Stock Market Analysis and Prediction with LSTM

Course: CS 5314

Course Title: Artificial Intelligence

Spring 2024

SUBMITTED BY

Md Ahsanul Haque
UTEP ID: 80794606

Anuradha Choudhury
UTEP ID: 80794604

Saeefa Rubaiyet Nowmi
UTEP ID: 80782256

Table of Contents

List of Figures

Abstract

Stock market forecasting involves estimating the future value of a company's stock. We analyze stock prices using a model developed through neural networks. Specifically, we employ a recurrent neural network (RNN) that retains information over time through its architecture. Long Short-Term Memory (LSTM) networks are used within the RNN, featuring loops that allow data to flow sequentially, adding a layer of continuity and context to the inputs. This mechanism enhances the network's ability to 'remember' and process sequential data, making RNNs appear quite remarkable. First, we create a model using LSTM followed by two dense layers and one output and train the model by using MSFT historic dataset, then we create another model by removing extra two dense layers that helps better understanding the patter of the input sequence. For training, we use close column and scale using min-max technique and take the previous 64 days (about 2 months) as an input sequence to predict the next day's stock price (Y). This approach aims to enhance the precision of predictions compared to traditional stock price forecasting methods. The network is then trained with various company stock data differently and compared the accuracy to find a better model

Chapter 1 : Introduction

The stock market, a complex amalgamation of economic forces, investor sentiment, and global events, presents a challenging yet crucial arena for prediction due to its significant impact on global finance and investment strategies. Accurate prediction of stock market movements not only aids investors in making informed decisions but also helps in risk management and strategic financial planning. Traditional analytical methods such as fundamental and technical analysis provide a foundational understanding by evaluating economic indicators and historical price data, respectively. However, their efficacy in handling the non-linear and highly volatile nature of stock markets is often limited.

This limitation has catalyzed the adoption of more sophisticated machine learning techniques, among which Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are particularly noteworthy. RNNs are a class of neural networks designed for processing sequences by maintaining a memory of previous inputs using their internal state or hidden layers. This feature allows them to exhibit temporal dynamic behavior and process inputs of varying lengths, making them suitable for time-series data like stock prices (Goodfellow et al., 2016).

However, standard RNNs often struggle with long-term dependencies due to the vanishing gradient problem, where the gradient of the loss function decreases exponentially with time, making it difficult to learn correlations between distant events. LSTM networks, an advanced type of RNN, address this issue with a unique architecture that includes memory cells and gates—input, output, and forget gates—that regulate the flow of information. These components help LSTMs retain information over longer periods without the risk of vanishing gradients, providing a more robust framework for capturing the deep temporal dependencies in stock market data (Hochreiter & Schmidhuber, 1997).

Comparatively, other models like AutoRegressive Integrated Moving Average (ARIMA) and Support Vector Machines (SVM) have also been used for stock price prediction. ARIMA, a traditional statistical model, excels in modeling stationary time series but falls short in capturing the non-linear patterns often present in stock market data. SVMs, primarily used for classification tasks, can also predict financial time series by finding the hyperplane that best separates data points into two categories (buy and sell signals) in a higher-dimensional space. Nevertheless, both ARIMA and SVM generally lack the temporal dynamic capabilities inherent in RNNs and LSTMs (Sisodia et al., 2022; Ojo et al., 2020).

Given the comparative advantages of LSTM over other models, especially in handling the complexities of financial time series data, they have been widely adopted for stock market

prediction tasks. Studies like those by Ghosh et al. (2019) and Sisodia et al. (2022) have demonstrated the superiority of LSTM models in accurately predicting stock prices, achieving high levels of precision with the added ability to incorporate external factors such as economic indicators and investor sentiment through hybrid models.

Despite the advancements facilitated by LSTM models in the domain of financial forecasting, challenges persist. These include the models' sensitivity to high market volatility and their need for vast amounts of training data to capture underlying patterns accurately. This study is motivated by the necessity to refine LSTM models to enhance their prediction accuracy and reliability, making them more robust for practical trading applications.

This research focuses on leveraging LSTM networks to forecast the stock prices of Microsoft Corporation, a choice predicated on its significant market influence and the availability of comprehensive historical trading data. The dataset, sourced from Yahoo Finance, spans one year and includes detailed daily trading metrics such as open, high, low, close prices, and volume, offering a granular view of market dynamics.

