STOCK PRICE PREDICTION

INT254(Fundamentals of Machine Learning)

PROJECT REPORT

By

ROHIT RAJ

Reg. no. :- 12006524

Submitted to – Md. Imran Hussain

School of Computer Science Engineering

Lovely Professional University, Jalandhar

Date of submission: 15-11-22

Table of Contents:

ACKNOWLEDGEMENT

I am particularly grateful for the precious help I got throughout the duration of the semester from my teacher **Md. Imran Hussain**, his guidance on this research-based capstone was very much valued and appreciated. I also feel grateful towards all the other teachers I had throughout my Computer Science bachelor's degree, they inculked invaluable skills that helped me throughout this project and more. In addition, I would like to thank each one who helped me compile this research by providing research papers, tutorials, figures, and pre-built models.

Moreover, I would like to thank people in my entourage especially my family and friends for all the moral support they showed me through both good and bad times. Finally, I would like to express my sincere gratitude to anyone taking the time and effort to read this report which I hope will be an interesting read for you

Sincerely,

Rohit Raj

(12006524)

ABSTRACT

In this project we attempt to implement machine learning approach to predict stock prices. Machine learning is effectively implemented in forecasting stock prices. The objective is to predict the stock prices to make more informed and accurate investment decisions.

We propose a stock price prediction system that integrates mathematical functions, machine learning, and other external factors for the purpose of achieving better stock prediction accuracy and issuing profitable trades. There are two types of stocks. You may know of intraday trading by the commonly used term "day trading." Intraday traders hold securities positions from at least one day to the next and often for several days to weeks or months. LSTMs are very powerful in sequence prediction problems because they're able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price. While predicting the actual price of a stock is an uphill climb, we can build a model that will predict whether the price will go up or down.

Keywords: LSTM, CNN, ML, DL, Trade Open, Trade Close, Trade Low, Trade High

INTRODUCTION

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities, and derivatives over virtual platforms supported by brokers. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. This market has given investors the chance of gaining money and having a prosperous life through investing small initial amounts of money, low risk compared to the risk of opening new business or the need of high salary career. Stock markets are affected by many factors causing the uncertainty and high volatility in the market.

This project seeks to utilize Deep Learning models, Long-Short Term Memory (LSTM) Neural Network algorithm, to predict stock prices. For data with timeframes recurrent neural networks (RNNs) come in handy but recent research have shown that LSTM, networks are the most popular and useful variants of RNNs. I will use Keras to build a LSTM to predict stock prices using historical closing price and trading volume and visualize both the predicted price values over time and the optimal parameters for the model.

PROBLEM STATEMENT

The challenge of this project is to accurately predict the future closing value of a given stock across a given period in the future. For this project I will use a Long Short Term Memory Neural Network.

GOALS:

1. Explore stock prices.
2. Implement basic model using linear regression.
3. Implement LSTM using Keras library.
4. Create a web Application for easy access.

Dataset Used

The data used in this project is of the Apple Inc from January 1, 2010, to October 31, 2022. This is a series of data points indexed in time order or a time series. Our goal was to predict the closing price for any given date after training. For ease of reproducibility and reusability, all data was pulled from the Yahoo Finance Python datareader library. The prediction has to be made for Closing (Adjusted closing) price of the data. Since Yahoo Finance already adjusts the closing prices for us, we just need to make prediction for “CLOSE” price.
The dataset is of following form:

