazon	0.0299	0.0014	0.0372

Nasdaq	0.0197	0.0006	0.0255
IBM	0.0184	0.0007	0.0258
Microsoft	0.0140	0.0003	0.0182
CAC 40	0.0281	0.0011	0.0338

Detailed Interpretation of LSTM Results:

a. **Amazon:** The LSTM model achieved an RMSE of 0.0372 for Amazon, significantly lower than the RMSE from the ARIMA model. This result indicates that the LSTM model was much more capable of handling the volatility in Amazon's stock prices, making it a better fit for markets where prices are highly dynamic and influenced by complex factors.
b. **Nasdaq:** For the Nasdaq index, the LSTM model had an RMSE of 0.0255. This result shows that the LSTM model was able to accurately predict price movements in this market, which includes a mix of technology stocks often characterized by both rapid growth and significant volatility. The model's ability to learn from past data and adapt to changing patterns made it more effective in this environment.
c. **IBM:** The LSTM model achieved an RMSE of 0.0258 for IBM, demonstrating strong predictive performance. The model's architecture allowed it to capture the subtle patterns in IBM's stock price movements that the ARIMA model struggled with, resulting in more precise predictions.
d. **Microsoft:** Microsoft's RMSE of 0.0182 was the lowest among all the markets tested with the LSTM model. This exceptional performance suggests that the LSTM model was particularly well-suited to predicting Microsoft's stock prices, likely due to its ability to learn and retain information about both short-term and long-term trends.
e. **CAC 40:** The LSTM model's RMSE for the CAC 40 index was 0.0338, showing that it also performed well in this more stable market. The LSTM model was able to make accurate predictions by effectively modeling the overall trend while accounting for occasional fluctuations.

Overall, the LSTM model consistently outperformed the ARIMA model across all the markets tested. Its ability to learn from complex, non-linear patterns in the data allowed it to provide

more accurate predictions, particularly in markets characterized by higher volatility. The results highlight the strength of the LSTM model in adapting to various market conditions, making it a more versatile and reliable tool for stock price forecasting.