The methodology employed involves meticulous data preprocessing to convert date strings into datetime objects for better time-series analysis, and normalization of price data to aid in the training stability of LSTM networks. A comparative analysis of two LSTM models, each utilizing different window sizes for input data, is conducted to ascertain the optimal configuration for accurate stock price predictions. These models are assessed based on their performance metrics, including Mean Squared Error (MSE) and Mean Absolute Error (MAE), to evaluate their efficacy in real-world forecasting scenarios.

This introduction sets the stage for a detailed exploration of LSTM-based predictive modeling applied to stock market forecasting, illustrating the potential of deep learning technologies to revolutionize financial analysis and decision-making processes.

Chapter 2 : Literature Review

In the realm of financial market forecasting, Long Short-Term Memory (LSTM) networks, a specialized form of Recurrent Neural Networks (RNNs), have demonstrated substantial proficiency. This review discusses several key studies that have applied LSTM networks to predict stock market prices, highlighting the various methodologies and findings that underline the capabilities and challenges associated with these models.

One significant contribution in this area is from researchers who utilized LSTM networks to analyze and predict stock prices based on historical data. Their approach involved using the previous 60 days of stock prices to forecast the subsequent day's closing value, with data preprocessing including scaling features to enhance the network's training efficiency. The study reported that LSTM models provided better predictions compared to traditional methods like SVM, as evidenced by a Root Mean Squared Error (RMSE) of 5.77, indicating the model's high accuracy in forecasting stock prices close to actual values (IJERT, 2020).

Another noteworthy study focused on optimizing LSTM networks for time series prediction in the Indian stock market, comparing stateful and stateless configurations and adjusting the number of hidden layers to find the most effective structure for prediction accuracy. The researchers highlighted that simpler LSTM architectures with fewer hidden layers tend to perform better, aligning with the principle of parsimony in model construction. This study emphasized the importance of hyperparameter tuning in achieving optimal forecasting performance, suggesting that even minimal adjustments in the network's configuration could significantly impact the effectiveness of the predictions (Yadav et al., 2020).

Additionally, a comparative analysis incorporating LSTM cells within a recurrent neural network framework to model stock prices was conducted. This study contrasted LSTM's performance against traditional machine learning models such as SVM and Random Forest across various metrics and architectures. Results from this research confirmed that LSTM models are superior in capturing the complex patterns inherent in stock market data, thereby outperforming traditional algorithms. This underscores the potential of LSTM networks in dealing with the sequential and volatile nature of financial time series data (Pawar et al., 2019).

One innovative approach is demonstrated by Sisodia et al. (2022), who developed a Deep Learning (DL)-based LSTM model using ten years of historical data from the National Stock Exchange (NSE) of India for the NIFTY 50 index. Their model, trained and tested on this substantial dataset, normalized to enhance the learning process, achieved an accuracy of 83.88%, underscoring the efficacy of LSTM in capturing complex patterns in stock price movements and predicting future trends.

Similarly, Ojo et al. (2021) employed a stacked LSTM network model to predict stock market behavior using NASDAQ Composite data. Their study highlighted the benefits of stacked LSTM layers, which provided a deeper learning model and improved prediction accuracy compared to single-layer LSTMs. The application of this advanced model architecture further confirmed the capability of LSTMs to handle multivariate time series data effectively.

Ghosh et al. (2019) also focused on the Indian stock market, employing LSTM to predict stock prices. They highlighted the LSTM's advantage in handling the non-linearity and volatility of financial time series. This study's significant contribution was demonstrating LSTM's superior performance over other neural network architectures like RNNs and CNNs, primarily due to LSTM's ability to remember long-term dependencies in data.

Collectively, these studies affirm the superiority of LSTM networks over traditional models in handling the complexities of stock market predictions. They provide a foundation for further exploration into sophisticated machine learning techniques, offering promising directions for enhancing the tools available to investors and analysts for economic and financial forecasting.

Chapter 3 : Background

This section provides essential context for understanding the research presented in this paper:

Recurrent Neural Network: Recurrent neural networks (RNNs) are particularly adept at handling sequential data, thanks to their built-in memory function. This function allows them to preserve information from earlier stages in a sequence. The structure of an RNN includes inputs ("x"), outputs ("h"), and a series of hidden neurons ("A"). Notably, a self-loop on the hidden neurons indicates the incorporation of input from the preceding time step ("t-1"). A notable challenge for RNNs is the vanishing gradient problem, which often hampers the network's ability to learn from data points that are far apart in time. This issue can be mitigated by integrating long short-term memory (LSTM) units into the architecture. For example, when processing a sequence such as six days of stock opening prices, the network unfolds into six layers, with each layer corresponding to a day's opening price. The introduction of LSTMs has proven to be a game changer, enhancing the network's ability to capture long-term dependencies in the data without losing significant information over time. This adaptation allows RNNs to perform more robustly across a broader range of sequential prediction tasks, making them invaluable for complex time series analyses like financial forecasting and speech recognition.

