Stock Price prediction using Big Data Visualization Technique

[1]Kent State University, Kent – Ohio, United States
e-mail: nroy2@kent.edu, pvyas2@kent.edu

Nibedita Roy[1], Priyanka Vyas[2]

November 22, 2022

Abstract

A financial instrument known as a stock indicates a company's ownership interest. A share of stock is a fraction of a corporation, which an individual purchases. Stocks of companies are purchased by investors who anticipate an increase in value. The stock can then be sold for a profit if this happens because it increases the worth of the firm as well. Businesses can raise money to invest in their operations and grow by issuing shares. Stocks are a means for investors to increase their capital and outpace inflation over time.

The stock market is characterized as volatile, dynamic, and nonlinear. Earlier, investors used to adopt the traditional approach, which involved seeking assistance from stockbrokers to buy or sell a stock but in the past few years exploratory data analysis methods have proved to be useful in providing statistical insight based on data and the graphical representation for visualization.

In this study, to understand and analyze the problem statement through a dataset, we have used Tableau to visualize the yearly price movement trend for a selected individual stocks and did a comparative study with Tesla's competitors. The data for this study is downloaded from Nasdaq and compiled separately for visualization purposes.

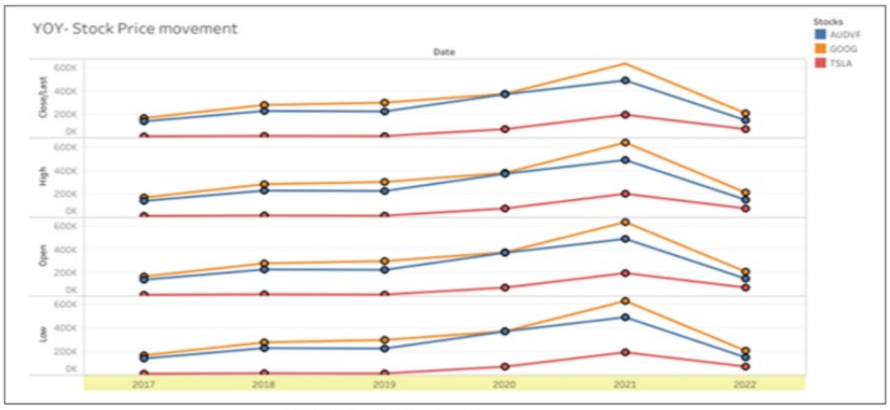


Fig1: Yearly Stock price movement

The individual stock selected for this study is Tesla stocks. Tesla stocks is a large index traded on the New York stock exchange and is a well known stock. The plot above shows the yearly trend visualization of price movement for Stocks of Tesla, Audi and Google. The dataset for the plot is a stock price history .csv file which is gathered and compiled for the time period of 2017 to 2022. The dataset contains a daily stock high, low, open and closing prices. It can be observed in Fig 1. plot that the stock price reaches a new high and low every time as the stock market is sensitive to the political and macroeconomic environment. There are features that cannot be included in data and can be considered as noise which means an incomplete information gap between past stock trading price and volume with a future price. The graph shows a rebound in 2021 post covid and again a low in 2022.

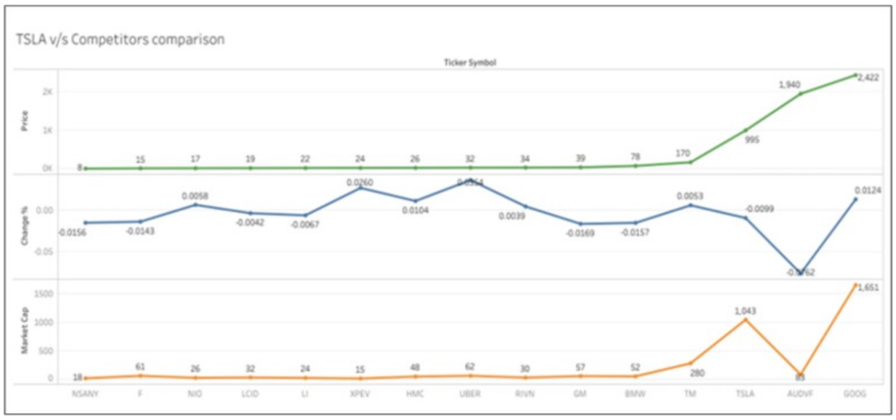


Fig 1.1: Tesla Competitors Comparative study of Stock price movement

Fig 1: Tesla Competitors Comparative study of Stock price movement

The graph above shows Fig 1.1. plot having stock data dated 04/24/2022 It is a comparative study of the parameters using Tableau to study the stock price movement in line with companies Market Cap,