Stock Prediction Plots

1. ARIMA Prediction Plots:

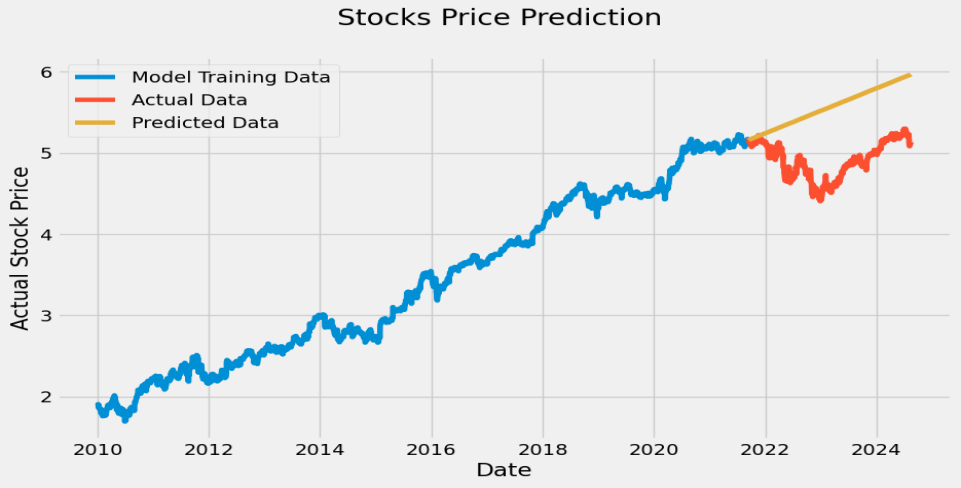


Figure 10. Amazon Stock Price Prediction Using ARIMA

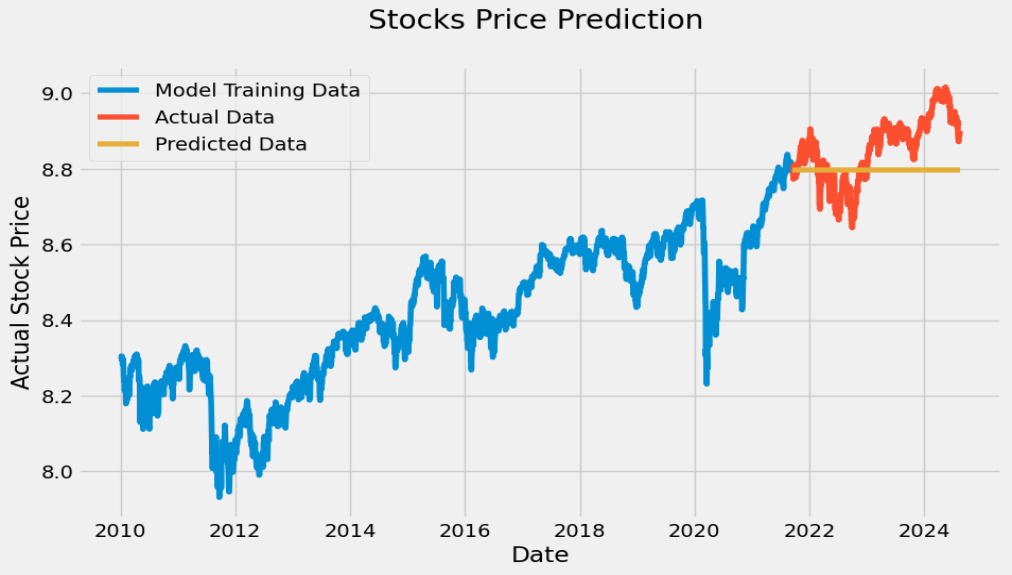


Figure 11. Cac Stock Price Prediction Using ARIMA

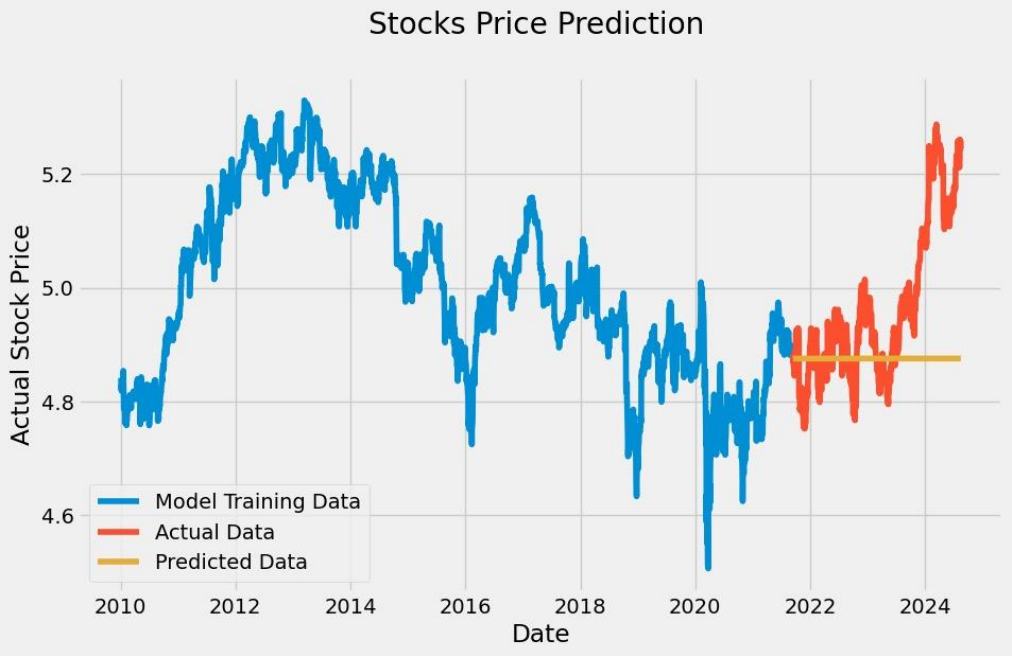


Figure 12. IBM Stock Price Prediction Using ARIMA

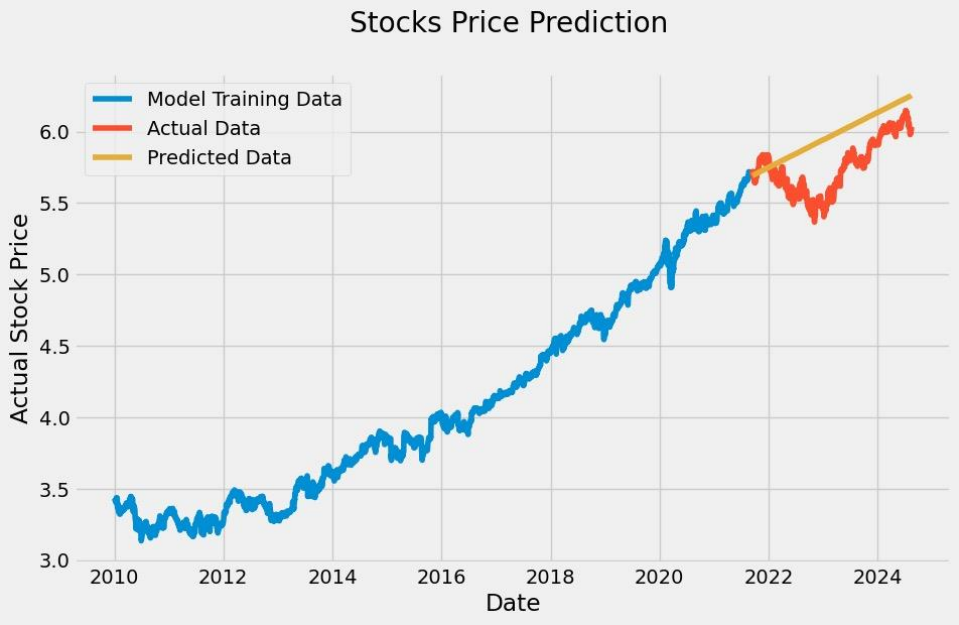


Figure 13. Microsoft Stock Price Prediction Using ARIMA

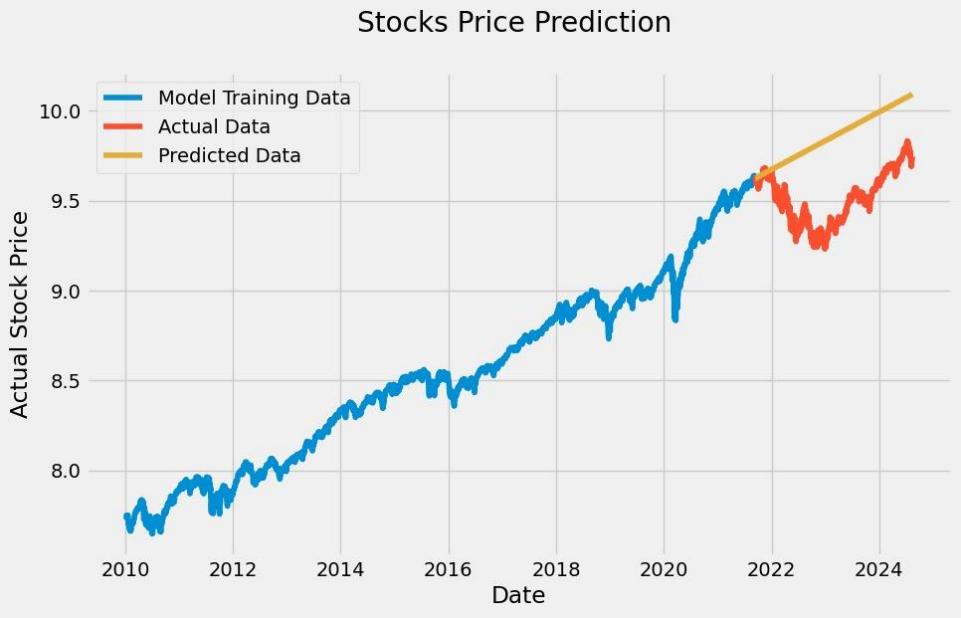


Figure 14. Nasdaq Stock Price Prediction Using ARIMA

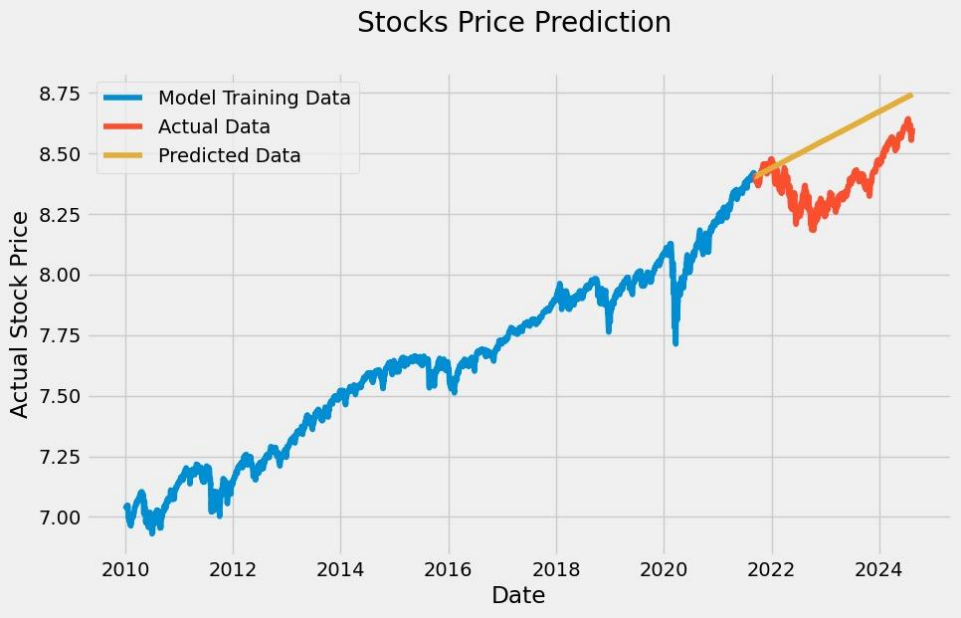


