Rochester Institute of Technology

B. Thomas Golisano College of Computing and Information Sciences

Master of Science in Information Sciences and Technologies

Project Approval Form

Student Name: **Siddharth Chauhan**

Project Title: **Predictive Analysis on Stock Market Data using Sentiment Analysis and Long Short-Term Memory (LSTM) Neural Network.**

Project Committee

Name

Prof. Erik Golen

_____

Chair

Prof. David Patric

_____

Committee Member

“Predictive Analysis on Stock Market Data using Sentiment Analysis and Long Short-Term Memory (LSTM) Neural Network”

by

Siddharth Chauhan

Table of Contents

1. ABSTRACT:

Stock market prediction has been one of the most interesting use cases of machine learning for a long time. Stock market prediction involves both fundamental and technical analysis of a particular stock. Fundamental analysis of stocks itself depends on multiple factors – physical factors, behavioral economics, news about a particular stock, demonetization, monetary policy, natural disaster, etc. Existing studies have shown that there has been a strong correlation between news article, blogs, and stock price momentum. The project aims to use text mining (Natural Language techniques) to predict stock market momentum and time-series data to further improve the accuracy of prediction. The time series model will use Long Short-Term Memory (LSTM) neural network to predict future 30 days stock close price which will cover the technical analysis of stock momentum. The hybrid approach of using both sentiment analysis and the LSTM model will result in high predictability of the stock market.

Keywords- NLP, LSTM, Neural Network, Stock Market, Sentiment Analysis.

2. INTRODUCTION:

The stock market has a direct impact on the country's economy, financial institutions, and individual investors. The growth of the stock market of a particular country gives the projection of financial investments both domestic and international. However, due to the highly volatile nature of the stock market and dynamic information flow, the prediction of the stock market remains a difficult topic in the machine learning domain. Several attempts in the past have been made to accurately predict the momentum of the stock using Support Vector Machine (SVM), Linear regression, and other machine learning techniques but due to the limitation of these models the accurate prediction of stock momentum has not been achieved. Moreover, studies have shown that there has been a strong correlation between news articles and stock prices.

The project aims to adopt a hybrid approach where in the first stage, sentiment analysis on multiple news feeds from different financial websites (Bloomberg News, Yahoo finance, Google finance, MarketWatch, Reuters, finviz) will be performed. Financial data will be gathered by calling APIs, parsing the data, and finally performing sentiment analysis using NLP techniques. The compounded score obtained for a particular stock for different news feeds for a particular day will then be averaged to obtain the final score. Based on this average value predictions can be made for a stock momentum in future. Finally, the realized predictions can then be analyzed visually with the help of Matplotlib, Plotly, or Seaborn.

In the second stage, the time series historic data will be gathered from multiple sources will be used as input for Recurrent Neural Network (RNN). For the technical analysis, Recurrent Neural Network (RNN) algorithm LSTM will be used to analyze at least 60 days of stock price data. Long-Short Term Memory has been one of the most successful RNN architecture and since stock data is time-based meaning present-day prices depend on previous day prices, using RNN is the most suitable model for stock price prediction.

Predicting stock market momentum is a difficult task, given the number of parameters involved. If one can predict the stock market momentum accurately one can easily increase the profits and reduce the losses. The stock market dynamic relies primarily on two analyses, fundamental and technical. The fundamental analysis covers the behavioral patterns of individual investors, financial institutions, and news circulating in the market. Hence it becomes extremely important to analyze the sentiments for performing fundamental analysis. Moreover, technical analysis is mostly performed on day-to-day stock prices. Since stock data is time-based it is useful to do a time-series analysis to perform technical analysis.

3. RELATED WORK:

Over the period researchers have explored many ways and methods for predicting the stock market, but recent studies [1] have shown growing interest in using sentiment analysis for stock forecasting. Some studies have also proven a strong correlation between stock prices, news articles, and investor's sentiments. Broadly sentiment analysis can be categorized into lexicon-based analysis and machine learning-based analysis. Lexicon-based analysis can be further classified into corpus-based, and dictionary-based. Machine Learning based analysis can also be classified into supervised and unsupervised.