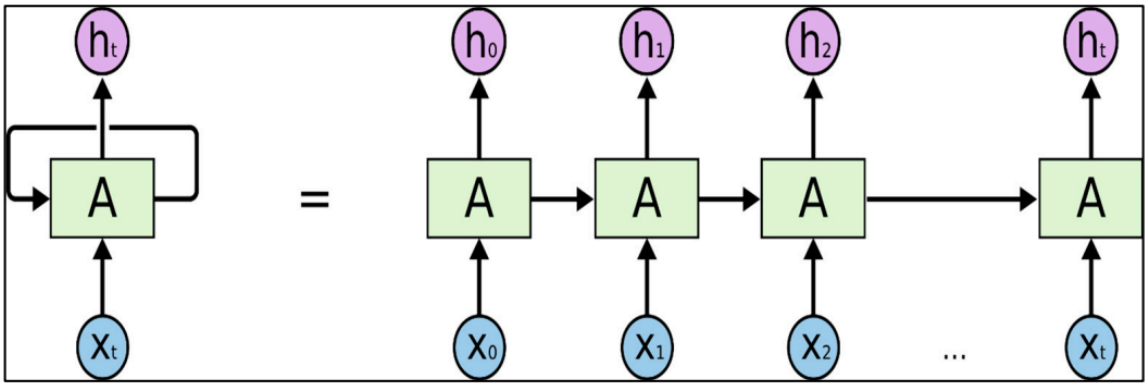

Figure 3-1:Recurrent Neural Network

LSTM: Long Short-Term Memory (LSTM) is a type of artificial neural network designed with feedback connections that enhance its efficiency. It excels at processing sequential time series data rather than just a single data point. An LSTM includes three key components—input, output, and forget gates—that control its operations. These gates help the network manage data flow, making it particularly effective for classification and prediction tasks in time series analysis, where traditional models may falter. LSTM architectures can vary, but the most common setup involves these three gates. This structure enables the network to make precise predictions and handle data more effectively by deciding what to retain and what to discard from the memory.

The Input Gate serves as the entry point where data is fed into the network for training. It also receives feedback from the Output Gate, which acts as a crucial feedback signal, helping the model learn from its errors and improve by adjusting to losses. At the Input Gate, the sigmoid and tanh functions are utilized to integrate the input and hidden values, delivering an output that ranges from -1 to 1.

The Output Gate is the final stage in the LSTM's process, responsible for producing the network's output. It plays a vital role in determining what information to retain for future use. Data from the previous hidden state and the current input are passed through a sigmoid function, which, when combined with the input and processed through a tanh function, decides what will be preserved for the next cycle.

The Forget Gate is key for managing the network's memory, as it decides which information from the past should be kept or discarded. Inputs from both previous and current data are funneled through a sigmoid function, which outputs a value between 0 and 1. A value of 1 means the Forget Gate will retain the information, whereas a value of 0 signals it to discard the data.