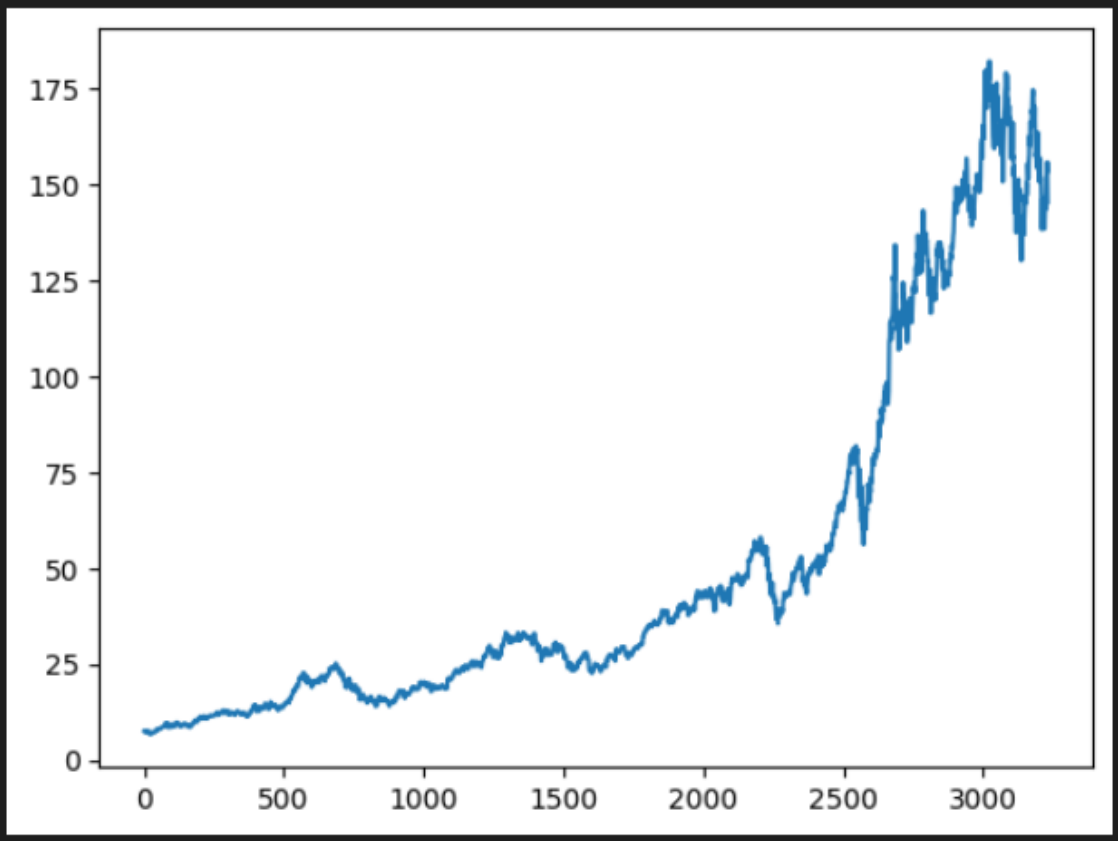
Date	High	Low	Open	Close	Volume	Adj Close
2009-12-31	7.619643	7.520000	7.611786	7.526071	352410800.0	6.415359
2010-01-04	7.660714	7.585000	7.622500	7.643214	493729600.0	6.515212
2010-01-05	7.699643	7.616071	7.664286	7.656429	601904800.0	6.526476
2010-01-06	7.686786	7.526786	7.656429	7.534643	552160000.0	6.422664
2010-01-07	7.571429	7.466071	7.562500	7.520714	477131200.0	6.410791

We can infer from this dataset that date, high and low values are not important features of the data. As it does not matter at what was the highest prices of the stock for a particular day or what was the lowest trading prices. What matters is the opening price of the stock and closing prices of the stock. If at the end of the

day we have higher closing prices than the opening prices that we have some profit otherwise we saw losses. Also, volume of share is important as a rising market should see rising volume, i.e., increasing price and decreasing volume show lack of interest, and this is a warning of a potential reversal. A price drop (or rise) on large volume is a stronger signal that something in the stock has fundamentally changed.

To visualize the data, we have used matplotlib library. We plotted Closing stock price of the data with the no of items(no of days) available.
Following is the snapshot of the plotted data:

```
plt.plot(df.Close)
✓ 0.1s
[<matplotlib.lines.Line2D at 0x2254cf96a00>]
```



Libraries Used

There exits a lot of libraries that make working with data and creating models much simpler. Some of the important libraries used are as follows:

- **Numpy:** NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

- **Pandas:** Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for grouping, combining and filtering data.

- **Matplotlib:** Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots.

- **Keras:** Keras is a very popular Machine Learning library for Python. It is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One of the best thing about Keras is that it allows for easy and fast prototyping.