Dev Shah et al [2] in their research used a dictionary-based approach where they used a python library and transformed the text corpus into numerical vectors. The approach used finally converts the words as positive and negative and assigns a sentiment score for each token of the word. One of the major drawbacks of this approach was that the tokens used for calculating sentiment scores have no weight assigned to them which can lead to false sentiment prediction.

In [3] the author has used bag-of- word approach to perform sentiment analysis using Twitter data and news articles. In this approach, the author labeled the tweets and news articles' headlines as positive, negative, or neutral. For pre-processing data, the author used tokenization, lemmatization, and n-gram methods also the use TF-IDF weighting schema is being used. Although, this method was successfully implemented the bag-of-words approach it ignored sentence structure due to which it failed to capture sentiments. Also, the method failed to capture sentiments of informal text like slang and acronyms.

In 2015 Alostad and Davulcu [4] used a hybrid approach to perform sentiment analysis using N-gram feature extraction They built the document matrix by collecting news articles from the NASDAQ website and applied OpenNLP for sentence extraction. Alostad et al [4] also applied classification techniques like logistic regression and Support Vector Machine (SVM) over the n-gram document matrix to improve the predictability of their model. Although the hybrid method used by the author to predict the stock price sentiments was unique, the document-level sentiment analysis could not improve the accuracy of the prediction.

In 2015 Chen et al [5] proposed an LSTM- based approach for predicting the Chinese stock market. They labeled 30 days stock data price using earning rate, which was calculated averaging the 3-day closing price. The model proposed uses a single layer for input/output, multiple LSTM layers, and a dense layer. The feature used for learning includes historic price data of 30 days and the historic price data for market indexes. One of the drawbacks of this approach was that it did not

consider the impact of technical analysis and fundamental analysis (GDP, oil prices, etc.). Also, setting the right threshold for effectively excluding the extreme high and extremely low prices resulted in low accuracy for this method.

In 2019 Nikou et al [6] compared different machine learning algorithms such as Random Forest, Support Vector Machine (SVM), Deep Learning, etc. for efficient prediction of the stock market. The result indicated that the LSTM model, a special case of Recurrent Neural Network (RNN) performed better as compared to other models.

Overall, the papers mentioned above have done a significant amount of work explaining the sentiment analysis and use of LSTM as one of the most effective RNN architecture [6] when it comes to predicting sequential data such as stock prices. However, Dev Shah et al [2] focused on a single sector (Pharmaceuticals) which does not give the accurate picture of other sector performance also the news source for sentiment analysis was limited to two which resulted in less accuracy for ~70%. The efficiency of the sentiment analysis can be further improved by selecting a comprehensive data source. Hedge et al [1] mentioned the use of both Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) for predicting the stock prices, the paper also mentioned Root Mean Squared Error (RMSE) as one of the performance measuring metrics. However, it uses a much less efficient optimizer "Nadam" and Epoch value of 200. The paper also considered using news and twitter data to predict the stock prices but fails to do so since not many companies new and blogs are available frequently, the paper uses mainly IT companies and banking data since they are readily available which makes the prediction less efficient. In 2015 Chen et al [5] mentioned the use of LSTM as one of the most efficient RNN, the paper used previous 30 days as the time stamp for learning the model. One of the drawbacks for the paper was that it is only restricted to stock market analysis of China.

The project tries to improve the efficiency of both sentimental analysis [6] and LSTM models [1][5] mentioned above. For improving the accuracy of sentiment analysis, a wide news source is selected which can help in generating a bigger dictionary of words and resolving the limitations mentioned in [2]. Also, a more learned and trained library VADER from NLTK is selected to improve the polarity scores generated. For improving the LSTM model, "Adam" optimizer is selected which is more efficient than "Nadam" [1]. Also, instead of using 30 days' timestamp [5] the project uses 100 days' timestamp to generate a bigger data set and scaling values. The limitations of [5] are also resolved using this project as different technical indicators such as Moving Average Convergence Divergence (MACD) 100 and 200 are selected for increasing the predictability of stock price momentum. One of the unique features of this project which has not been covered in previous work is predicting the next 30 – 60 days stock price momentum.