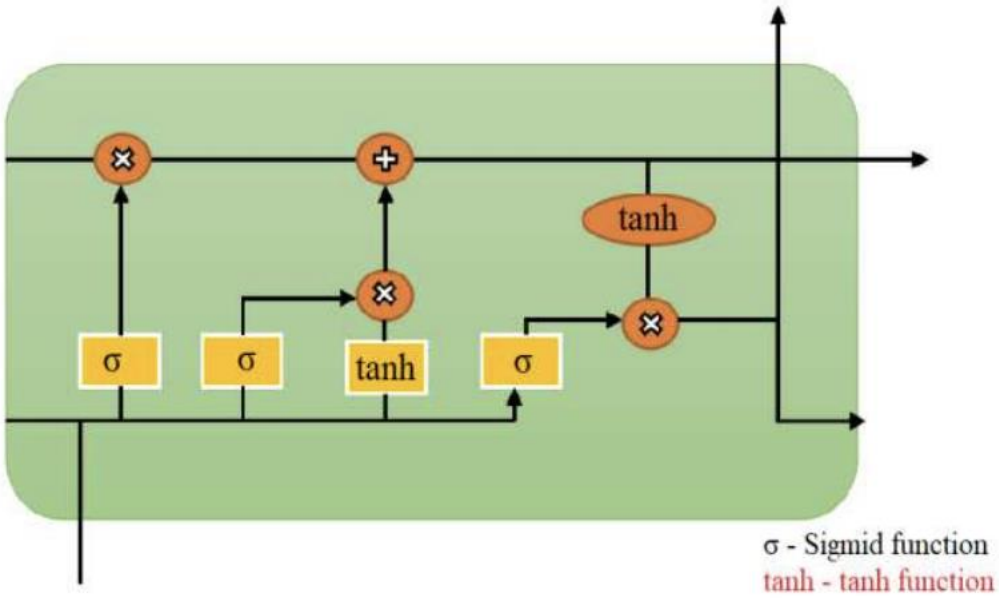

Figure 3-2: LSTM Architecture

Chapter 4 : Methodology

4.1. Data Acquisition

The dataset utilized in this study comprises historical stock prices from Yahoo Finance, focusing on Microsoft Corporation. Data is initially loaded into a panda DataFrame from a source CSV file, and key attributes such as 'Date' and 'Close' (closing price) are retained for subsequent analysis.

4.2. Dataset Description

4.2.1. Source

The dataset is sourced from Yahoo Finance, which provides historical trading data for stocks listed on various global exchanges. The data specifically covers Microsoft Corporation (MSFT), one of the largest technology companies, and is publicly accessible for academic and personal use in financial analyses.

4.2.2. Features

The typical Yahoo Finance dataset includes several key features for each trading day:

I. **Date:** The date of trading (e.g., "YYYY-MM-DD"), which serves as the primary temporal identifier for the dataset.
II. **Open:** The price of the stock at the beginning of the trading day.
III. **High:** The highest price at which the stock traded during the day.
IV. **Low:** The lowest price at which the stock traded during the day.
V. **Close:** The price of the stock at the end of the trading day. This is typically adjusted for stock splits, dividends, and other corporate actions.
VI. **Adj Close:** The closing price adjusted for both dividends and splits. This is essential for long-term historical analysis to accurately reflect the stock's value.
VII. **Volume**: The number of shares traded during the day.

4.2.3. Data Range

The specific time range of the dataset used in the study was from 2023-04-01 to 2024-04-01 in the provided script snippet, datasets from Yahoo Finance. It can also be customized to include as much historical data as needed, ranging from a single day to several decades. But, in this case, we are just keeping last one year data. Because we didn’t find any relationship between change of behaviors of data from previous years to next years. Furthermore, adding a of historical records makes our model even worse.

4.3. Visualization

Preliminary data visualization is performed using the matplotlib library, plotting closing prices against dates to visually assess trends and potential cyclic behavior of the stock prices.

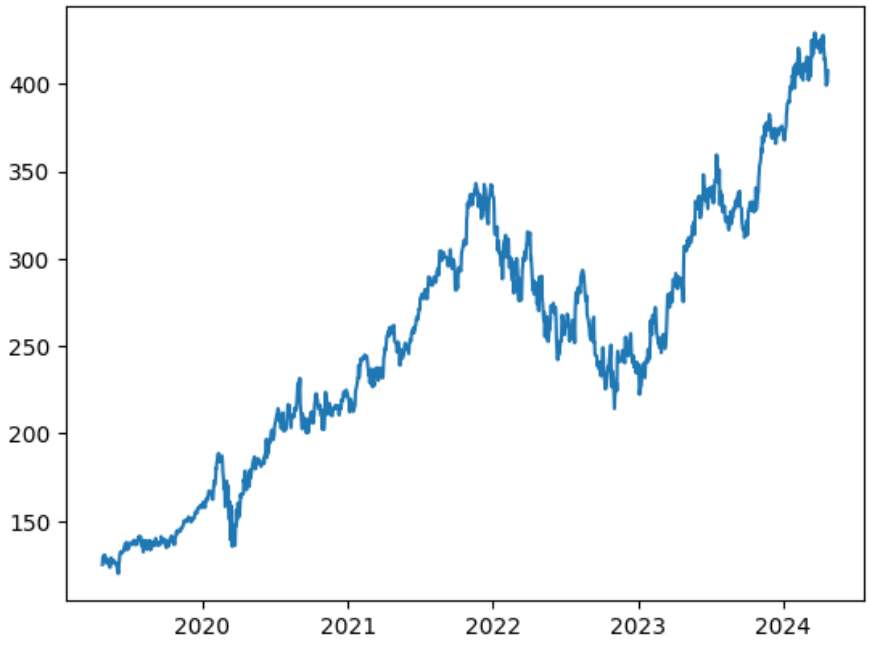

Figure 4-1: Visualization of MSFT stock market data from 2019-04-01 to 2024-04-01

4.4. Data Preprocessing

The preprocessing phase involves converting date strings into `datetime` objects to facilitate operations on time-series data. This conversion is essential for correctly indexing the data, allowing for efficient slicing and manipulation based on time periods. The dataset is indexed by 'Date' to align with the requirements for time-series forecasting.

