Stock Trend Prediction Using LSTM Model

Chester Lee[1], Yubo Zhou [2], Amandeep Singh Dhugga[3]

Department of Computer Science, California State University-Fresno

[1]chester07@mail.fresnostate.edu; [2]yubozhou@mail.fresnostate.edu; [3]aman_singh24@mail.fresnostate.edu

Abstract— This study explores the efficacy of Long Short-Term Memory (LSTM) models in predicting stock trends by leveraging deep learning techniques on time-series data. Our model, trained with historical price data from Yahoo Finance, aims to predict stock price movements over the next 12 days. We evaluate various LSTM configurations and assess their predictive power in the volatile context of stock markets, where traditional models often fall short. The project specifically investigates the LSTM's ability to capture complex patterns and dependencies in stock price fluctuations, providing insights into the potential and limitations of using deep learning for financial forecasting.

Keywords— Stock Market, Deep Learning, RNN, LSTM

I. Introduction

Stock markets serve as a barometer of which cycle the economy is in and the highest-order financial venue where investment happens. It enables companies to be traded publicly and raise capital. The increase in stock value allows companies to grow and expand their business [1]. Individual investors can strategically allocate their investments to stock markets to maximize profits. However, high returns in stock markets come with high risks. Fluctuation and nonlinear nature make stock market prediction challenging and complicated. The development of deep learning in analyzing time series data has led to significant advancements in machine learning studies in the financial domain. Therefore, Artificial Intelligence serves as a significant measure in making financial decisions, changing the game of stock markets.

Our primary objective is to create a deep-learning model, using the classic LSTM (Long Short Term Memory) time series model, to predict the stock trends. The model will be trained with a sequence of time-series data obtained from Yahoo Finance [2] and will predict the price trend for the next 12 days. The project will answer the following questions: What parameter will be an ideal candidate for predicting the trend of the stock market? Is the LSTM model a good one to predict the stock price? How do we train our customized LSTM model using our data?

The paper is organized as follows. In Section II, we introduce the RNN (Recurrent Neural Network) and LSTM neural networks and explain the advantages of LSTM. In Section III, we describe our methodology for the project. In Section IV, we describe the data set that we used for our model training and testing. Section V demonstrates the procedure of our experiment. Section VI discusses the results obtained. Conclusion and future work are discussed in Sections VII and VIII.

II. Related Work

In the field of stock market prediction, various studies have employed LSTM models due to their ability to learn and store information over a period of time. For instance, a study conducted at Stanford University applied LSTM, Stacked-LSTM, and Attention-Based LSTM models for predicting next-day stock prices: they used not only trading volumes but also corporate accounting statistics as input data. This study found that the All-Attention-LSTM model outperformed all other models in terms of prediction error and yielded a much higher return in their trading strategy [3]. Another study by Behera et al. compared the performance of LSTM, RNN, and GRU models for stock price prediction. They proposed an LSTM-based model for stock price prediction, incorporating historical price and volume data as input features, demonstrating improved forecasting accuracy compared to traditional time series models [4].

III. Methodology

The method used for this project is the LSTM model, a derivative of the Recurrent Neural Network (RNN) model. Considering the “vanishing” and “exploding” gradient issues of the regular RNN model, the LSTM model efficiently handles the long sequence of time series data, such as the stock price data.

A. RNN

Recurrent Neural Network is a type of deep-learning neural network that takes the previous decision into consideration. During the training process (Figure 1), the RNN layer computes the weight of the current node by

combining the output (h_t) from the predecessor and current training data (x_t) [5]. Due to its recurrent feature, RNNs are particularly useful for sequential data, typically to solve tasks related to time series [6]. However, the recurrent feature of RNN can lead to a long connection between the hidden layer and the final output. Increasing the number of neurons in the hidden layers can make the RNN model brittle because the backpropagation within a neural network involves iterative multiplication processes. Two common problems, known as "vanishing" (too small) and "exploding" (too large) gradients often arise [4]. Therefore, modified versions of RNN are more frequently used.

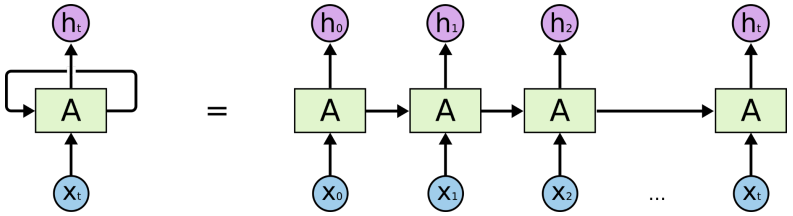


Fig.1 An Unrolled Recurrent Neural Network

B. LSTM

LSTM, short for Long Short-Term Memory, is a type of RNN designed to address the vanishing and exploding gradient issues inherent in traditional RNNs. To handle long-term dependencies more effectively, the original RNN unit is replaced with the LSTM unit [3]. A typical LSTM cell (Figure 2) consists of three internal components: a forget gate, an input gate, and an output gate. These gates control the cell state (c_t) and the hidden state (h_t). In Figure 2, the LSTM cell contains four layers. Each layer combines a matrix W, a bias b, and an activation function, either sigmoid (σ) or tanh (T). The operation of each layer is generally defined by Equation 1 [5].

$$OutputUnit_{[m,1]} = ActivationFunction\,(W_{[m,m+n]} * [h_{[m,1]}, x_{[n,1]}] + b_{[m,1]}) \quad (1)$$

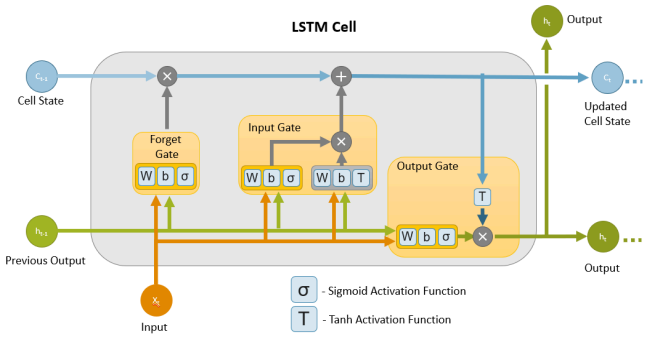


Fig. 2 The Architecture of an LSTM Cell

Forget Gate

The forget gate uses a sigmoid function to determine what information and how much information to retain or erase from the cell state. The output of the sigmoid function ranges from 0 to 1 where 1 means to retain all information, and 0 means to erase it. The update operations for the forget gate are represented by Equations 2 and 3 [7].

$$f_t = \sigma\,(W_f * [h_{t-1}, x_t] + b_f) \quad (2)$$

$$c_f = c_{t-1} * f_t \quad (3)$$

Input Gate

The input gate uses two activation functions: a sigmoid and a tanh. The tanh function yields an output between -1 and 1 and decides the change of the cell state of the current step ($\check{c}_t$). The sigmoid function is responsible for the magnitude of the change (m_t) [7]. The update process is defined by:

$$\check{c}_t = tanh\ (W_c * [h_{t-1}, x_t] + b_c) \quad (4)$$

$$m_t = \sigma\ (W_m * [h_{t-i}, x_t] + b_m) \quad (5)$$

$$c_t = c_f + \check{c}_t * m_t) \quad (6)$$

Output Gate

The update of the output gate can be represented as equations 7 and 8 [7]. The gate receives information from the current input and the previous output as the input (o_t) of its sigmoid activation function. The updated cell state (c_t) will be transformed by a tanh function as part of the prediction of the next output (h_t).

$$o_t = \sigma\ (W_o * [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t * tanh\ (c_t) \quad (8)$$

IV. Data

We used financial data imported from Yahoo Finance on popular technology stocks such as Nvidia, Tesla, Apple, Google, and Microsoft stocks. For the interval of stock price, we chose a 15-minute interval as it is a short-term interval that carries significant price changes for our model to train. The dataset we used spans over 60 days with around 1040 to 1092 records of several features such as the Open, High, Low, Close, Adjusted Close, and Volume. For this study, we mainly focused on the Close price as it is considered the most accurate valuation of a stock price. Due to the sequential nature of the data, it is not appropriate to do random validation and testing and instead we chose to train our model with the first 80 percent of data and test it with the remaining data as shown in Figure 3 below. We prepared two different types of datasets where they perform predictions on either stock prices or stock trends. The stock price dataset consists of 60 actual close-price as predictors and the model needs to predict 1 target. A sequence of data will be used to predict the remaining 20 percent of the stock price. On the other hand, the stock trend dataset consists of the same number of predictors but the subsequent predictors will use predicted price instead of actual price.