4. METHODOLOGY:

4.1 Phase 1: Sentiment analysis on news data

1. **Data Pre-processing:** The data for sentiment analysis will be collected from multiple sources. The news headlines will be collected from(https://finviz.com/) using the python beautifulsoup module. The field which is not related to financial data will be removed during this phase also new articles headlines will be considered for now and not the entire article. The Ticker values will be separated based on timestamps. Since the new articles come with a different timestamp. For example, some of the news has timestamp as yyyy:mm:dd: hh: mm whereas some timestamps are in the format of yyyy:mm: dd. Similarly, tweets from Twitter will also be collected for specific tickers using "SNSCRAPE" library. The duplicate tweets will be removed using drop duplicate function of pandas library. The data received after pre-processing and transformations will be stored in a dictionary or panda's data frame.

2. **Sentiment Analysis:** The data frame obtained from the previous step will hold the stock name, date, time, and articles related to stock. The sentiment analysis will be performed on the news articles using NLTK Vader lexicon. Vader uses the polarity scores method to generate the compounded score for positive, negative, or neutral. The compounded value is a value between [-1, 1] where -1 means extremely negative and +1 means extremely positive. Next, the compounded score for all the articles for a particular stock will be done, and the mean value will be taken for it.

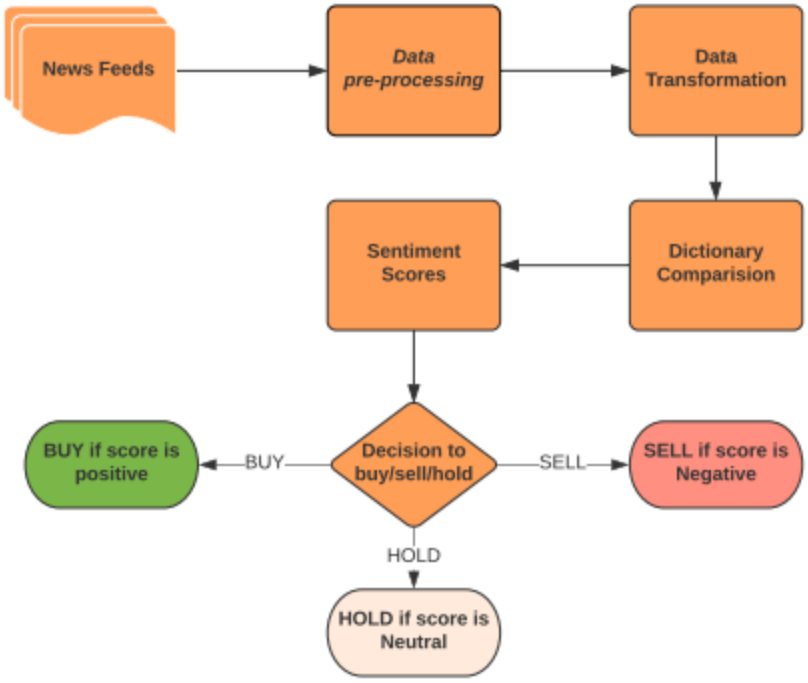


Fig [1]: Performing sentiment analysis on news feeds

4.2 Phase 2: Long Short-Term Memory (LSTM)

4.2.1 RNN Architecture

Recurrent Neural Network (RNN) have inbuilt memory where they can remember previous input when a huge sequential data is given. The difference between RNN and CNN is that CNN doesn't have a memory cell and as a result they cannot remember previous inputs. The diagram below shows xt as the input, ht as the output and a loop which can be used to pass information from one step to another. Fig [3] shows multiple cells connected and passing information to one another. The first output is dependent on the first input, but the second output is dependent on both first and second input and so on.

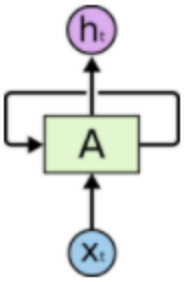


Fig [2] Recurrent Neural Network loop.