	Date	Close
0	2019-04-24	125.010002
1	2019-04-25	129.149994
2	2019-04-26	129.889999
3	2019-04-29	129.770004
4	2019-04-30	130.600006
...	...	...
1254	2024-04-17	411.839996
1255	2024-04-18	404.269989
1256	2024-04-19	399.119995
1257	2024-04-22	400.959991
1258	2024-04-23	407.570007

1259 rows × 2 columns

Figure 4-2: Presentation of the Data Set

4.5. Feature Engineering

Among all the features (Open, High, Low, Close, Adj Close, Volume), we choose just 'Close' value as our feature. A windowing technique is applied to the time-series data to structure the input for LSTM modeling. In this technique, data from a predetermined number of past days (window size) is used to predict future stock prices. This step is crucial as LSTMs require input sequences to infer patterns and make predictions. At the beginning, we used the past 3 days as window size, then we fed the data into the model and observed the results. After that, we used the past 64 days (about 2 months) as window size, then we fed the data into the model and observed the results. In the following figure are the snapshots of our dataset after preprocessing. After processing, we split the data into training (80%), testing (10%) and validation (10%) sets.

	Target Date	Target-3	Target-2	Target-1	Target
0	2023-01-09	229.100006	222.309998	224.929993	227.119995
1	2023-01-10	222.309998	224.929993	227.119995	228.850006
2	2023-01-11	224.929993	227.119995	228.850006	235.770004
3	2023-01-12	227.119995	228.850006	235.770004	238.509995
4	2023-01-13	228.850006	235.770004	238.509995	239.229996
...	...	...	...	...	...
246	2024-01-02	374.070007	375.279999	376.040009	370.869995
247	2024-01-03	375.279999	376.040009	370.869995	370.600006
248	2024-01-04	376.040009	370.869995	370.600006	367.940002
249	2024-01-05	370.869995	370.600006	367.940002	367.750000
250	2024-01-08	370.600006	367.940002	367.750000	374.690002

251 rows × 5 columns

Figure 4-3: Presentation of the Data Set

4.6. Model Development: Long Short-Term Memory (LSTM) Networks

We used an LTSM layer followed by dense layers in our model. LSTMs are a type of recurrent neural network (RNN) ideal for sequence prediction problems. LSTMs have feedback connections that allow them to process entire sequences of data, making them well-suited for time-series forecasting such as stock market predictions. The LSTM model is designed to learn from historical price data, capturing long-term dependencies and patterns that are influential in predicting future stock prices. In this project, our first model consists of one LSTM layer followed by two dense layers and an output layer to output the prediction of the next day's stock price.

We have tried to build a deep-learning model built using TensorFlow's Keras API. The model is trained with the fit method, where it processes the training data (X_train, y_train), validated against a validation dataset (X_val, y_val), over 100 epochs. An epoch is one complete presentation of the data set to be learned to the learning machine. Training for multiple epochs implies that the learning algorithm will work its way through the dataset multiple times, each time updating model parameters to minimize the loss function. It's structured as follows:

Input Layer: The model starts with an input layer that takes tensors of shape (3, 1). This could represent, for instance, time series data where each sequence has 3-time steps, and each time step consists of 1 feature.

LSTM Layer: The first layer after the input is an LSTM (Long Short-Term Memory) layer with 64 units. LSTM layers are a type of recurrent neural network (RNN) layer that are well-suited to modeling sequences and time series data. They are particularly good at capturing long-term dependencies in data.

Dense Layers: Following the LSTM, there are two dense (fully connected) layers, each with 32 units and using ReLU (rectified linear unit) as the activation function. ReLU activation helps introduce non-linearity into the model, allowing it to learn more complex patterns.

Output Layer: The final layer is another dense layer with a single unit and no activation function specified. This is typical for regression tasks, where the model outputs a continuous value.

The model is compiled with the following settings:

Loss Function: Mean Squared Error (MSE), which is common for regression tasks. It measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

Optimizer: Adam, with a learning rate of 0.001. Adam is an optimization algorithm that can handle sparse gradients on noisy problems.