While devising this comparison as the focus was mainly on individual stocks i.e.Tesla, the plot shows a comparative between direct competitors of Tesla i.e. from the automobile industry and Indirect competitors from IT services. The plot shows a rise in stock prices of Tesla, Audi and Google and compared to the market capital of Tesla and Audi, Tesla has more capital infusion compared to peers as Tesla works on the ability to generate recurring sales and potential to add to profits. Tesla is an indirect competitor of Uber and Google and the reason is because Uber and Google are also working on self-driving car projects. Through initial assessment using tableau graphs it was observed that accurate stock price prediction is challenging because of multiple factors that affect the stock price, like political, global economic conditions, unexpected events, a company's financial performance etc. but it also crucial to predict as the supply-demand balance is driven by market sentiment which in turn affects the stock price trend.

Through stock price prediction investors can increase the rate of investment and business opportunities in the stock market by devising an algorithm to predict the short term price of individual stock. In this study, a novel method is proposed to predict and visualize stock price for individual Tesla stocks using a neural network. The stock analysis in this project is based on a 10-year stock trading record of Tesla Inc., where we load and read the data, clean and prepare it for further analysis, then visually portray the data analysis , trends , forecasting using various ways with the pandas and seaborn libraries. For now, we have equally distributed the workload among the team for the data analysis, visualization, documentation and presentation.

0.1 Data Analysis and Prediction Method

Stock data is a time series data and its prediction dependency lies on four types of component: trend component, seasonal component, cyclical component and irregular component. The stock data study requires a specific way of analyzing a sequence of data points collected over an interval of time. In this study the historical time series stock data tracks the movement of selective data points, like, security price, over a specified period of time (10 years) with data points recorded at regular intervals. The workflow for the study is as shown in figure 2 below:

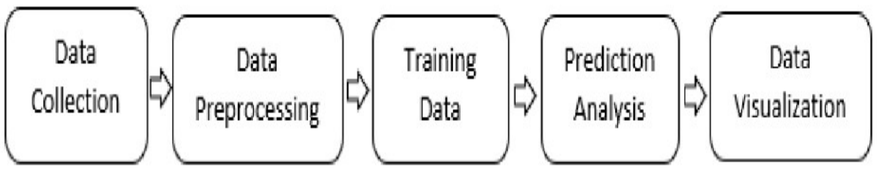


Fig 2: Illustration of proposed methodology

Fig 2: Illustration of proposed methodology

The first step in the Machine learning model is the Data preparation stage which includes sub stages i.e. Data collection and Data Pre-processing. The data is collected from the source and is pre-processed using transformation techniques where the raw data is modified into desired format. The second step in the process is the data training and learning stage which does the feature extraction. The final step is prediction analysis and data visualization where the trained model identifies the pattern that fits on the historical data and uses it to predict future value over a period of time. The process develops a model based on historical data and applies it to predict future value of stocks.

0.2 Data Collection

The financial data is mostly available in either structured or unstructured format and is accessible as historical data or real-time data in the public domain. The real-time data is not available from public APIs and requires to be purchased. Hence, structured historical data with attributes date, open price, high price, low price, closing price, Adj close price and volume is used in this study.

The code for data collection and download has been shared in the git repository for perusal. In this study the framework explored for coding are python, tensorflow using google collaboratory and the libraries like panda, panda-dataframe reader,numpy and yahoo finance(yfinance). The library pandas organizes data in data frames i.e. table structures and pandas-data reader is used to import the financial data from public domain into python data frames. YahooFinance (‘yfinance’) provides historical stock prices from yahoo finance and is one of the widely used API for stock data study as it provides the data from New York Stock Exchange (NYSE) and Nasdaq Stock Exchange (Nasdaq). Then, a ticker object for data (i.e. TSLA-Tesla stock) that we want to get for is created to get the financial data information related to the stock. The function yf.ticker.history() is used to specify the argument period i.e. 10 years, for which the historical data is required in the study.