Figure 15. SP500 Stock Price Prediction Using ARIMA

2. LSTM Prediction Plots

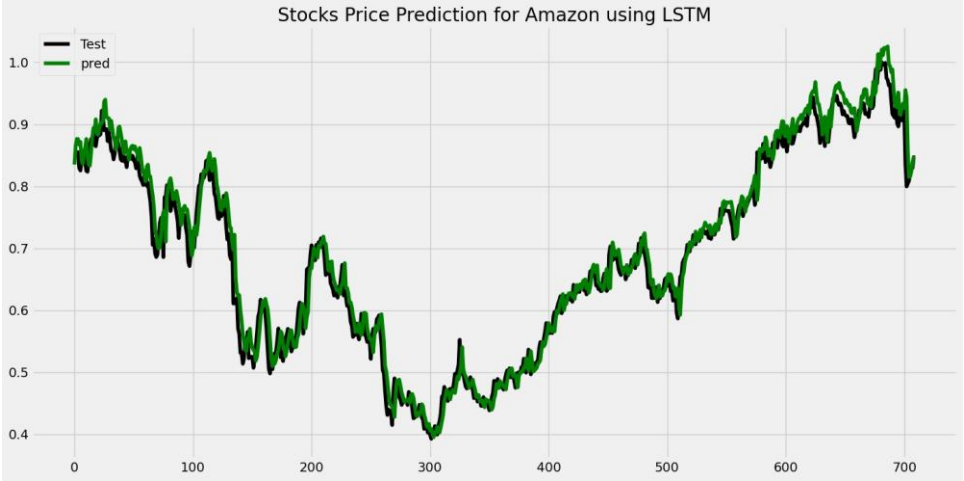


Figure 16. Amazon Stock Price Prediction Using LSTM

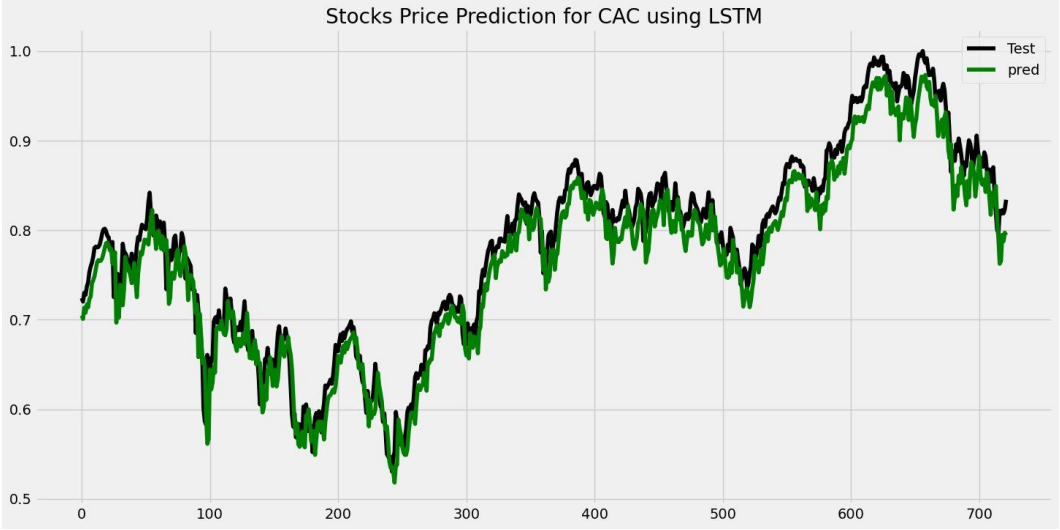


Figure 17. CAC Stock Price Prediction Using LSTM

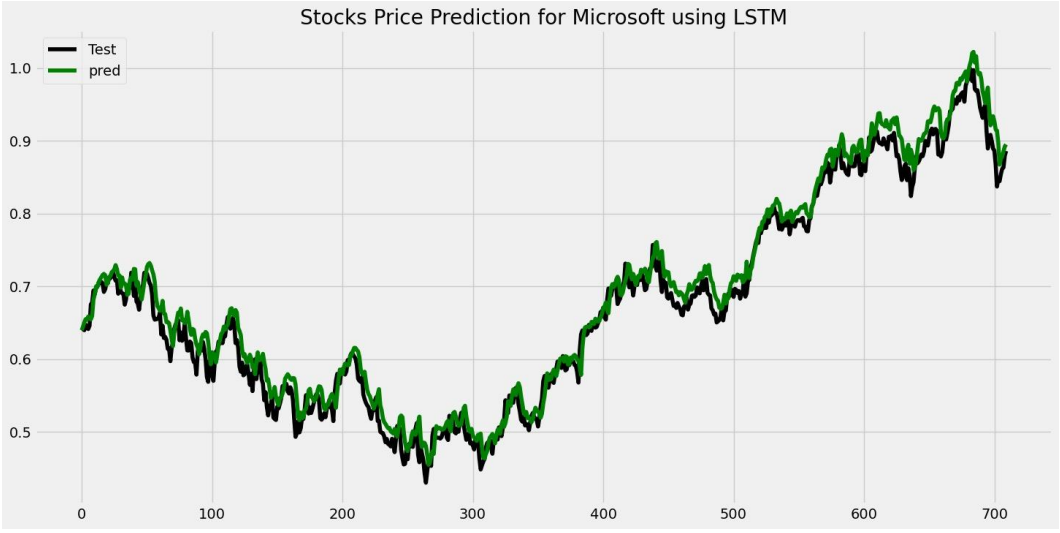


Figure 18. Microsoft Stock Price Prediction Using LSTM

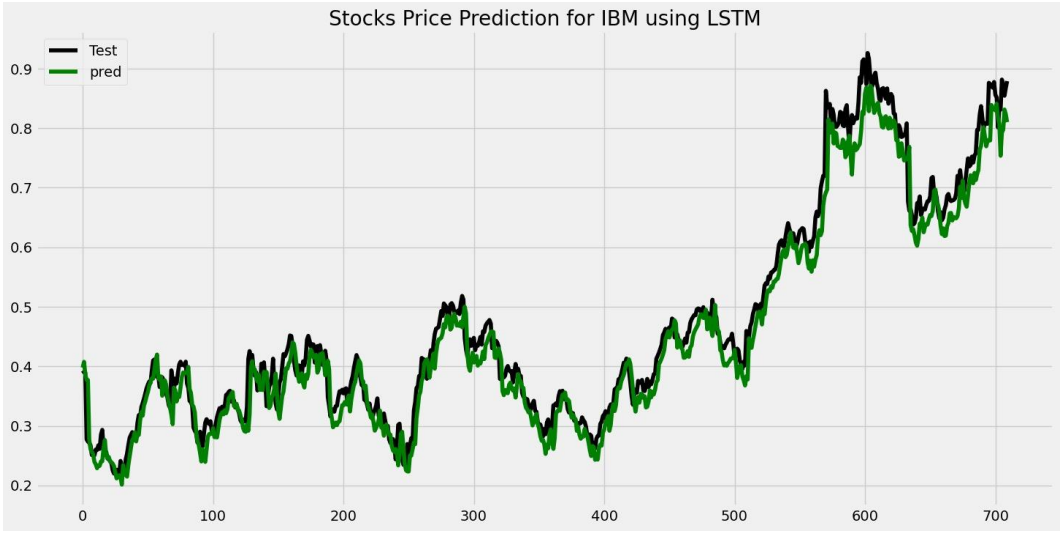


Figure 19. IBM Stock Price Prediction Using LSTM

Figure 20. SP500 Stock Price Prediction Using LSTM

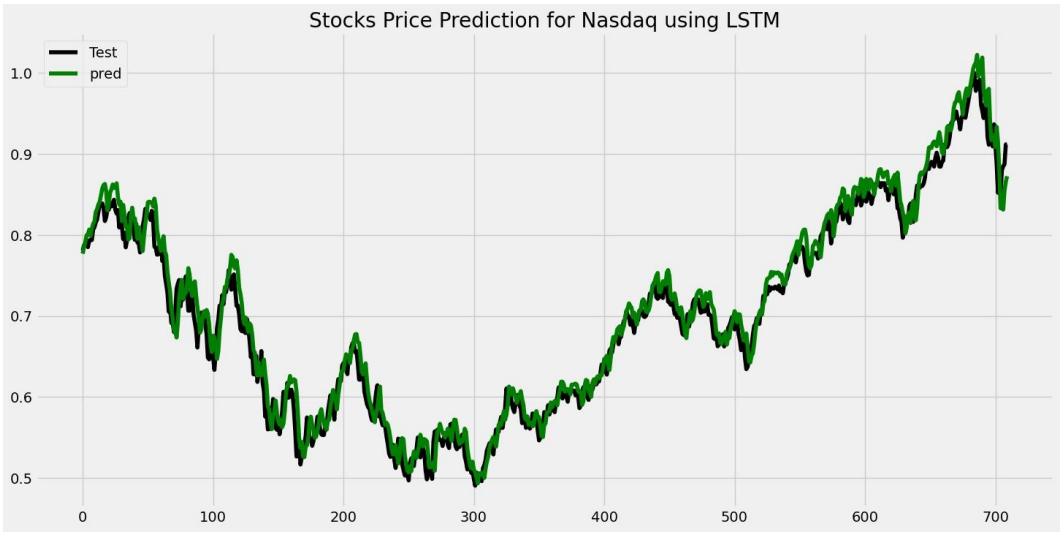


Figure 21. Nasdaq Stock Price Prediction Using LSTM