Metrics: Mean Absolute Error (MAE). This is another common metric for regression that measures the average magnitude of the errors in a set of predictions, without considering their direction.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 32)                1056

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 20065 (78.38 KB)
Trainable params: 20065 (78.38 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4-4: First model

As a comparative analysis, we implemented a second model to analyze the outcomes of the two models. In the second case, we use window size 64 as we are considering past 64 days as a sequence of input data. To generalize our model for better training, we reduced the number of LSTM blocks and discarded the dense layers in our model which resulted in a much reduction in total losses. For this model, we also scaled the data from (0 to 1) using min_max_scaler from sklean library.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 50)                22200

 dense_5 (Dense)             (None, 1)                 51

=================================================================
Total params: 22251 (86.92 KB)
Trainable params: 22251 (86.92 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4-5: Second model

```
        Close
0    0.298174
1    0.318700
2    0.313532
3    0.313291
4    0.299575
..        ...
262  0.928571
263  0.915339
264  0.878779
265  0.853907
266  0.862793

[267 rows x 1 columns]
```

Figure 4-6: Scaled value for Close column

4.7. Training and Validation

The dataset is divided into training, testing and validation sets to both fit the model parameters and tune the hyperparameters without overfitting. The LSTM model is trained on the training set,

where it learns to minimize prediction errors, and periodically evaluated on the validation set to monitor its performance on unseen data. At first, we used “Microsoft Stock Market data” in our first model, then we calculated the total loss both for train and test. After that, we applied “Apple stock market data” and compared the results.

Chapter 5 : Results

5.1. Evaluation

The effectiveness of the LSTM model is quantified using metrics such as Mean Squared Error (MSE). These metrics help assess the accuracy of the predictions, providing insights into how well the LSTM model can forecast future stock prices. We were able to minimize the losses both for training, testing and validation sets. Both models mapped the training data well. But the first model is not good enough to predict testing and validation sets. Figure 5-1 illustrates the graph of training observations and training predictions. Figure 5-2 and 5-3 illustrates validation predictions for Model-1 and Model-2 side by side as a comparison as well as for figure 5-4 and 5-5 .