In this study, it is learned that using pip install ‘yfinance’ library installation is easy and python friendly also it provides data for free whereas Quandi and Nasdaq requires registration and provides data for a price through their premium and free options. The coding for this stage was explored using Nasdaqdatalink and Quandi also, but data fetching from the source failed repeatedly due to API, ticker or path issues and the historical stock price movement is visualized by plotting a line chart using Matplotlib plot method.

0.3 Data Preprocessing and Training

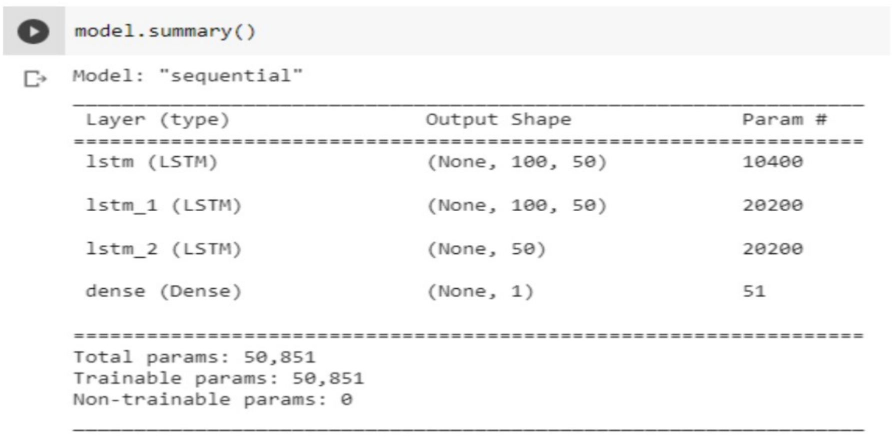
Using Matplotlib plot method a line chart is created for the historical close prices of TSLA and the plot shows an upward trend in terms of the rise in the stock price. To build an LSTM model for price prediction, first split the dataset into training set and test set and then normalize the data using Scikit-Learn MinmaxScaler function to transform the values so that the values are ranged from 0 to 1 and then reshape the normalized data into two dimensional array.

Further split the dataset such that the training size of the dataset is 65

An empty list for a sequence of feature data(dataX) and sequence of label data (data Y) is created and for the window size of $n_s tepitisconvertedintoaNumpyarraywhichisanacc$
$dimensionalarraywhichistheacceptableinputdatafortheLSTMmodel.Thetrain(X_train, y_train)andt$