- **Streamlit:** Streamlit lets you turn data scripts into shareable web apps in minutes, not weeks. It’s all Python, open-source, and free! And once you’ve created an app you can use our Community Cloud platform to deploy, manage, and share your app.

- **Scikit-learn:** Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data mining and data-analysis, which makes it a great tool who is starting out with ML.

- **TensorFlow**: TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, TensorFlow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

Theoretical Background

Long Short-Term Memory model (LSTM)

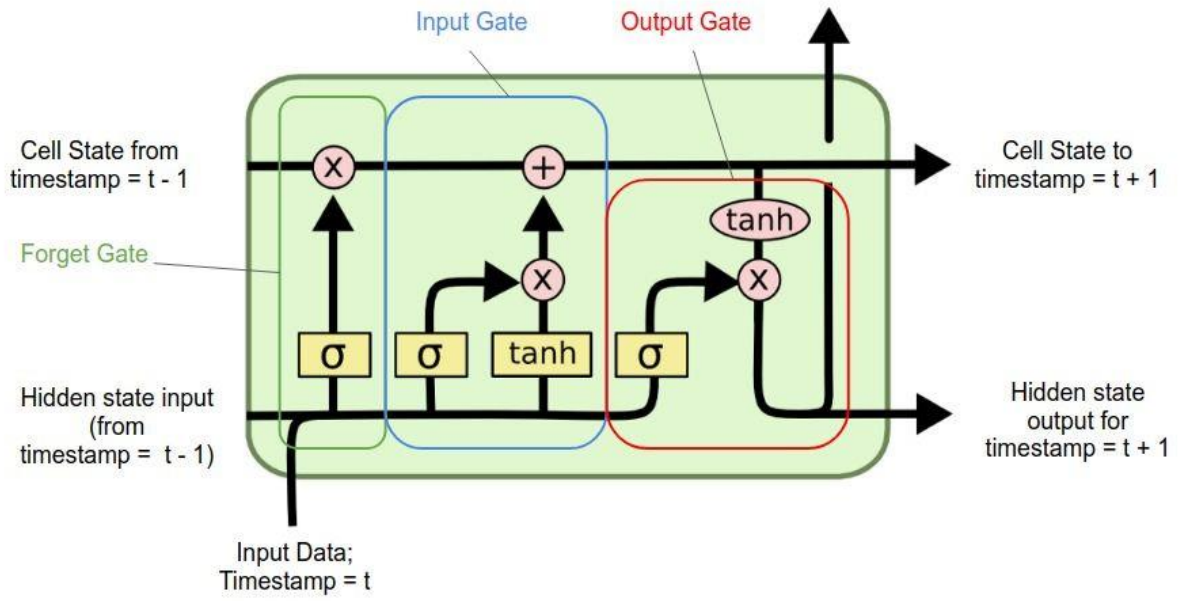
Long Short-Term Memory (LSTM) is one of many types of Recurrent Neural Network RNN, it's also capable of catching data from past stages and use it for future predictions. In general, an Artificial Neural Network (ANN) consists of three layers:

1) input layer,
2) Hidden layers,
3) output layer.

In a NN that only contains one hidden layer the number of nodes in the input layer always depend on the dimension of the data, the nodes of the input layer connect to the hidden layer via links called 'synapses. The relation between every two nodes from (input to the hidden layer), has a coefficient called weight, which is the decision maker for signals. The process of learning is naturally a continues adjustment of weights, after completing the process of learning, the Artificial NN will have optimal weights for each synapse. The hidden layer nodes apply a sigmoid or tangent hyperbolic (tanh) function on the sum of weights coming from the input layer, which is called the activation function, this

transformation will generate values, with a minimized error rate between the train and test data using the SoftMax function.