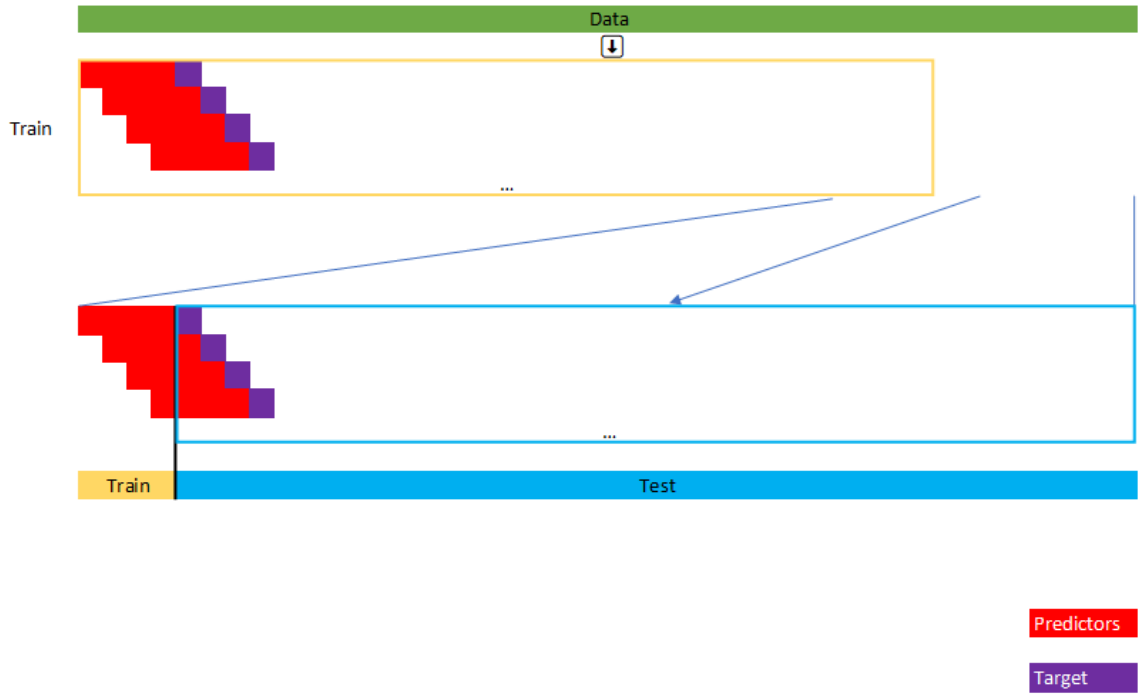


Fig. 3 Visualization of the dataset

Before using the dataset in training the model, preprocessing of the training and testing data is performed over two steps. A MinMaxScaler from the Scikit-learn library is used to normalize the data. Next, we used pad sequences from the Keras library to replace the empty values with zeros in the stock trend dataset. MinMaxScaler was chosen because it can scale the maximum and minimum values to 1 and 0 which allows our model to converge faster.

V. EXPERIMENT

The objective of this experiment is to train an LSTM model with historical data and make a prediction using the trained model. This project utilizes multiple Python tools such as Keras, Scikit-learn, Matplotlib, etc. The overall experiment contains four sections, getting the data, preprocessing, training the model, and testing the model.

A. Getting the data

The first step of the experiment was to load the data from the Yahoo Finance API. 60-day trading records of five selected stocks were imported into the module, with a time stamp for every 15 minutes. In Figure 4 below, the imported data contains multiple attributes: Open, High, Low, Close, Adjust Close, and Volume. In our project, only the close price was selected as the training and testing data.

Datetime	Open	High	Low	Close	Adj Close	Volume
2024-02-29 09:30:00	792.500000	798.299927	791.715027	794.109985	794.109985	6863344
2024-02-29 09:45:00	794.054993	796.729980	790.070007	794.779297	794.779297	3212402
2024-02-29 10:00:00	794.679993	797.849976	791.099976	792.020081	792.020081	2411887
2024-02-29 10:15:00	792.200012	795.400024	791.650085	793.489990	793.489990	1531932
2024-02-29 10:30:00	793.466003	796.450012	792.500000	794.179993	794.179993	1444651

Fig 4. A Snippet of the Nvidia Stock Data

B. Preprocessing

After the selected ‘Close’ data was loaded into the module, it needed to be preprocessed for further training purposes. The first step involves scaling the entire dataset using a Min-Max scaler, normalizing the data between 0 and 1. Next, the data was split into training and validation sets, with 80% designated as the training set and teh remaining 20% as the validation set. For the testing set, each new predicted value was fed back into the model as testing input to generate subsequent predictions. Following the split, we chose 60 data points as the time steps and reshaped the input data in a specific shape: (samples, time steps, features). The number of features was set to one, as we only chose the closing price for our training model.

C. Defining and Training the Model

After preparing our training data, we created an LSTM model using Keras. Figure 5 demonstrates the summary of the model, which includes two LSTM layers, one with 128 nodes and the other with 64 nodes. It also contains two hidden dense layers and one output layer. To avoid overfitting, we implemented a dropout with 0.1 after each LSTM layer. After constructing the LSTM, we trained our mode using the preprocessed data from the previous step.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_12 (LSTM)              (None, 60, 128)           66560

 dropout_12 (Dropout)        (None, 60, 128)           0

 lstm_13 (LSTM)              (None, 64)                49408

 dropout_13 (Dropout)        (None, 64)                0

 dense_18 (Dense)            (None, 32)                2080

 dense_19 (Dense)            (None, 16)                528

 dense_20 (Dense)            (None, 1)                 17

=================================================================
Total params: 118593 (463.25 KB)
Trainable params: 118593 (463.25 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 5. LSTM Model Summary

D. Testing the Model

After finishing the training process, we loaded our model and tested it using the remaining 20% of the data. The testing data were preprocessed with the same scaling and reshaping steps as the training dataset. We applied two types of testing to the trained model: one using the actual data and the other using the new predicted value for subsequent predictions. The latter method required a padding process because the newly predicted values were cycled back into the testing dataset after each prediction. This specially designated process helps avoid the contamination of the newly predicted data with the actual historical data.

VI. Result, Analysis and Limitations

For the results, we trained the model five times individually with different datasets from each stock market we chose. This is to make sure the model is specifically fit to predict prices and trends. The results shown in Figures 6, 7 and 8 below are extracted from the complete graph for easier side-by-side comparison. The complete results are attached to the appendix at this report's end.

