Stock Price Prediction Using LSTM

Naveena Pokala
Department Of Computer Science
University Of North Carolina At Greensboro
Greensboro, USA
n_pokala@uncg.edu

***Abstract*—This project predicts the stock price of Apple using the LSTM neural network model. The dataset includes the date, opening price, highest price, lowest price, closing price, adjusted closing price and volume. This project uses the closing price for prediction with the neural network.**

I. Introduction

This project predicts the stock prices of a particular company using the Long Short-Term Memory (LSTM) neural network algorithm. The LSTM algorithm is an advanced variant of recurrent neural networks (RNNs) that can capture the temporal dependencies and patterns of time-series data, making it suitable for predicting stock prices. The project involves data preprocessing, the training of the LSTM model using historical stock market data and analyzing the results.

II. Problem Statement

Estimating correct stock price is crucial for investors to make informed decisions and manage risks. The stock market is highly volatile and the price fluctuations are more, making it difficult to forecast. The current stock price is highly dependent on the previous stock price and pattern. Traditional methods struggle to memorize the temporal data. Hence, we need a technique which can memorize this sequential and non-linear data.

III. Problem Solution

LSTM networks are a type of Recurrent Neural Networks (RNNs) in deep learning that are able to handle long term dependencies of the sequential data such as time series, text and speech. Hence, this project leverages the LSTM technique to analyze the stock prices and pattern for the past 60 days and uses them to forecast the stock price for the current day.

IV. Project Objective

The objective of this project is to predict the current stock price, analyze the historical stock prices and capture temporal dependencies by using LSTM techinique. This project also focuses on effect of hyperparamters on the model and finding the best fit.

V. Theoretical Knowledge

Recurrent Neural Networks have a hidden state that stores the information and it gets updated with every new timestamp making it difficult to learn long-term dependencies, Hence hidden states are used for short term memory. To overcome this problem, the LSTM has a memory cell which is responsible for retaining the information for extended periods. Memory cell is controlled by three gates: the forget gate, the input gate, and the output gate. These gates control what information needs to stored and what information needs to be discarded. The representation of the gates are shown in Fig.1, where C_{t-1} and C_t are the previous and current memory cell states and H_{t-1} and H_t are the previous and current hidden states.

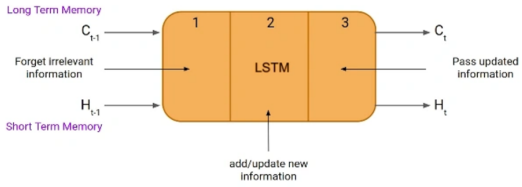

Fig. 1. Visual Representation Of Forget,Input and Output gates

A. Forget Gate

In LSTM Neural Network, forget gate is the first step where it decides whether to retain the information or discard any irrelevant information. The function for forget gate can be written as,

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

where,
f_t is the forget gate output,
h_{t-1} is the previous hidden state,
x_t is the current input, and
W_f is the weight of the input,
b_f is the bias of the forget gate,
σ is the sigmoid activation function

An activation function is applied to the forget gate whose output is between 0 and 1. The forget gate output is later multipled with previous cell state. If the result is 0, the information is discarded. If the result is 1, the data is retained as shown in Fig.2.

$C_{t-1} * f_t = 0$ ...if $f_t = 0$ (forget everything)

$C_{t-1} * f_t = C_{t-1}$ ...if $f_t = 1$ (forget nothing)

Fig. 2. Applying Previous Cell state to the forget gate

B. Input Gate

Input gate is the second step in LSTM where the new information is filtered based on the importance and any irrelevant information is discarded and required information is added to the cell state. The input gate function can be written as,

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

where,

i_t is the input gate output,

h_{t-1} is the previous hidden state,

x_t is the current input, and

W_i is the weight matrix of sigmoid operater between the input gate and forget gate,

b_i is the bias of the input gate.