The values obtained after this transformation constitute the output layer of our NN, this value may not be the best output, in this case a back propagation process will be applied to target the optimal value of error, the back propagation process connect the output layer to the hidden layer, sending a signal conforming the best weight with the optimal error for the number of epochs decided. This process will be repeated trying to improve our predictions and minimize the prediction error. After completing this process, the model will be trained. The classes of NN that predict future value base on passed sequence of observations is called Recurrent Neural Network (RNN) this type of NN make use of earlier stages to learn of data and forecast futures trends. The earlier stages of data should be remembered to predict and guess future values, in this case the hidden layer act like a stock for the past information from the sequential data. The term recurrent is used to describe the process of using elements of earlier sequences to forecast future data RNN can't store long time memory, so the use of the Long Short-Term Memory (LSTM) based on "memory line" proved to be very useful in forecasting cases with long time data. In a LSTM the memorization of earlier can be performed trough gates with along memory line incorporated. The following diagram-1 describe the composition of LSTM nodes.



Methodology

Data Preprocessing:

Acquiring and preprocessing the data for this project occurs in following sequence.

- Request the data from the Yahoo Finance Python Data reader library and stored.
- Remove unimportant features(date, Adj Close) from the acquired data.

	High	Low	Open	Close	Volume
0	7.619643	7.520000	7.611786	7.526071	352410800.0
1	7.660714	7.585000	7.622500	7.643214	493729600.0
2	7.699643	7.616071	7.664286	7.656429	601904800.0
3	7.686786	7.526786	7.656429	7.534643	552160000.0
4	7.571429	7.466071	7.562500	7.520714	477131200.0

- Normalized the data using MinMaxScaler helper function from Scikit-Learn.

- Using Rolling function found mean of every 100 and 200 data sets

- Split the dataset into the training (70%) and test (30%) datasets for LSTM model.

```
data_training=pd.DataFrame(df['Close'][0:int(len(df)*0.70)])
data_testing =pd.DataFrame(df['Close'][int(len(df)*0.70):int(len(df))])
print(df.shape)
print(data_training.shape)
print(data_testing.shape)
```

✓ 0.4s

```
(3231, 5)
(2261, 1)
(970, 1)
```

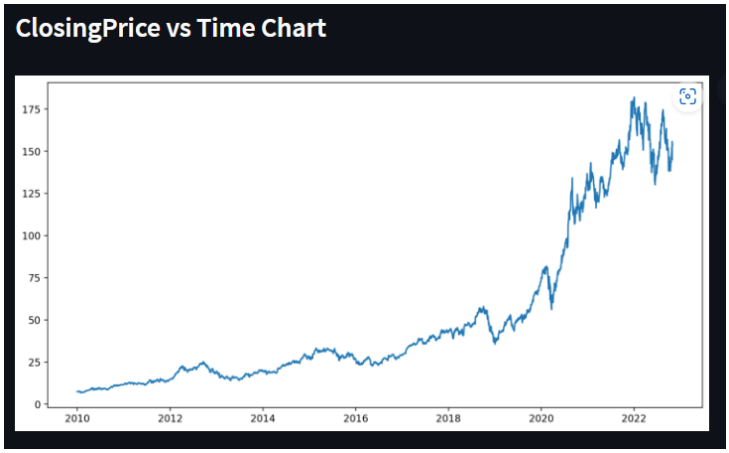
LSTM Model:

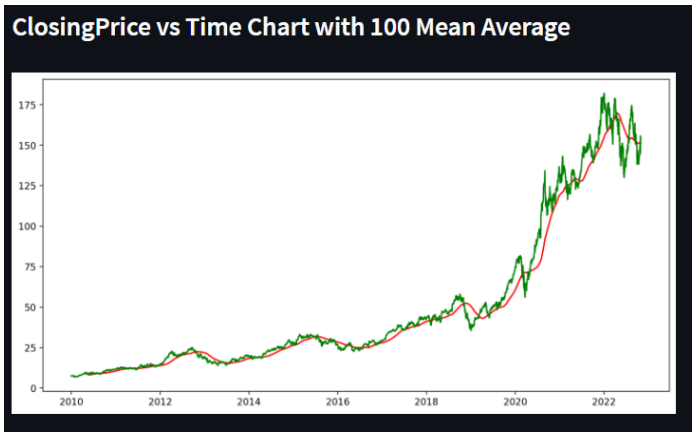
```
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential

model =Sequential()
model.add(LSTM(units=50, activation='relu', return_sequences=True,input_shape=(x_train.shape[1],1)))
model.add(Dropout(0.2))

model.add(LSTM(units=60, activation='relu', return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(units=80, activation='relu', return_sequences=True))
model.add(Dropout(0.4))

model.add(LSTM(units=120, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(units=1))

model.summary()
```