A. Stock Price Prediction

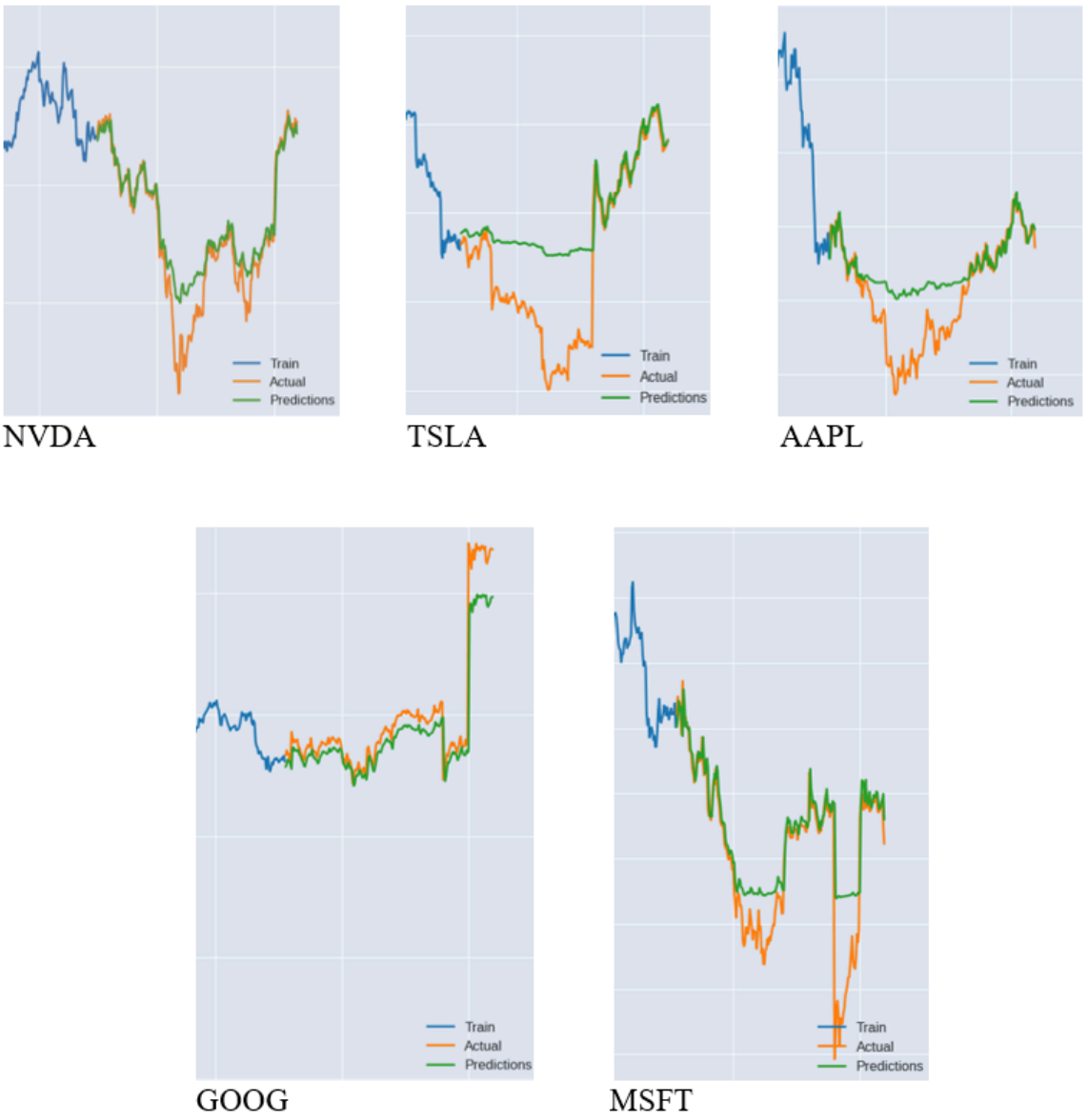


Fig 6. Result of Stock Price Prediction

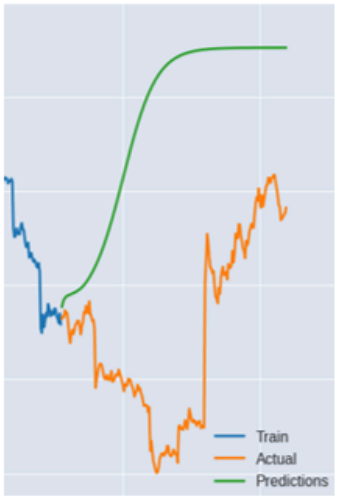
For the result shown in Figure 6, we tested our models with five technology stocks. The result is quite accurate as the model gets actual data to predict the next price. It would mean that the model just needs to

predict one step ahead for each stock market. An interesting observation can be made where the predicted price tends to flatten out on a falling wedge in TSLA, AAPL, and MSFT stocks. This scenario might be caused by the testing data going out of range as compared to the model's training data. The minimum of the training data can support this hypothesis as the model predictions are not likely to go lower than the minimum even though the actual data say otherwise. The lower prediction on GOOG stock when the actual price surges also tells us that the model tends to predict prices within the maximum and minimum range from its training data. This issue can be solved by increasing the training data to cover a larger range of prices or changing the parameter during the normalization.

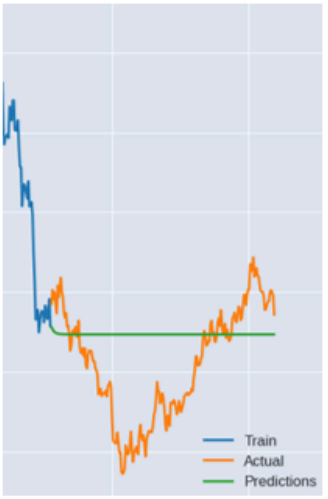
The high accuracy of these predictions seems like the optimal solution for predicting stock prices but it has a major flaw. The model can only predict 1 step ahead which is 15 minutes ahead in our case as it needs the next set of actual data to predict the next price. Thus, it is not useful as it only provides one prediction for us to evaluate the stock market. Therefore, we tested the same model on the stock trend by feeding the model a combination of actual data and predicted data.

B. Stock Trend Prediction

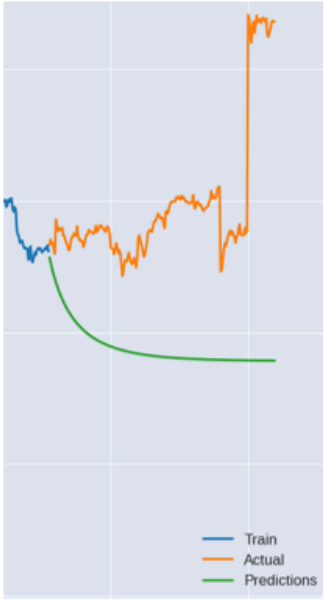
NVDA



TSLA



AAPL



GOOG

MSFT

Fig 7. Result of Stock Trend Prediction

For the result shown in Figure 7, we tested the same model with a different dataset where it used actual data for the first prediction and combined its prediction with actual data for the remaining predictions. As the predictor size we used is 60, the model would expect a dataset filled with predicted values at its 61st prediction. This setup would allow the model to predict the trend of the stock market as it makes guesses on its previous guess. The result would seem very inaccurate at first glance but it still provides useful information in the first few predictions. The model can predict the short-term trends for NVDA, TSLA, AAPL, and MSFT stock as the market follows the trend in the short term. As for GOOG stock, the model predicts a downward trend as it receives testing data that approaches its maximum range. The model did not have training in data with a higher price, hence it predicts the trend is going down. One interesting observation from the results is that the predicted graph will flatten out as the model makes further predictions. This proves that the model tends to converge the prediction to one point given that the testing data does not have much fluctuation. This issue can be solved by making fewer predictions and try predicting the trends at different timestamps.

The overall trends predicted by the model are fairly accurate assuming that the user needs short-term trends. The accuracy will get worse when more predictions are made. The user could get a precise trend if they make frequent predictions with the model.

C. Stock Trend Prediction (alternative method)

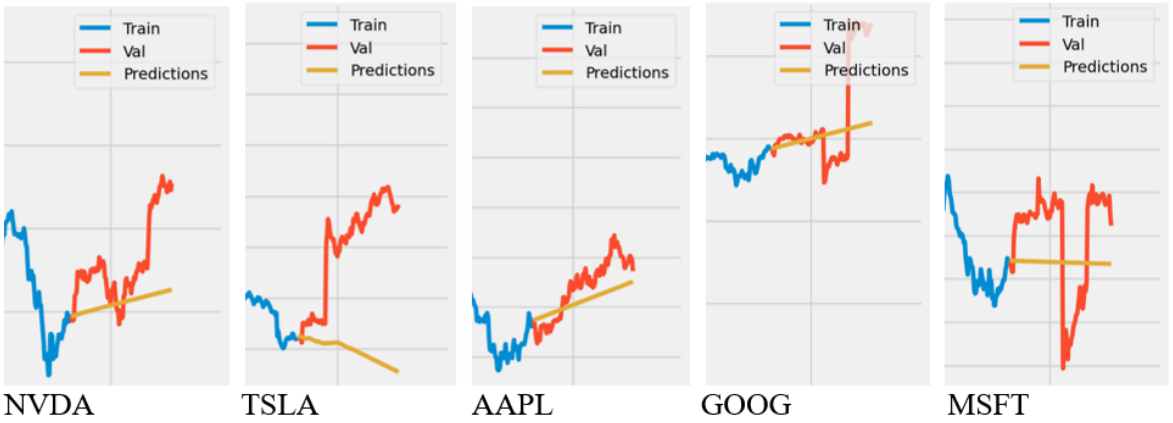


Fig 8. Result of Stock Trend Prediction (alternative method)

In this alternative method, we used price changes as datasets instead of close prices. The preprocessing of data is StandardScaler from the Scikit-learn library as the data are more suitable for standardized than normalized. The results above used 90 percent of the data as the training set and the remaining are used for testing. We could observe that the model is highly accurate in predicting price changes where the predictions fit with the actual trendline. It gets the TSLA stock wrong as the training data consists of mostly a downward trend. This issue can be solved by providing the model larger training set that consists of upward and downward trends.