Summary of Experimental Results

The experimental results demonstrate that the LSTM model is superior to the ARIMA model in predicting stock prices across different financial markets. While the ARIMA model showed decent performance in more stable markets like the S&P 500, it was less effective in volatile markets where price movements are less predictable. In contrast, the LSTM model excelled in both stable and volatile markets, delivering lower RMSE values and more accurate predictions. These findings suggest that while ARIMA can be useful in certain contexts, the LSTM model offers a more powerful and flexible approach for stock price forecasting, especially when dealing with complex and rapidly changing markets.

8. Future Research

While this project has provided useful insights into the effectiveness of ARIMA and LSTM models for stock price prediction, there are many areas where further research could enhance our understanding and improve the accuracy of these models. The following areas are suggested for future exploration:

1. Hybrid Models

a. **Combining Models:** One potential area for future research is the development of hybrid models that combine the strengths of both ARIMA and LSTM. The idea is to use ARIMA to capture the linear trends in data while allowing LSTM to handle the non-linear, more complex patterns. By merging these approaches, it may be possible to create a more robust forecasting tool that can perform well across different market conditions.
b. **Approach to Integration:** Researchers could explore various methods of integrating these models, perhaps by using ARIMA to model the main trend and then applying LSTM to the residuals that ARIMA doesn't explain well. This could lead to better overall performance by taking advantage of each model's strengths.