The sigmoid function is applied and the result will be between 0 and 1. Then, A vector is created using tanh function that gives an output from -1 to +1, which contains all the possible values from h_{t-1} and x_t. The function can be expressed as,

$$\hat{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

where,

h_{t-1} is the previous hidden state,

x_t is the current input, and

W_c is the weight of the input,

b_c is the bias

i_t says how much of information is needed and $\hat{C}_t$ says what information is required. Now, this new information needs to be passed to cell state. This can be expressed as,

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t$$

where,

C_t is the memory cell state of the current timestamp,

C_{t-1} is the memory cell state of the previous timestamp,

$\odot$ is element-wise multiplication

C. Output Gate

The output gate returns the hidden state for the next timestamp. The output gate consists of two parts: first, a Sigma function decides on the percentage of the required information and secondly, the newly updated cell state is passed through a Tanh function and multiplied with the output from the sigma function; this is now the new hidden state. The equations for this can be expressed as,

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

where,

o_t is the output,

W_o is the weight function of the input

b_o is the bias of the output gate

$$h_t = o_t \cdot \tanh(C_t)$$

where,

h_t is the hidden state of the current timestamp.

The output of the current timestamp can be retrived from current hidden state and can be expressed as,

$$\text{output} = \text{softmax}(h_t)$$

VI. Working Of LSTM

The first layer's inputs are connected through the gates and output of the first layer is fed to the input of the following layer, and so on. The final layer's outputs are a neural network's outputs as shown in Fig.4. This network is fully connected and is referred to as feed-forward because all data travels in the forward direction with no back-loops, and every neuron receives outputs from every neuron in the layer before it.

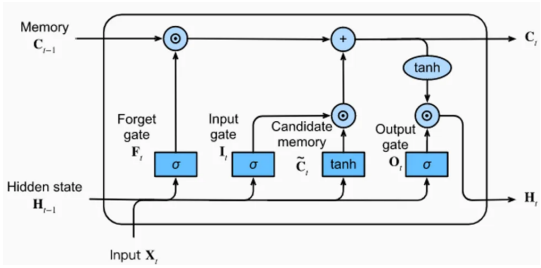

Fig. 3. Architecture Of LSTM layer

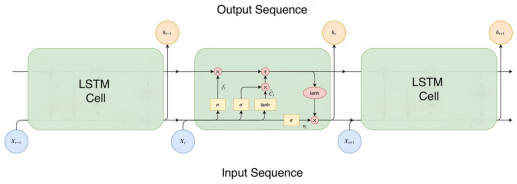

Fig. 4. Representation of LSTM Layers

VII. Project Implementation

A. Dataset

For this mini project, Apple stock prices dataset was obtained from Kaggle, which is a publicly available resource. This dataset consists of date, opening price, highest price, lowest price, closing price, adjusted closing price and volume columns and 10468 rows. Opening Price is the stock price at the beginning of the trading day. Closing Price is the stock price at the end of the trading day. High Price is the highest price reached during the trading day. Low Price is the lowest price reached during the trading day. Volume is the number of shares traded during the day. A snapshot of the dataset is presented in Fig. 5.

Date	Open	High	Low	Close	Adj Close	Volume
12-12-1980	0.128348	0.128906	0.128348	0.128348	0.100178	469033600
15-12-1980	0.12221	0.12221	0.121652	0.121652	0.094952	175884800
16-12-1980	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
17-12-1980	0.115513	0.116071	0.115513	0.115513	0.09016	86441600
18-12-1980	0.118862	0.11942	0.118862	0.118862	0.092774	73449600
19-12-1980	0.126116	0.126674	0.126116	0.126116	0.098436	48630400
22-12-1980	0.132254	0.132813	0.132254	0.132254	0.103227	37363200
23-12-1980	0.137835	0.138393	0.137835	0.137835	0.107583	46950400