VII. Conclusion

In conclusion, our project confirms the potential of LSTM models in forecasting stock trends, albeit with limitations that require further exploration. While the LSTM effectively captures short-term dependencies in stock price data, its predictions for longer-time sequences are less reliable, highlighting the challenges of volatile market conditions. Further work is necessary to integrate more variables and diverse data inputs such as Moving Averages (MA), Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), etc. Additionally, experimenting with hybrid models combining LSTM with other deep-learning neural networks could yield improvements in forecasting performance. Our project lays a foundational step towards more sophisticated financial forecasting tools that can adapt to and predict complex market dynamics.

VIII. Future Work

From this study, we found that the LSTM model alone cannot give us an accurate prediction for stock prices and trends. The graph for a stock market is usually volatile and our current model is not capable of recreating the up and down in the stock market. We are able to identify several approaches for more realistic predictions:

1. Adding Random Walk / Brownian Motion to the prediction for uncertainty.
2. Exploring other features such as Volume, Moving Average, Bollinger Bands, and more as datasets or using two or more features to create a multivariate model
3. Removing market noise in datasets such as using the Renko chart before training the model
4. Training the model with other machine learning layers such as using the LLM, ARIMA, Bi-LSTM, CNN-LSTM, or DNN model

Work Contribution

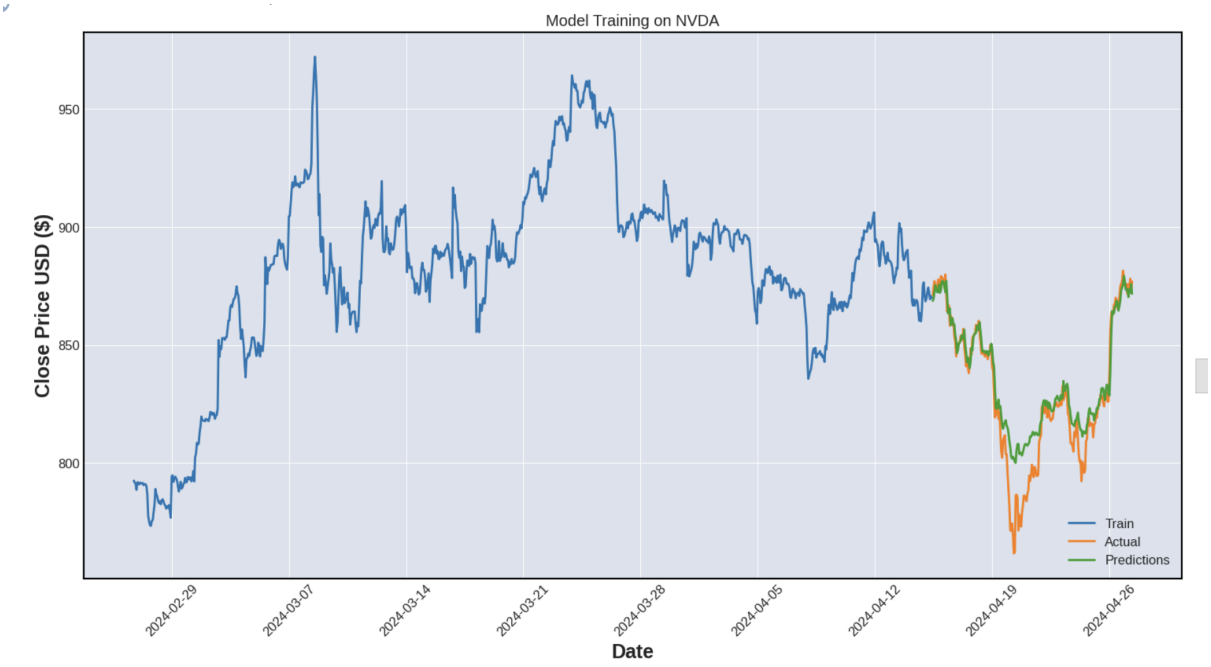
Yubo Zhou: Contributed to developing and training the model, and final report (Sections I, II, III, V, and VII).
Chester Lee: Contributed to developing the code, training the model, testing the model, and final report (Sections IV, VI, and VIII).
Amandeep Singh Dhugga: Contributed to making the slides and validating the code.