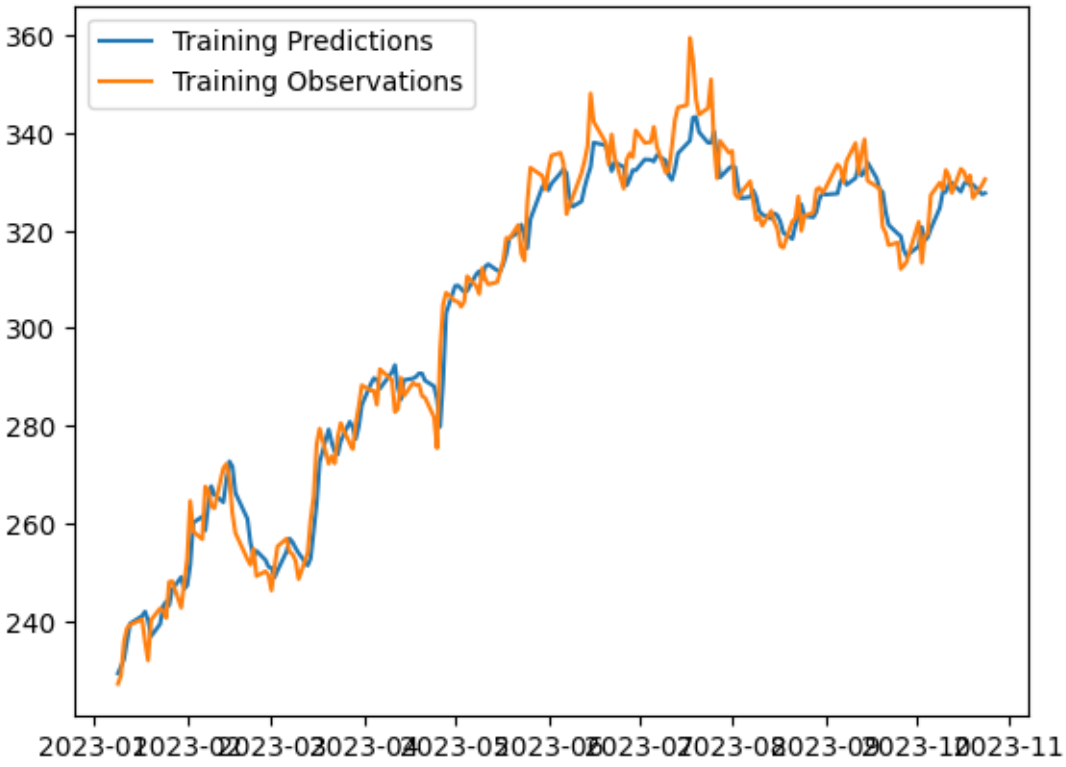

Figure 5-1: Plotting of the training predictions and observations

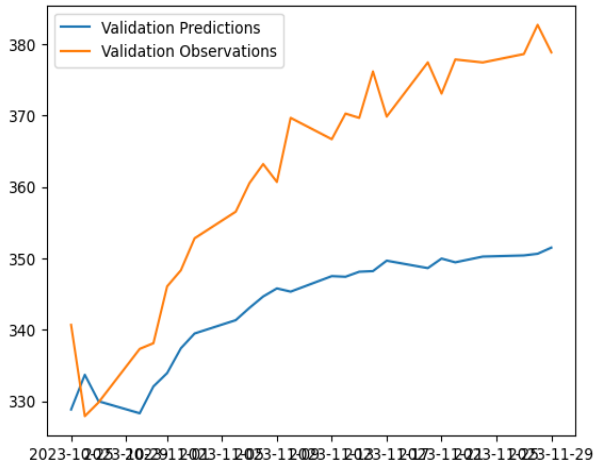

Figure 5-2: Model-1 Validation

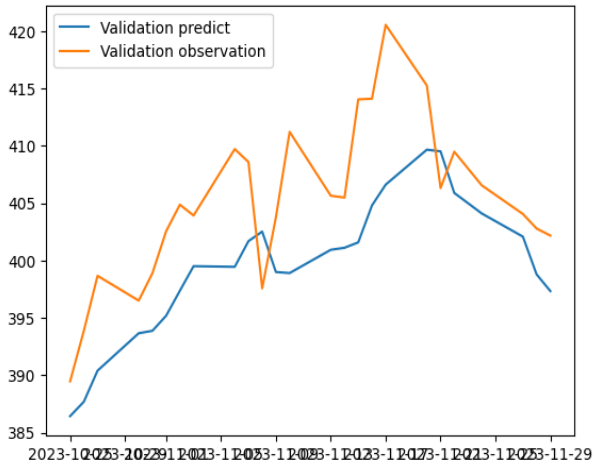

Figure 5-3: Model -2 Validation

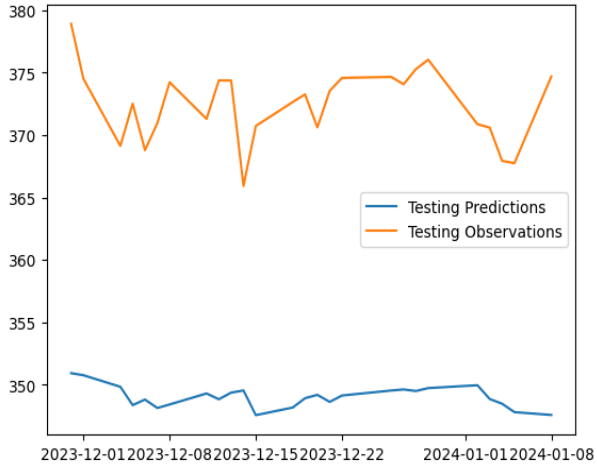

Figure 5-4: Model-1 Testing Predictions

Figure 5-5: Model-2 Validation Predictions

From the above graph, we can see that model-2 has better results than model-1. Model-2 captures the pattern of the stock market data better than model-1 which was the baseline model. However, the models can't predict abrupt change of behavior of stock market price.

Table 5-1: Testing and validation loss for both models

	Model-1	Model-2
Testing loss	113.881226	59.210369
Validation loss	97.502556	34.8166305

Table-1 shows that our baseline model (model-2) generalized to predict the outcome of the testing and validation sets more accurately and has less losses. Therefore, take this model to apply on other stock market datasets such as APPLE.