Fig. 5. Snapshot of the Dataset

```
Total Missing Values Present :
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Fig. 6. Missing Values In The Dataset

B. Data Preprocessing - Exploratory Data Analysis

Before training the model, exploratory data analysis is performed to understand and check if the data is in correct format. As shown in Fig. 6, the dataset was analyzed for missing values, and there are no missing values. Apple stock trend is plotted to analyze the pattern as shown in Fig.7.

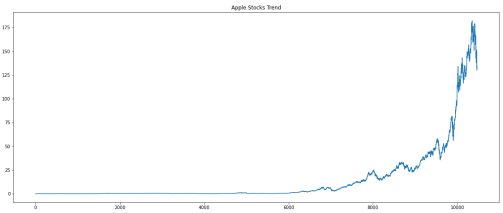

Fig. 7. Apple Stock Trend

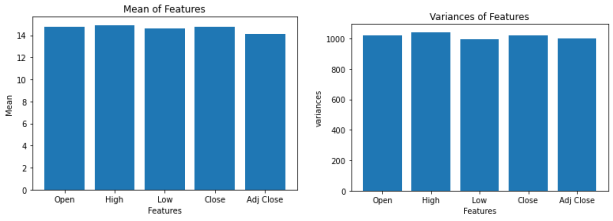

Fig. 8. Mean And Variance Of The Features

Statistical measures are calculated and plotted as shown in Fig.8. Violin plot is used to check for any possible outliers as shown in Fig.9. From the violin plot, we can understand that there are no outliers present in the data.

To better understand how the features are correlated with each other, heat map is plotted as shown in Fig.10. Correlation between Open, High, Low, Close and Adj Close have is 1, which means there is a perfect positive linear relationships between these features and they provide the same information. Hence considered the Closing price for the modeling and predicting the current stock price.

C. Data Normalization

Normalized the Close price Feature using min-max scaler and saved them in a spreadsheet. The scaled prices after normalization are between 0 and 1. The normalization function is as follows,

$$X_{\text{new}} = \frac{X_{\text{old}} - \min(X)}{\max(X) - \min(X)}$$

Fig. 9. Violin Plot Of The Dataset

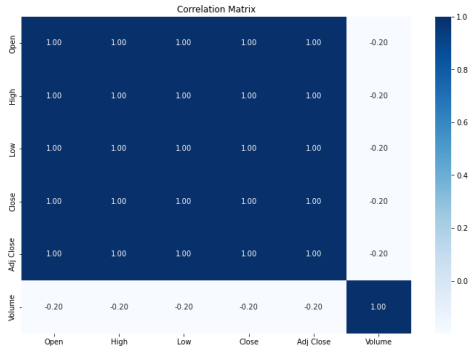

Fig. 10. Heat map of the features

D. Modeling the data

The dataset is divided into 80:20 where 80% is the train data and 20% is the test data and saved them in spreasheets. The model has three LSTM layers each containing 50 memory cells. The model is trained by using stock price data from the first 60 days as input which are considered as features and predicting the 61st day's stock price as output for the current day. This process is repeated iteratively, with the output from the first LSTM layer passed as input to the subsequent LSTM layer as shown in Fig.11 . Adam optimizer is used to adjust the

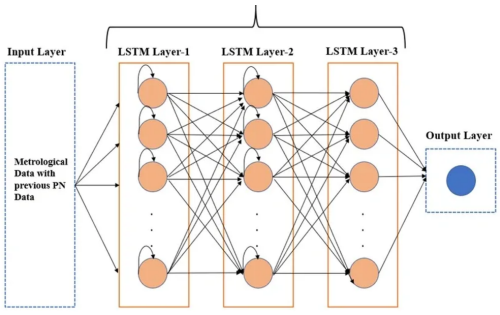

Fig. 11. Representation of LSTM Layers with Memory Cells

weights and biases of the model such that the error between the model's predictions and the true values is minimum. The error is calculated by Mean Squared Error which is used as a loss function. The model is trained by iterating the train dataset and updating the model's parameters and minimizing

```
Epoch 1/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 27s 59ms/step - loss: 9.1736e-05
Epoch 2/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 16s 61ms/step - loss: 7.3769e-06
Epoch 3/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 59ms/step - loss: 3.7861e-06
Epoch 4/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 16s 63ms/step - loss: 3.7516e-06
Epoch 5/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 3.6647e-06
Epoch 6/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 59ms/step - loss: 3.0483e-06
Epoch 7/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 3.1035e-06
Epoch 8/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 3.1506e-06
Epoch 9/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 57ms/step - loss: 2.7815e-06
Epoch 10/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 2.7028e-06
Epoch 11/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 2.1061e-06
Epoch 12/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 57ms/step - loss: 2.8061e-06
Epoch 13/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 2.2051e-06
Epoch 14/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 1.9473e-06
Epoch 15/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 1.8204e-06
Epoch 16/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 58ms/step - loss: 1.8886e-06
Epoch 17/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 57ms/step - loss: 1.8336e-06
Epoch 18/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 59ms/step - loss: 1.6192e-06
Epoch 19/24
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 57ms/step - loss: 1.5824e-06
```

Fig. 12. Minimizing The Loss Function