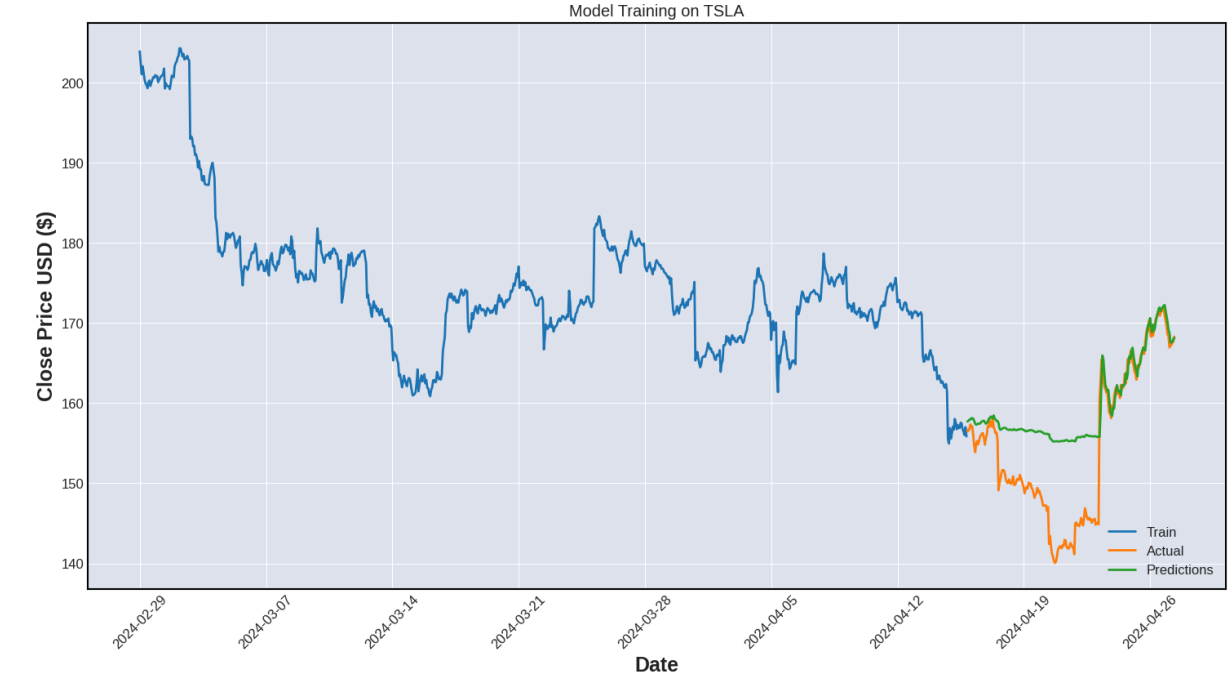
References

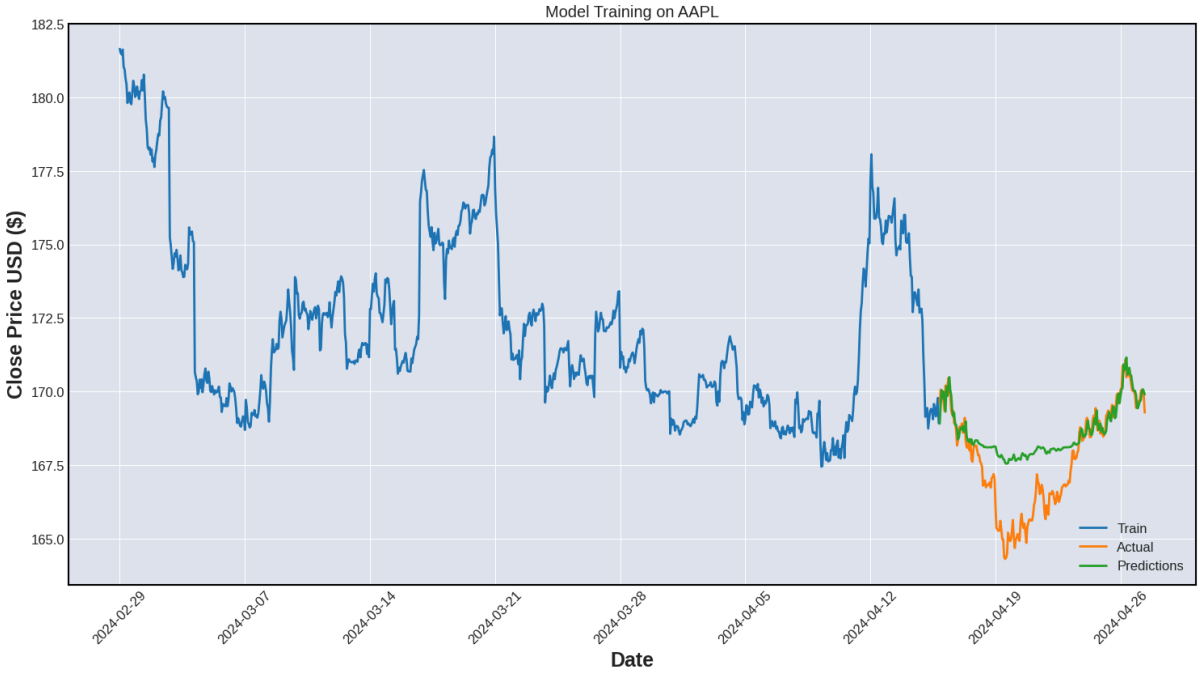
[1] H. Tan, "The Importance of Stock Markets," finance.yahoo.com, Jul. 27, 2020. https://finance.yahoo.com/news/importance-stock-markets-075948914.html
[2] Yahoo Finance, "Yahoo Finance - Business Finance, Stock Market, Quotes, News," Yahoo Finance, 2024. https://finance.yahoo.com/
[3] Z. Zou and Z. Qu, "Using LSTM in Stock Prediction and Quantitative Trading," Stanford University, 2020.
[4] S. Behera, C. Prakash, and N. Sharma, "A Combined Model for INDEX Price Forecasting Using LSTM, RNN, and GRU," in Advances in Data-Driven Computing and Intelligent Systems, S. Das, S. Saha, C. A. C. Coello, H. Rathore, and J. C. Bansal, Eds. Singapore: Springer, 2024, vol. 890, doi: 10.1007/978-981-99-9531-8_40.
[5] C. Olah, "Understanding LSTM Networks," Github.io, Aug. 27, 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[6] "Recurrent Neural Network," NVIDIA Developer, Oct. 11, 2018. https://developer.nvidia.com/discover/recurrent-neural-network
[7] C. B. Vennerød, A. Kjærran, and E. S. Bugge, "Long Short-term Memory RNN," arXiv.org, May 14, 2021. https://arxiv.org/abs/2105.06756