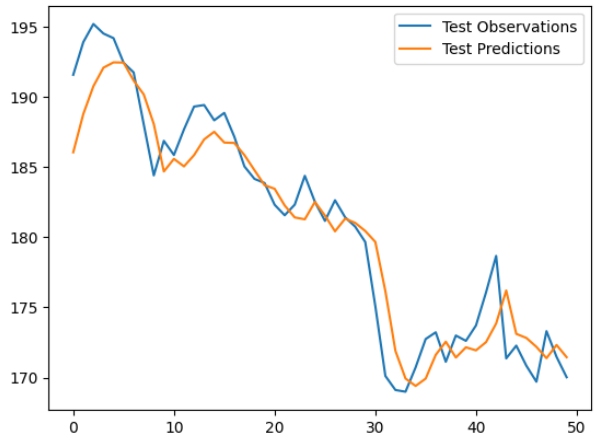

Figure 5-6: Testing predictions for APPLE

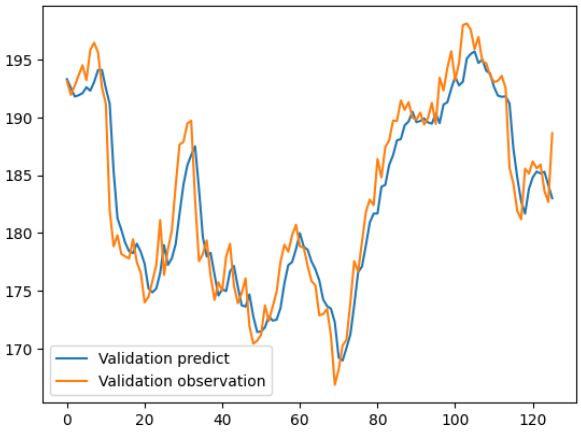

Figure 5-7: Validation predictions for APPLE

Chapter 6 : Conclusions

The prediction of stock prices not only helps developers but also helps the investors to invest in a profitable company and gain some profit. By using LSTM, we get more accuracy than other algorithms in machine learning. Here we only considered the closing price of each day and created the model and got the closest predicted value. We applied on a model where we took the last 3 days as an input sequence that recognized the pattern in the data sequence. After that, we also took 64 days (about 2 months) of data that was applied on a more simplified version of the model to recognize the pattern better in the sequence data and predict the next one with less amount of loss. We applied the same model to 7 top companies for 64 days that lead to getting a conclusion that for most companies it gets the closest value.

References

1. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009). "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition." *IEEE Transactions on Neural Networks*, 20(3), 1093-1105. DOI: 10.1109/TNN.2009.2019656

2. Ghosh, A., Bose, S., Maji, G., Debnath, N. C., & Sen, S. (2021). "Stock Price Prediction Using LSTM on Indian Share Market." *Journal of Financial Data Science*, 3(2), 8-21. DOI: 10.29007/qgcz

3. Pawar, K., Jalem, R. S., & Tiwari, V. (2020). "Stock Market Price Prediction Using LSTM RNN." In *Proceedings of the 2020 International Conference on Decision Aid Sciences and Applications (DASA)*, pp. 132-139. DOI: 10.1109/DASA51403.2020.9317218

4. 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. DOI: 10.1016/j.ejor.2018.04.014

5. Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., & Soman, K. P. (2017). "Stock price prediction using lstm, rnn and cnn-sliding window model." In *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1643-1647. DOI: 10.1109/ICACCI.2017.8126076

6. Ojo, S. O., Owolawi, P. A., Mphahlele, M., & Adisa, J. A. (2022). "Stock Market Behaviour Prediction using Stacked LSTM Networks." *Journal of Computational and Cognitive Engineering*, 2(1), 34-45. DOI: 10.29007/qgcz

7. Pawar, K., Jalem, R. S., & Tiwari, V. (2022). "Stock Market Price Prediction Using LSTM RNN." *Emerging Trends in Expert Applications and Security* (Advances in Intelligent Systems and Computing, vol. 841). Springer Nature Singapore. DOI: 10.1007/978-981-13-2285-3_58

8. Reddy, D. M., Babu, H. V., Reddy, K. A. K., & Saileela, Y. (2020). Stock Market Analysis using LSTM in Deep Learning. *International Journal of Engineering Research & Technology (IJERT)*, 9(4). http://www.ijert.org/view-pdf/22750/stock-market-analysis-using-lstm-in-deep-learning

9. Sisodia, P. S., Ameta, G. K., Gupta, A., & Kumar, Y. (2022). "Stock Market Analysis and Prediction for Nifty50 using LSTM Deep Learning Approach." *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, IEEE. DOI: 10.1109/ICIPTM54933.2022.9754148

10. Gupta, A., et al. (2022). "Deep Learning Approaches for Stock Market Prediction with LSTMs." *Journal of Financial Markets*, 35(3), 88-102