```
LSTM Model Evaluation With 20 Epochs:
MSE: 46.42
RMSE: 6.81
MAE: 3.92
MAPE: 4.16%
R2 Score: 0.98
```

```
LSTM Model Evaluation With 25 Epochs:
MSE: 14.98
RMSE: 3.87
MAE: 2.06
MAPE: 2.24%
R2 Score: 0.99
```

Fig. 13. Evaluation Metrics For LSTM Model With 20 Epochs and 25 Epochs

the loss function as shown in Fig.12.

E. Effect of hyperparameters And Hyperparameter Tuning

The model is experimented with 20 and 25 epochs to understand the effect of hyperparamters on the model. An epoch refers to one complete pass through the entire training dataset and with each iteration hyperparamters are updated through Adam optimizer. Plotted the graphs with actual stock price and predicted stock price with 20 Epochs and 25 Epochs as shown in Fig.14 and Fig.15. Though both of them follow the stock trends and patterns effectively, there is a significant difference in the metrics as shown in Fig.13.

F. Comparision Between LSTM Model with 20 and 25 Epochs

From the Fig 13, we can understand that the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are decreased drastically from 46.42 to 14.98 and from 6.81 to 3.87. This indicates that the predictions are closer to the actual values. Mean Absolute Error (MAE) is improved from 3.92 to 2.06 with 25 Epochs.This explains that the model has better overall fit. Mean Absolute Percentage Error (MAPE) is also improved from 4.16% to 2.24%, which means the model's relative error percentage has decreased.

G. Results

LSTM model has analyzed the historical stock price and able to predict the current stock price. From the graphs and

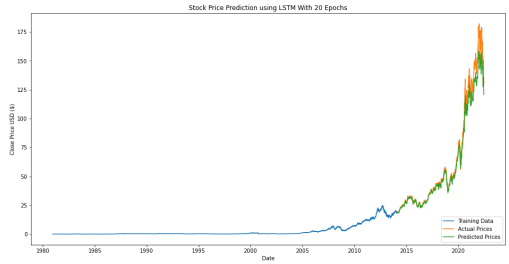

Fig. 14. Actual Vs Predicted Stock Prices with 20 Epochs

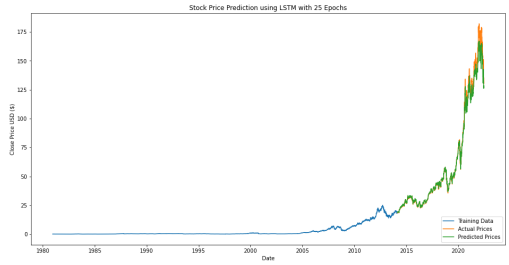

Fig. 15. Actual Vs Predicted Stock Prices with 25 Epochs

metrics, it is evident that LSTM has the best fit with 25 Epochs, with R2 Score as 0.99, meaning the model is able to describe the 99% of variance in the data.

VIII. Conclusion

In this project, we have performed data preprocessing, training the model and tuning the hyperparameters and leveraged the LSTM Technique to predict the current stock price. Model's performance is evaluated using Mean Squared Error(MSE), Root Mean Squared Error(RMSE), Mean Absolute Error(MAE), Mean Absolute Percentage Error (MAPE) and R2-Scores.

IX. References

1) Analytics Vidhya. (2021). Introduction to Long Short-Term Memory (LSTM). Retrieved from https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/
2) GeeksforGeeks. Deep Learning: Introduction to Long Short-Term Memory. Retrieved from https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/
3) ProjectPro. LSTM Model. Retrieved from https://www.projectpro.io/article/lstm-model/832