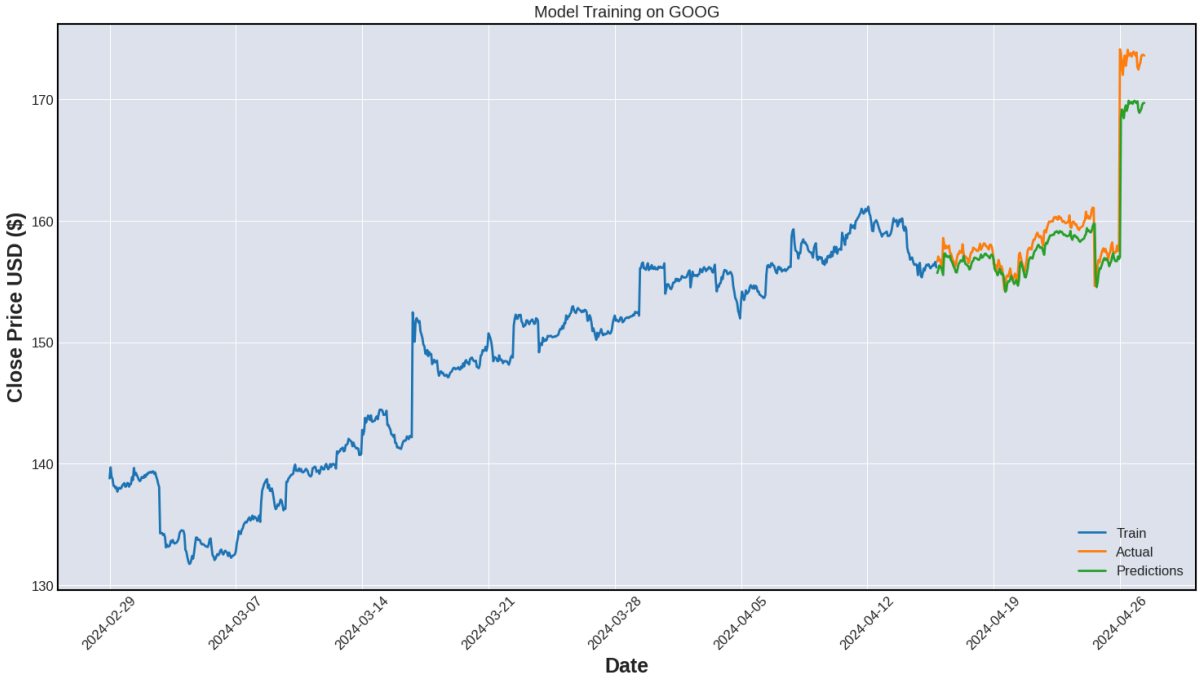
Appendix

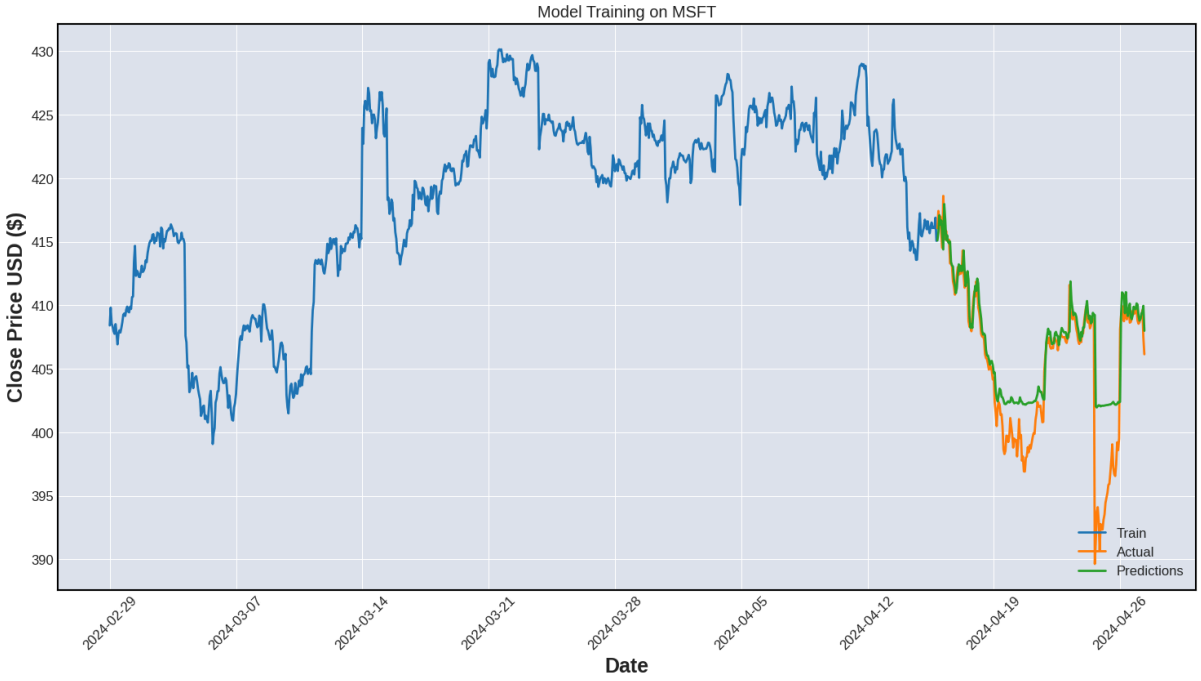
A. Stock Price Prediction Results



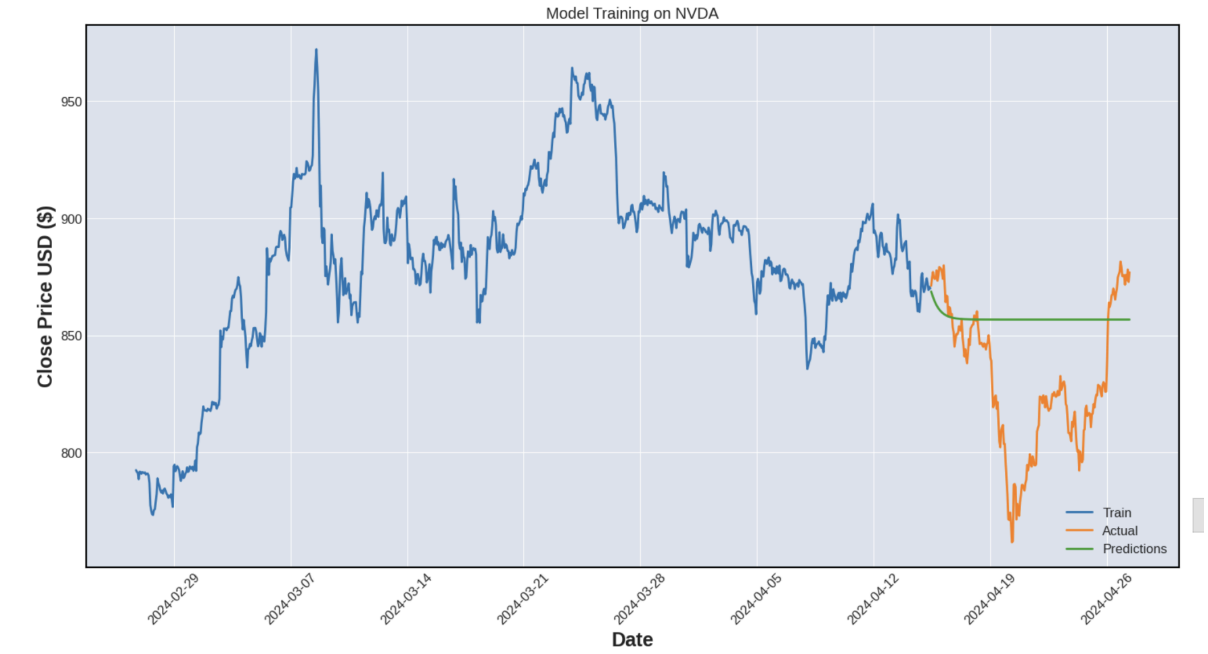


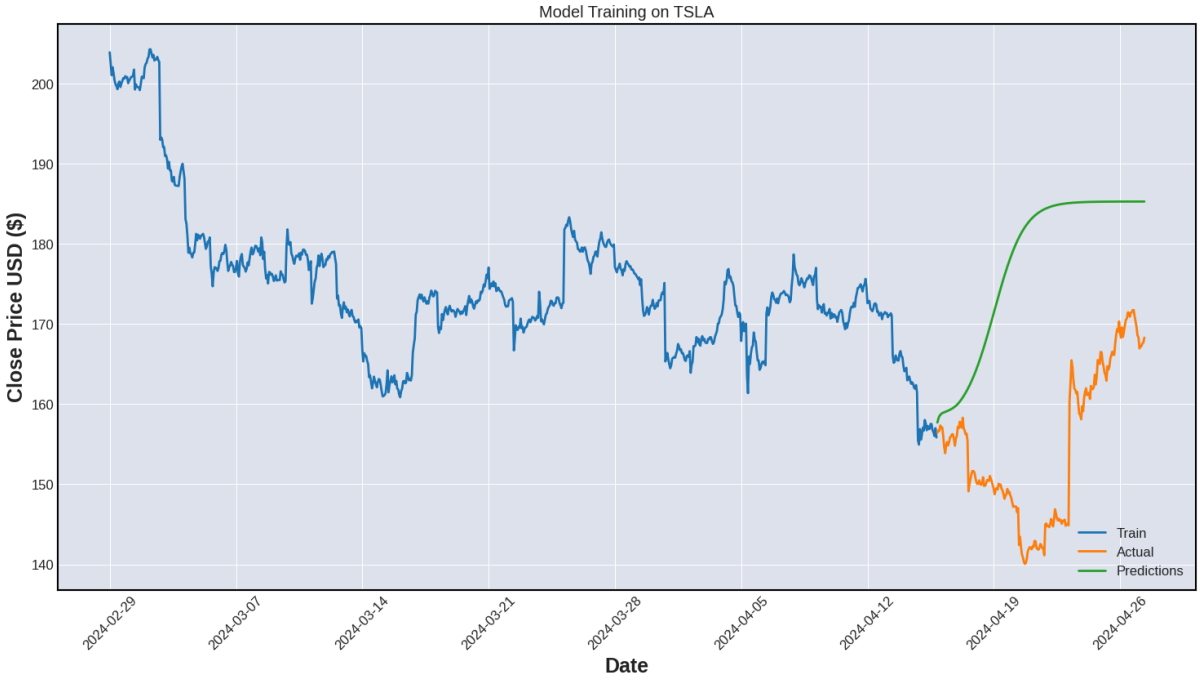

Model Training on AAPL
Close Price USD ($)
182.5
180.0
177.5
175.0
172.5
170.0
167.5
165.0
2024-02-29
2024-03-07
2024-03-14
2024-03-21
2024-03-28
2024-04-05
2024-04-12
2024-04-19
2024-04-26
Date
Train
Actual
Predictions

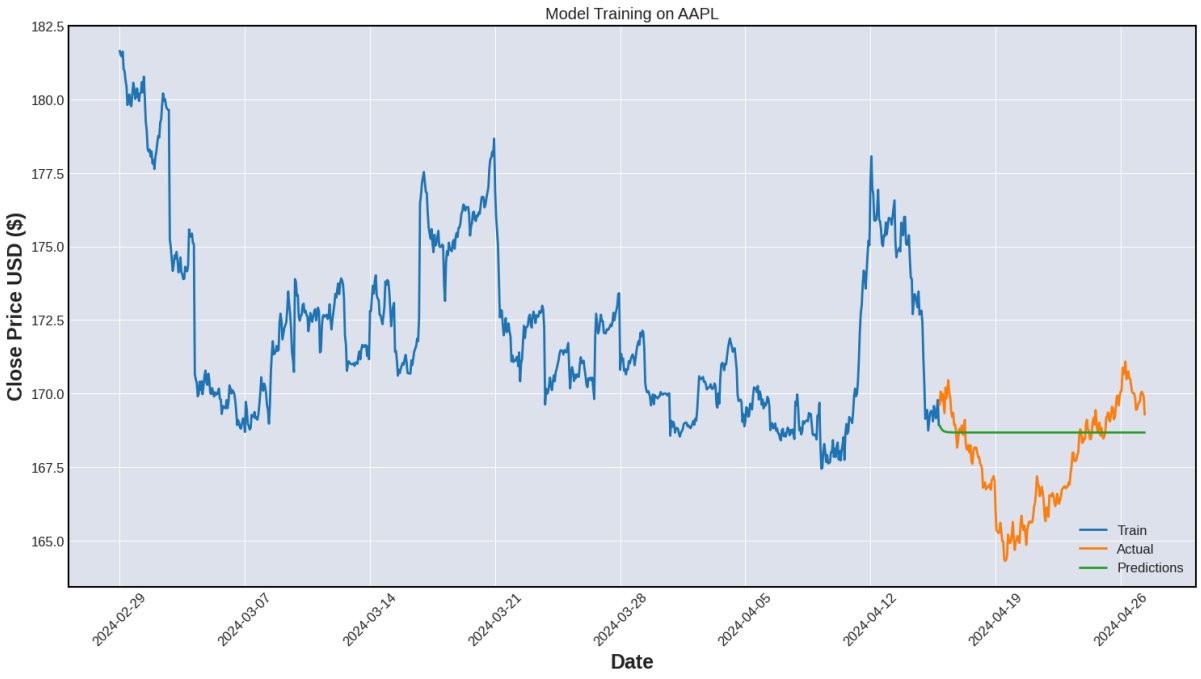

Model Training on GOOG
Close Price USD ($)
170
160
150
140
130
2024-02-29
2024-03-07
2024-03-14
2024-03-21
2024-03-28
2024-04-05
2024-04-12
2024-04-19
2024-04-26
Date
Train
Actual
Predictions