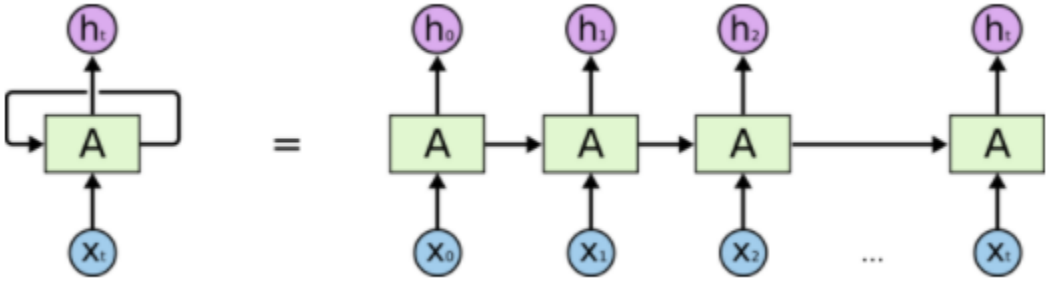


Fig [3] An unrolled recurrent neural network

Although, RNN appears to present a perfect solution for remembering previous information it suffers from long term dependency problem fig [4]. Which appears when the gap in learning and remembering information is big.

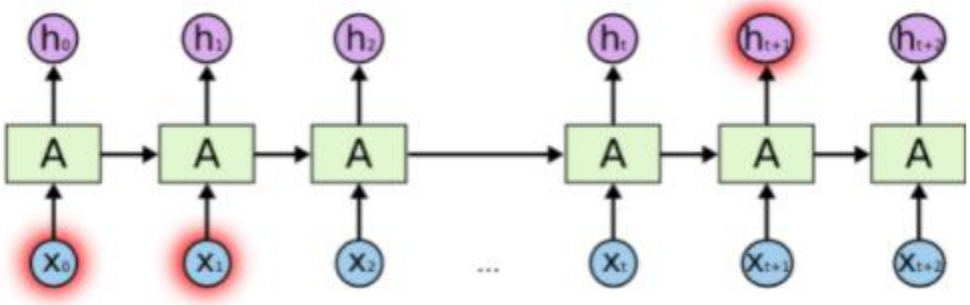


Fig [4] Long term dependency problem

There are two main long term dependency problem

1. Vanishing gradient
2. Exploding gradient

RNN works on the principle of back propagation as can be seen from Fig [5]. As we receive the output it is compared with the actual output in case there is an error the weights of

neural network are updated. If the error is less than 1 for 1st input cell, then after getting multiplied with the learning rate and squaring the result will also be less than 1 and will tend towards 0. By the time we reached at the last cell the error is negligible this is termed as vanishing gradient. Whereas in case the error value is more than 1 in that case the square after multiplying with learning rate and taking square will be a much higher value and can result in exploding gradient problem. To overcome the problem of vanishing gradient, exploding gradient Long Short-Term Memory (LSTM) is selected as an appropriate solution.

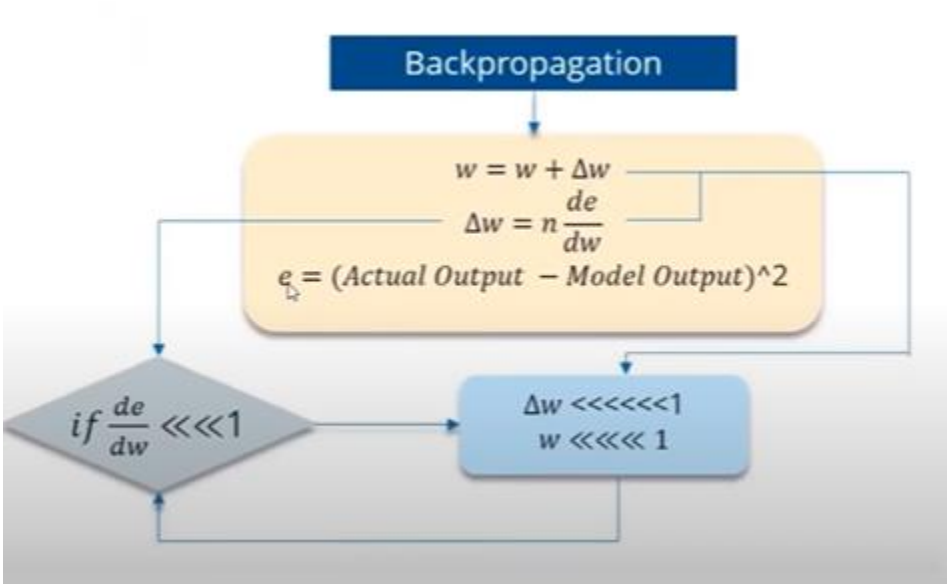


Fig [5] Back Propagation in RNN

4.2.2 LSTM Architecture

LSTM are special types of recurrent neural networks which are capable of learning long-term dependencies. They are designed to avoid long-term dependency problem. Fig [6] and [7] shows the difference between traditional RNN and LSTM layer.