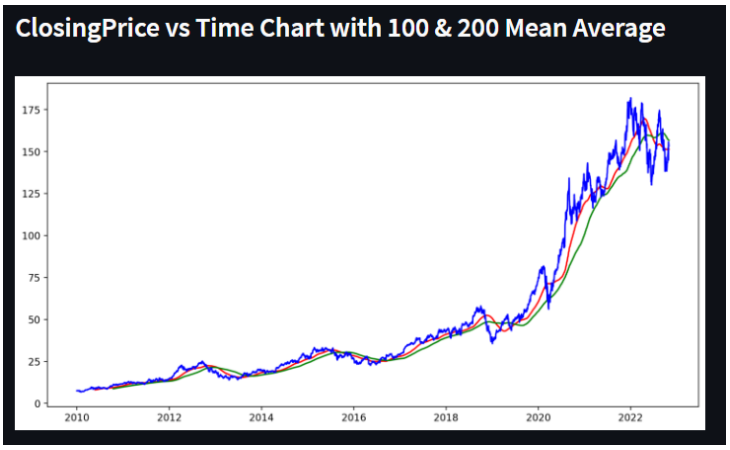
Model Summary:

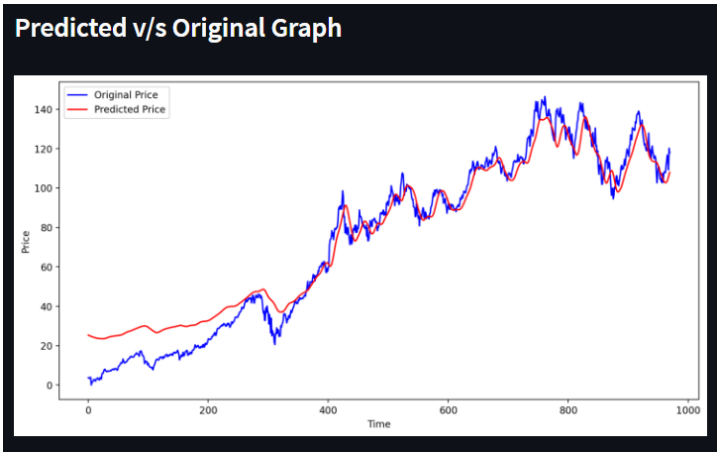
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (None, 100, 50)           10400