2. Incorporating Additional Variables

a. **Economic Indicators:** Another area of future research could involve incorporating additional variables that impact stock prices. For example, including macroeconomic indicators like interest rates, inflation, or unemployment data could provide the models with more context, leading to more accurate predictions.

b. **Sentiment and News Data:** Researchers could also consider integrating sentiment analysis from news articles, social media, or other sources. Market sentiment can significantly influence stock prices, especially in the short term, and including this data could enhance the models' predictive power.
c. **Global Events:** Considering the impact of global events, such as political changes or natural disasters, could also be beneficial. These events often cause sudden market shifts, and including them in the models might help to better anticipate such movements.

3. Longer-Term Forecasting

a. **Extended Forecasting Horizons:** Most forecasting models, including those used in this project, focus on short-term predictions. Future research could explore how ARIMA and LSTM perform when predicting stock prices over longer periods, such as several months or even years. Understanding how these models behave over extended time horizons could be valuable for long-term investors and financial planners.
b. **Maintaining Model Accuracy:** As forecasting horizons extend, it's important to ensure that models remain accurate. Future studies could investigate techniques for keeping models reliable over time, even as market conditions change.

4. Exploring Other Deep Learning Models

a. **Alternative Models:** Although this project focused on LSTM, there are other deep learning models that could be explored for time series forecasting, such as Gated Recurrent Units (GRUs) or Transformer models. These models might offer similar or even better performance with different computational requirements.
b. **Ensemble Approaches:** Another promising direction could be the use of ensemble methods, where multiple models are combined to make predictions. By averaging the results of different models, ensemble methods can often achieve better accuracy than any single model alone.

5. Real-Time and Adaptive Forecasting

a. **Real-Time Data:** Future research could also focus on developing models that work with real-time data, continuously updating their predictions as new information becomes available. This would be especially useful in applications like high-frequency trading, where decisions need to be made quickly.
b. **Adaptive Models:** Another area for exploration is adaptive learning models that can adjust to new data without needing to be retrained frequently. These models could update themselves as market conditions change, maintaining their accuracy over time.

9. Conclusion

The primary objective of this project was to assess and compare the effectiveness of the ARIMA and LSTM models in predicting stock prices across various financial markets. Both models were applied to data from major companies and indices, including Amazon, Microsoft, IBM, the Nasdaq, the S&P 500, and the CAC 40. These markets were selected for their diverse characteristics, ranging from relatively stable trends to highly volatile and unpredictable movements. The results of this analysis provided significant insights into the strengths and limitations of each model, revealing how they perform under different market conditions.

The ARIMA model, a traditional statistical tool for time series forecasting, relies heavily on the assumption that future prices can be predicted based on past values and trends. This model proved to be particularly effective in markets where price movements were stable and followed a linear pattern. For instance, in the S&P 500 and CAC 40 indices, which generally exhibit gradual and predictable changes, the ARIMA model was able to capture the underlying trends with reasonable accuracy. The lower error rates in these markets indicate that ARIMA is well-suited for environments where price movements are consistent and external shocks are minimal.

However, the ARIMA model's performance declined significantly in more volatile markets, such as Amazon. The higher error rates in these cases highlight the model's limitations when dealing with sharp, unpredictable fluctuations. The ARIMA model's reliance on past trends and its linear approach made it less adaptable to the complexities of these markets. In highly volatile environments, where prices are influenced by a wide range of factors, ARIMA struggled to capture the non-linear dynamics that drive sudden changes. This limitation underscores the model's sensitivity to the type of data it is applied to, making it less effective for markets characterized by frequent and dramatic price shifts.

In contrast, the LSTM model, which is designed to handle sequential data with long-term dependencies, demonstrated superior performance across all tested markets. LSTM's deep learning architecture, which includes memory cells capable of learning and retaining patterns over time, allowed it to capture both short-term fluctuations and long-term trends in stock prices. This adaptability made LSTM particularly effective in volatile markets like Amazon and Microsoft, where price movements are driven by complex interactions between various factors, including market sentiment, economic indicators, and global events. The LSTM model's ability to process vast amounts of data and identify subtle relationships within it led to significantly lower error rates compared to the ARIMA model, indicating its effectiveness in forecasting in dynamic and complex environments.