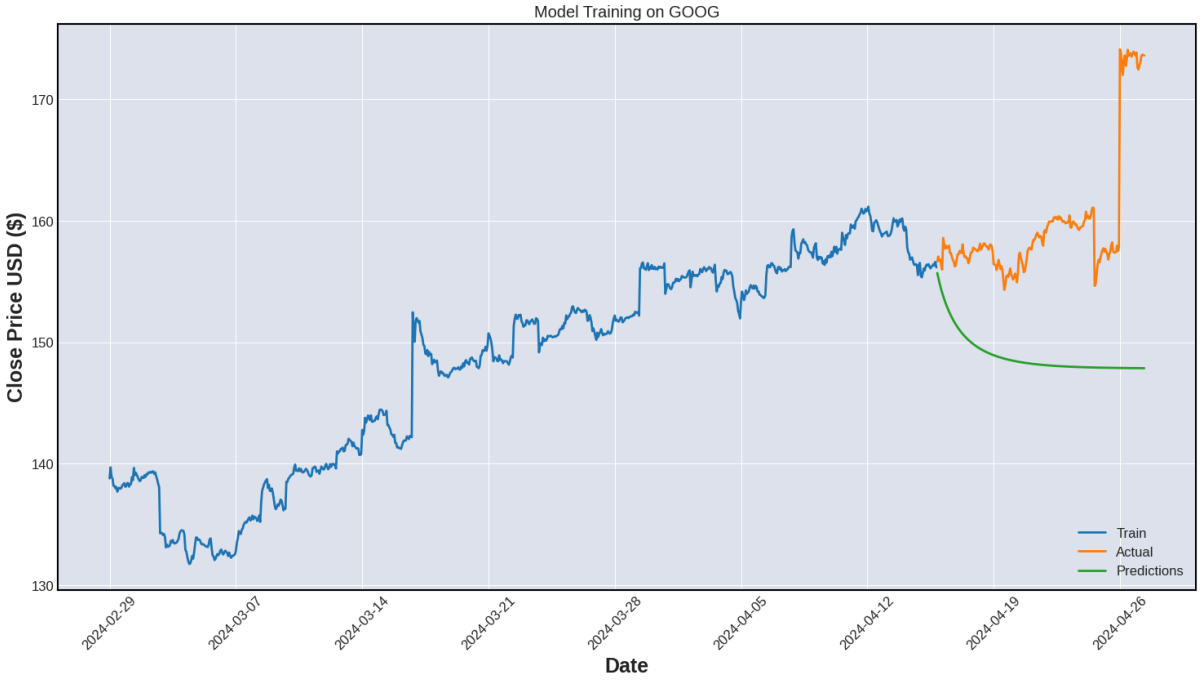


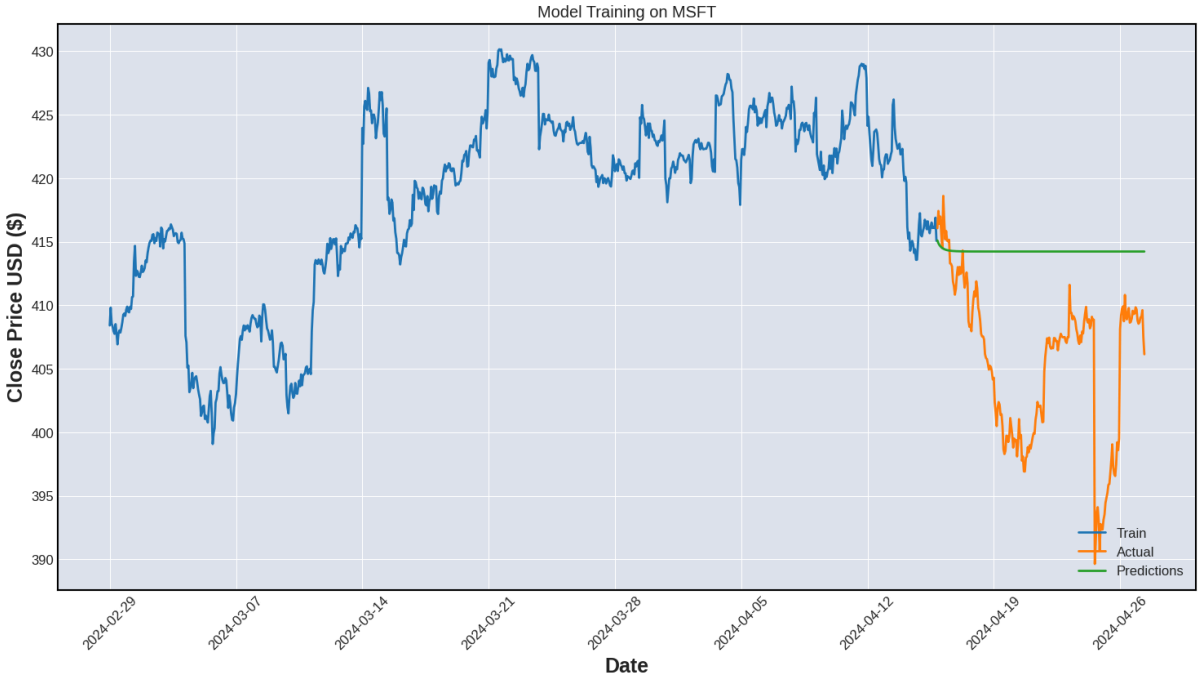
B. Stock Trend Prediction Results



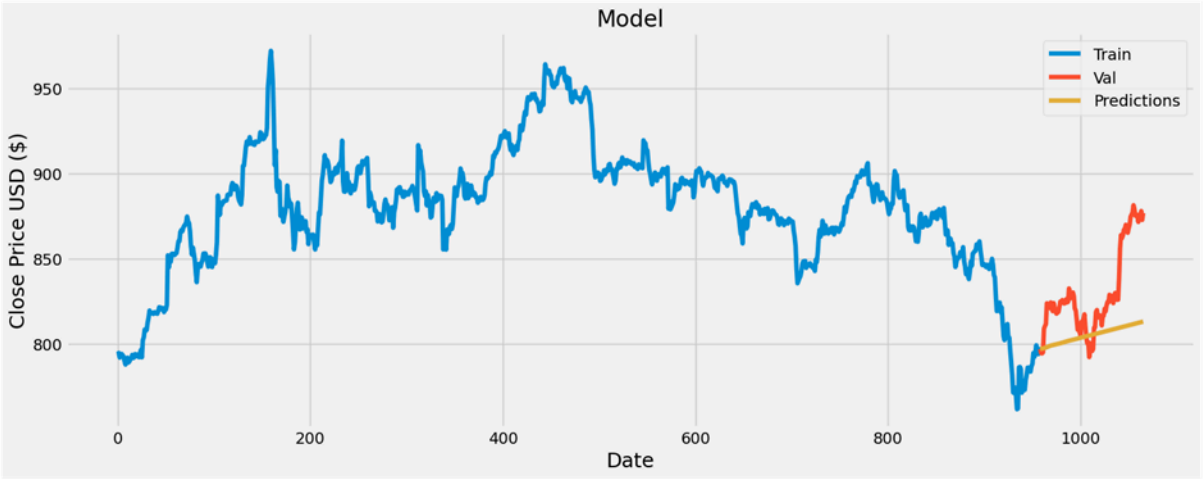

Model Training on TSLA
Close Price USD ($)
200
190
180
170
160
150
140
2024-02-29
2024-03-07
2024-03-14
2024-03-21
2024-03-28
2024-04-05
2024-04-12
2024-04-19
2024-04-26
Date
Train
Actual
Predictions