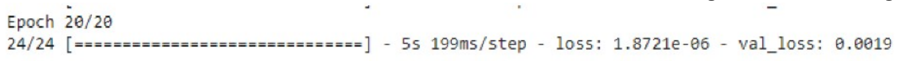
0.4 Design LSTM model and Algorithm

All important and relevant libraries required to train the model are imported during Milestone 2. Tensorflow, which is an open source machine learning library, is imported to set up the LSTM network architecture. To build the model, first define the Sequential model that consists of the four layers. The first and second added for LSTM layer has 50 network units in the model and the return sequence is set to true so that the output of the layer will be another sequence of the same length. The third layer added is again an LSTM layer with 50 network units and lastly the densely connected layer that provides the output of one network unit. The summary of the LSTM model is as below:



```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 lstm_1 (LSTM)               (None, 100, 50)           20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

As shown above the total trainable parameters are 50,851 and to train the model by fitting it with the training set,first compile the model and set adam optimiser and a loss function MSE i.e. Mean square error. Finally, the model is trained by fitting it with the training set and running it for 20 epochs and batch size 64 to calculate the validation loss and training loss. With limited parameters and the loss and val$_{l}ossvaluesitcanbesaidthedesignedmodelisagoodmodel.$



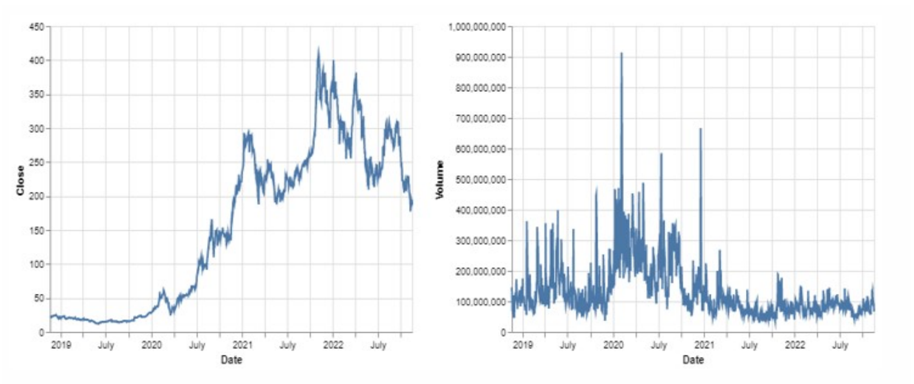
```
Epoch 20/20
24/24 [==============================] - 5s 199ms/step - loss: 1.8721e-06 - val_loss: 0.0019
```

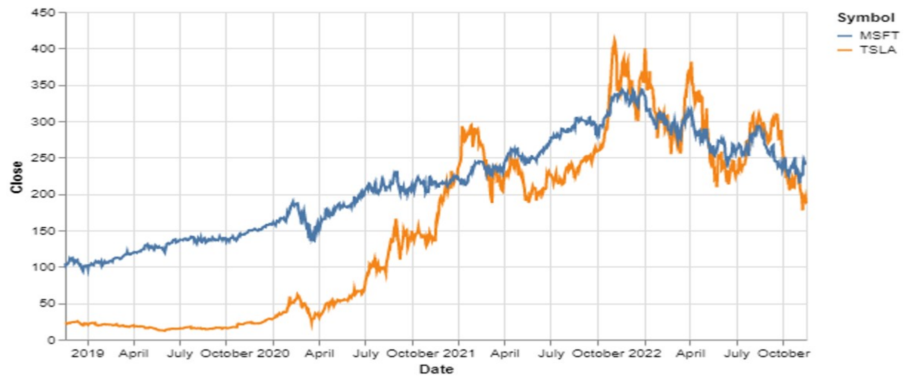
For the next milestone, the focus is to assess the performance of the trained model and visualize the results in line with the epochs and model loss.

0.5 Model Evaluation

Stock price analysis is an example of time series analysis and is one of the primary area in predictive analytics. The time series data contains measurements or observations attached to sequential time steps. In case of stock prices, the time stamps can seconds, minutes or days.

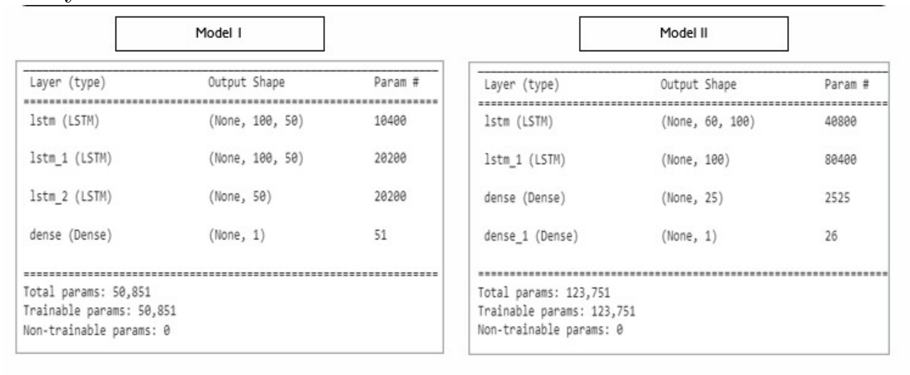
In this project we have used pandas for data manipulation and altair, matplotlib for data visualization. For visualization through altair we have used the data reader API of Pandas. We first import the dependencies necessary for the project and then used data reader to create Pandas dataframe that contain stock price data.





Altair makes it simple to have multiple plots in one visualization. As we can see in the slide we can plot closing price movement and volume for TSLA stocks using a logical operator.The data is for time period 2018-2022 as Altair has a limitation to visualize upto 5000 records.

For performance evaluation, we have build two models using LSTM and added layers.


Model I

```
Layer (type)                Output Shape              Param #
=================================================================
lstm (LSTM)                 (None, 100, 50)           10400