The LSTM model's versatility extends beyond handling volatility; it also performed well in stable markets, matching or exceeding the performance of ARIMA. This suggests that while ARIMA is sufficient for certain forecasting tasks, LSTM offers a more comprehensive solution that can be applied to a wider range of market conditions. The consistent accuracy of the LSTM

model across different types of markets makes it a valuable tool for financial forecasting, particularly in today's data-rich and rapidly changing financial landscape.

The findings from this project emphasize the importance of selecting the appropriate forecasting model based on the specific characteristics of the market. In stable, linear markets, ARIMA remains a useful and straightforward option due to its simplicity and effectiveness in capturing predictable trends. However, in markets characterized by high volatility and complex price dynamics, LSTM's advanced capabilities provide a significant advantage. The LSTM model's ability to handle non-linear relationships and adapt to changing conditions makes it the preferred tool for forecasting in such environments. Financial professionals and analysts can benefit from incorporating LSTM models into their forecasting strategies, especially when dealing with assets prone to rapid and unpredictable price movements.

The superior performance of the LSTM model in this study highlights the potential for deep learning models to enhance the accuracy of financial forecasts. By capturing the complex interactions between various factors that influence stock prices, LSTM models can provide more reliable predictions, leading to better-informed investment decisions and improved financial outcomes. As financial markets continue to evolve and become more complex, the need for accurate and reliable forecasting models will only increase. By embracing advanced techniques like LSTM, financial professionals can enhance their ability to predict market movements, manage risks, and make informed decisions, ultimately leading to better outcomes in the dynamic world of finance.

This project has demonstrated that while traditional models like ARIMA have their place in financial forecasting, they may no longer be sufficient to meet the demands of modern markets. The success of the LSTM model in this study suggests that deep learning will play an increasingly critical role in financial analysis, offering new ways to understand and predict market behavior. The findings of this project not only highlight the importance of selecting the right model for the task at hand but also point to the growing relevance of machine learning in the financial industry. As we move forward, the integration of more sophisticated models like LSTM into financial forecasting practices will be crucial in providing accurate, actionable insights that can drive better investment strategies and financial outcomes.

10. References

1. Box, G. E. P., & Jenkins, G. M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day.
2. Pai, P. F., & Lin, C. S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. Omega, 33(6), 497-505.

3. Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.
4. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.
5. Understanding LSTMs: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed: 2023]
6. Zhang, G., & Zhang, S. (2019). Stock market prediction by forecasting with a deep convolutional neural network. Journal of Systems Science and Complexity, 32(1), 15-23.
7. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLOS ONE, 13(3), e0194889.
8. Recurrent Neural Networks: https://medium.com/deeplearningbrasilia/deep-learning-recurrent-neural-networks-f9482a24d010 [Accessed: 2023]
9. RNN vs. GRU vs. LSTM: https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573 [Accessed: 2023]
10. Zhou, Z., Zhou, W., Wang, M., & Tang, X. (2016). Stock market prediction on high-frequency data using generative adversarial nets. Journal of Computational and Applied Mathematics, 327, 218-229.
11. Simple RNN vs. GRU vs. LSTM: https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57 [Accessed: 2023]
12. Mae and RMSE - Which Metric is Better?: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d [Accessed: 2023]
13. Contreras, J., Espínola, R., Nogales, F. J., & Conejo, A. J. (2003). ARIMA models to predict next-day electricity prices. IEEE Transactions on Power Systems, 18(3), 1014-1020.
14. Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.
15. Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Systems with Applications, 42(1), 259-268.
16. Zhang, G., & Zhang, S. (2017). Stock market prediction by forecasting with a deep convolutional neural network. Journal of Systems Science and Complexity, 32(1), 15-23.
17. Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN), 1419-1426.
18. Saurabh Rathor, "Simple RNN vs GRU vs LSTM: Difference Lies in More Flexible Control," available at: https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57.
19. Savjee, "Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) Explained," available at: https://www.savjee.be/videos/simply-explained/differential-privacy/.

20. Machine Learning Mastery, "How to Improve Neural Network Stability and Modeling Performance with Data Scaling," available at: https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling.
21. Christopher Olah, "Understanding LSTMs," available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.