 dropout_4 (Dropout)         (None, 100, 50)           0

 lstm_5 (LSTM)               (None, 100, 60)           26640

 dropout_5 (Dropout)         (None, 100, 60)           0

 lstm_6 (LSTM)               (None, 100, 80)           45120

 dropout_6 (Dropout)         (None, 100, 80)           0

 lstm_7 (LSTM)               (None, 120)               96480

 dropout_7 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 178,761
Trainable params: 178,761
Non-trainable params: 0
_________________________________________________________________
```

Various graphs plotted:



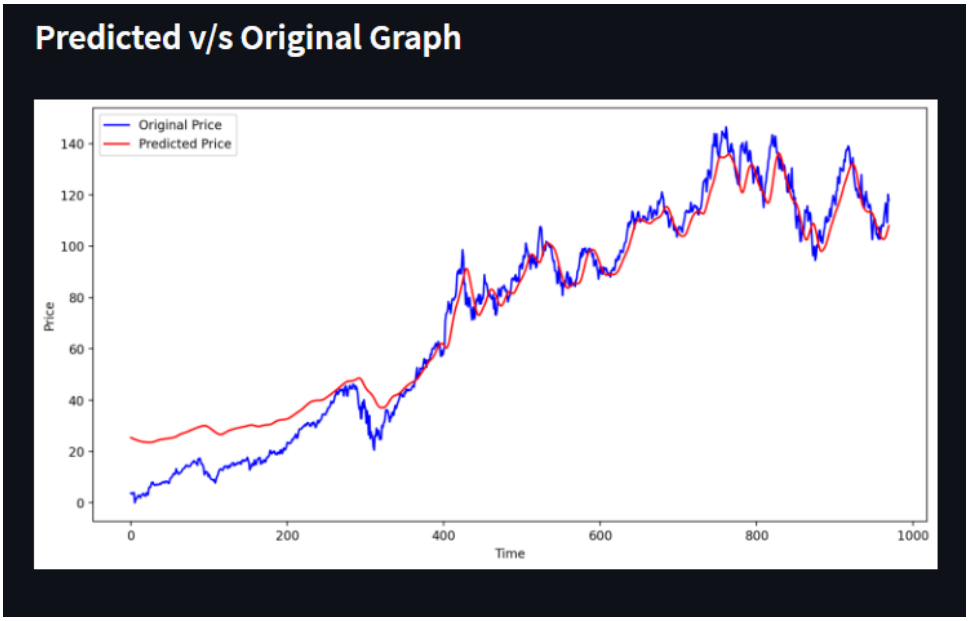






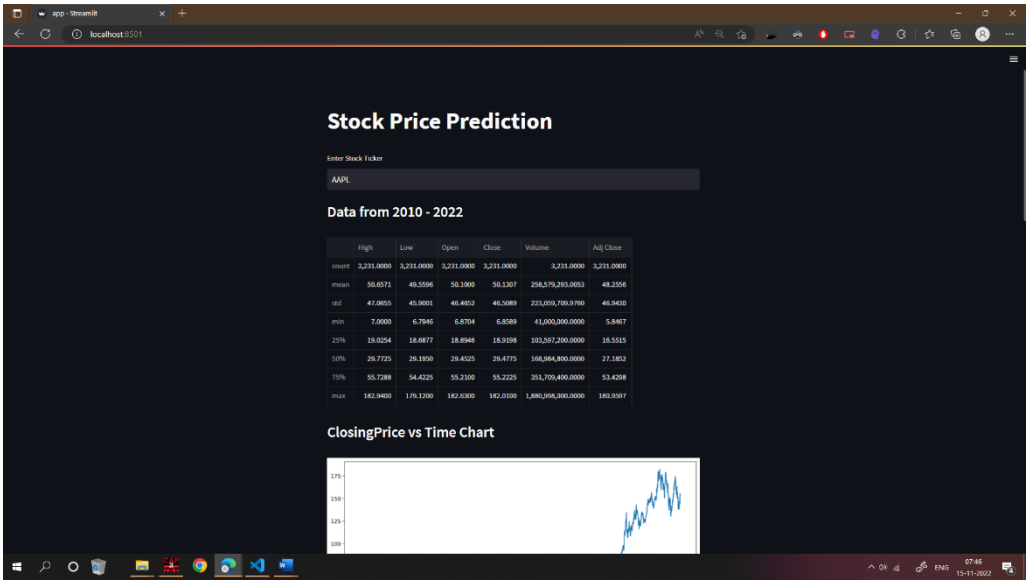
Results

We achieved the follow:



Web-App

After successfully creating the model and testing it out on multiple stocks we created a web application using streamlit. It helps user to enter any stock of his choice and the webapp will predict the stock price.



Conclusion

In this project, we are predicting closing stock price of any given organization, we developed a web application for predicting close stock price using LMS and LSTM algorithms for prediction. We have applied datasets belonging to Google, Nifty50, TCS, Infosys and Reliance Stocks and achieved above 90% accuracy for these datasets.

Future Scope

One main issue we might encounter in this case would be the sheer volume of available data which is not nearly enough for an optimal and profitable deep learning model. This could be solved by applying some data augmentation techniques to the already existing data sets in orders to grow it in size and make it viable for a deep learning project.

References

- "What Is Deep Learning?," mathworks, [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html.
- Tipirisetty, "Stock Price Prediction using Deep Learning," 2018. [Online]. Available: https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1639&context=etd_projects.
- V. Palaniappan, 2018. [Online]. Available: https://github.com/VivekPa/AIAlpha.
- “Stacked Autoencoders,” Ufldl,Stanford. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders.
- W. Bao, J. Yue, and Y. Rao, “A deep learning framework for financial time series using stacked autoencoders and long-short term memory,” PLOS ONE. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0180944#pone-0180944-g002.