lstm_1 (LSTM)               (None, 100, 50)           20200

lstm_2 (LSTM)               (None, 50)                20200

dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```

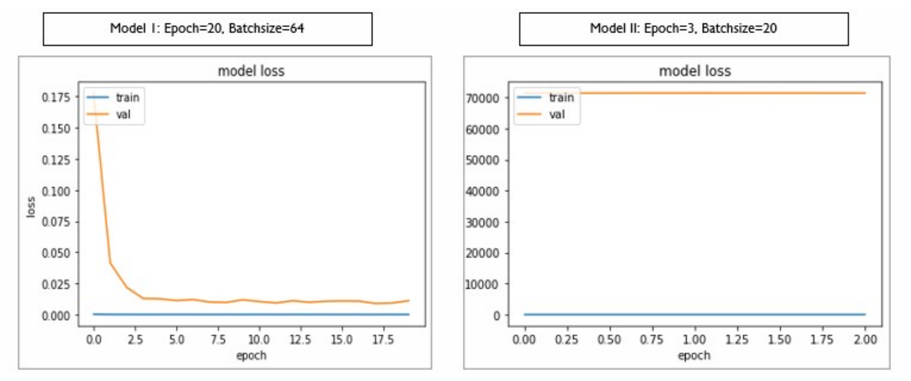
Model II

```
Layer (type)                Output Shape              Param #
=================================================================
lstm (LSTM)                 (None, 60, 100)           40800

lstm_1 (LSTM)               (None, 100)               80400

dense (Dense)               (None, 25)                2525

dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 123,751
Trainable params: 123,751
Non-trainable params: 0
```

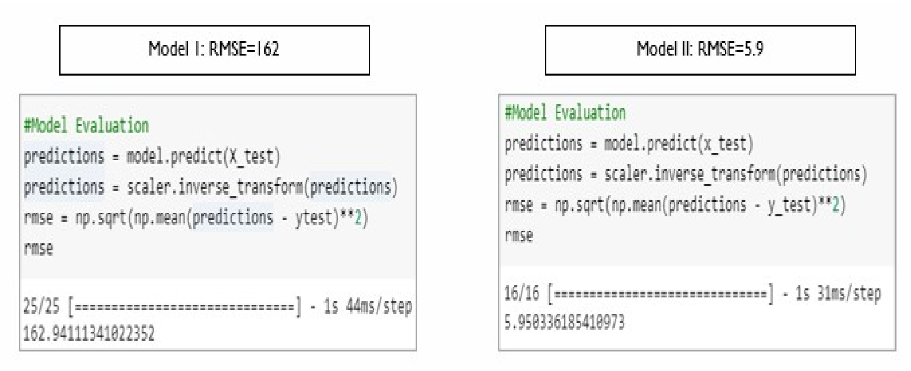
Model 1 has 3 LSTM layers with 100,100 and 50 neurons resp and 1 Dense layer with output dimension 1.

Model 2 has 2 LSTM layers with 60 and 50 neurons resp and 2 Dense layer with output dimension 25 and 1 respectively.

Further, Model 1 is updated with with a batch size of 64 and epochs 20, the plots shows that after a certain epochs the model manages to follow the trend and Model 2 is updated with with a batch size of 20 and epochs 3, the plots shows that the model manages to follow the trend from the first epoch run.



The plots in the figure above are the diagnostic plot that shows the model behaviour. All result and analysis plots has been made using Matplotlib library. The training loss v/s val loss(test loss) It is observed that the performance of the model is good on both the train and validation set.The plot shows the train and val loss decrease and stabilise around the same point. The plot of train and validation loss for model 2 is showing the characteristics of an underfit model. The performance of the model may be improved by increasing the capacity of the model, such as number of neurons in the layers or number of hidden layer(LSTM layer).



Model I: RMSE=162

```
#Model Evaluation
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
rmse = np.sqrt(np.mean(predictions - ytest)**2)
rmse

25/25 [==============================] - 1s 44ms/step
162.94111341022352
```

Model II: RMSE=5.9

```
#Model Evaluation
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rmse = np.sqrt(np.mean(predictions - y_test)**2)
rmse

16/16 [==============================] - 1s 31ms/step
5.950336185410973
```

We evaluated the model for predictions using the loss function root mean square error and it is observed that for Model 1 = RMSE is 162 which means the model badly overfits the data but it tests well in the sample and has little predictive value when tested out of the sample size and Model 2= RMSE is 5.95 that shows model is able to fit the dataset and is giving better predictions.

RMSE score shows how far prediction falls from the measured true values using Euclidean Distance. The RMSE statistic provides information about the short-term performance of a model by allowing a term-by-term comparison of the actual difference between the estimated and the measured value.

The smaller the value, the better the model's performance.

From the plots it is observed that both the models performed fairly good. We can fairly follow the unexpected jumps and drops however, for the most recent date stamps, we can see that the model expected (predicted) lower values compared to real values of the stock price.