Model Training on AAPL
Close Price USD ($)
182.5
180.0
177.5
175.0
172.5
170.0
167.5
165.0
2024-02-29
2024-03-07
2024-03-14
2024-03-21
2024-03-28
2024-04-05
2024-04-12
2024-04-19
2024-04-26
Date
Train
Actual
Predictions

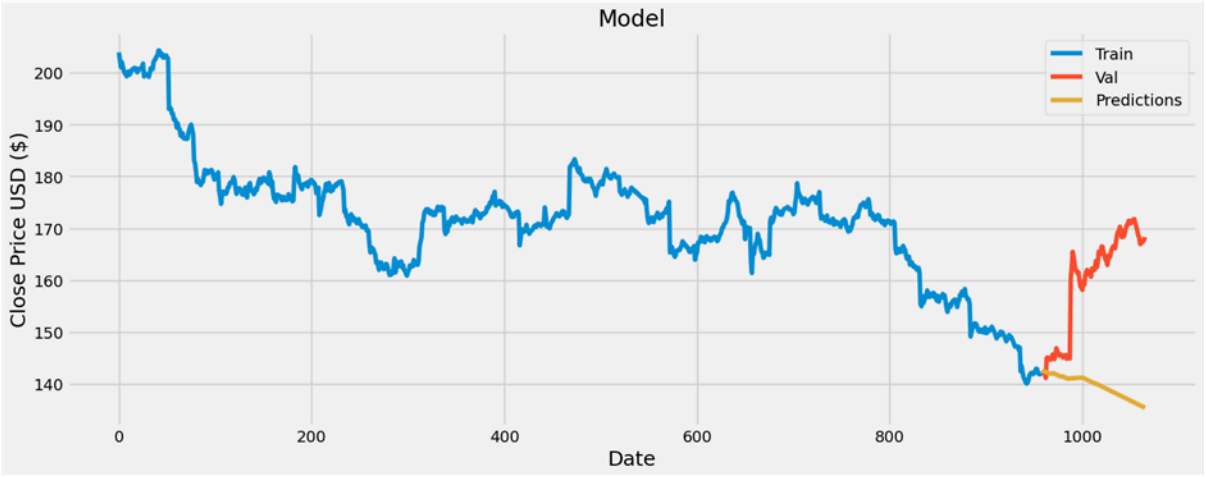




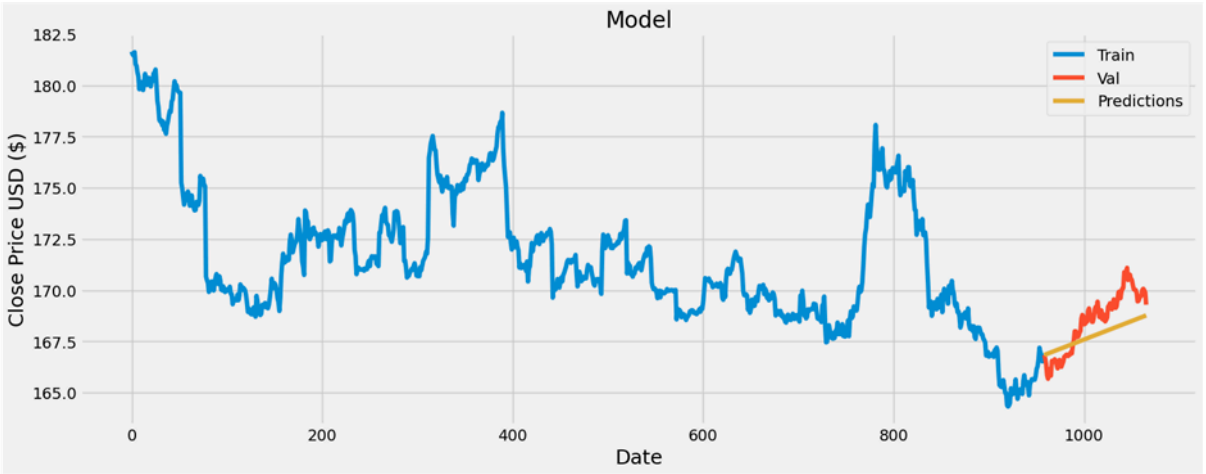
C. Stock Trend Prediction Results (alternative method)



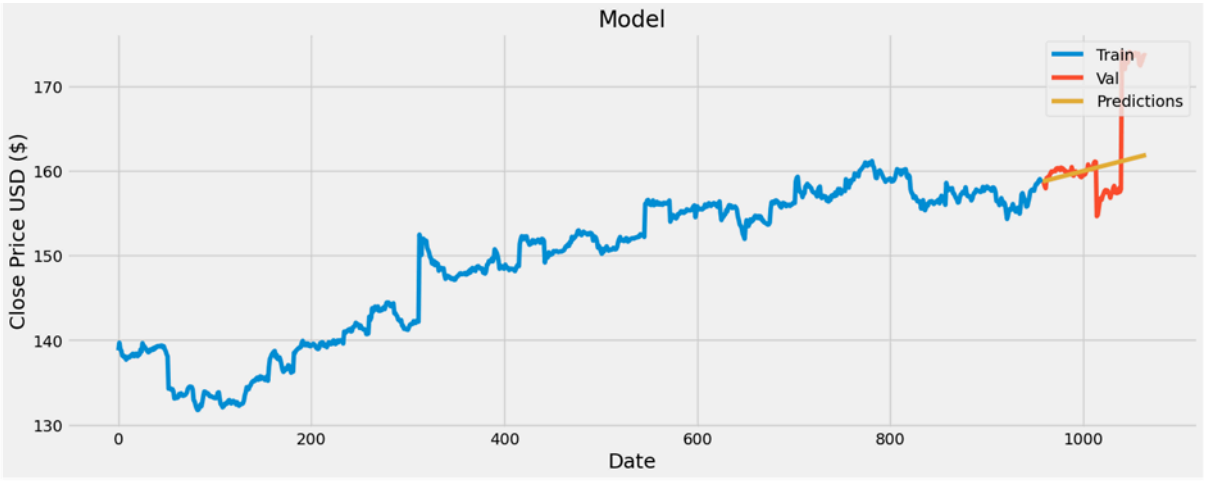
NVDA stock



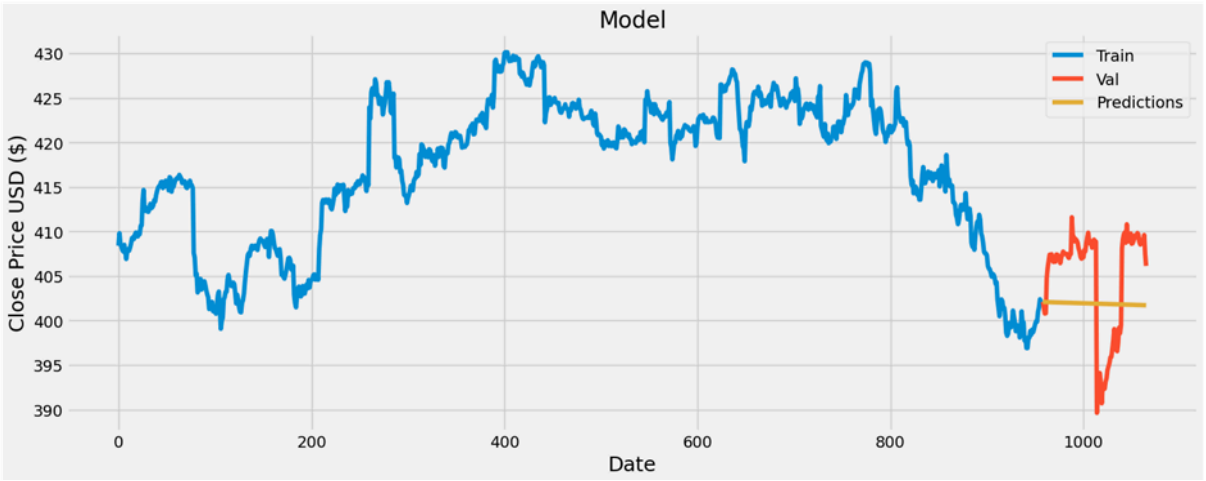
TSLA stock



AAPL stock



GOOG stock



MSFT stock

GitHub Repo: https://github.com/Deletenomore/CSCI164_LSTM_Stock_Trend_Prediction.git