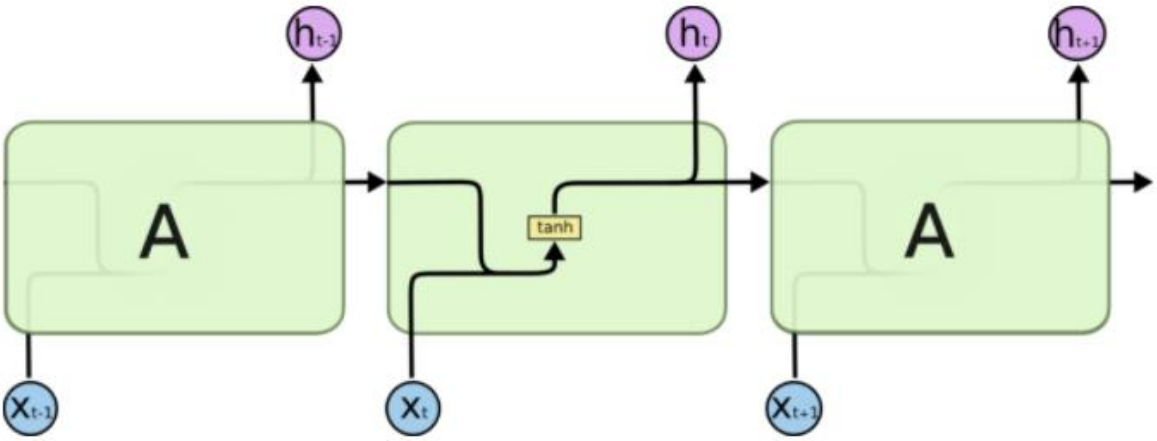


Fig [6] Standard RNN with single layer.

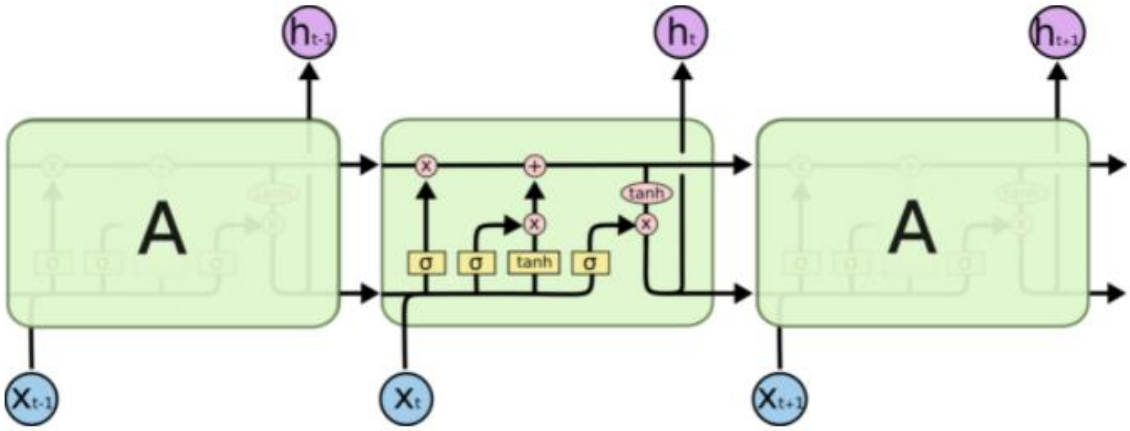


Fig [7] LSTM with four interactive layers.

LSTM cell contains two major functions, tanh and sigmoid. The sigmoid can output values from 0 to 1 and can be used to forget or remember information. Whereas tanh is used to overcome vanishing gradient problem.

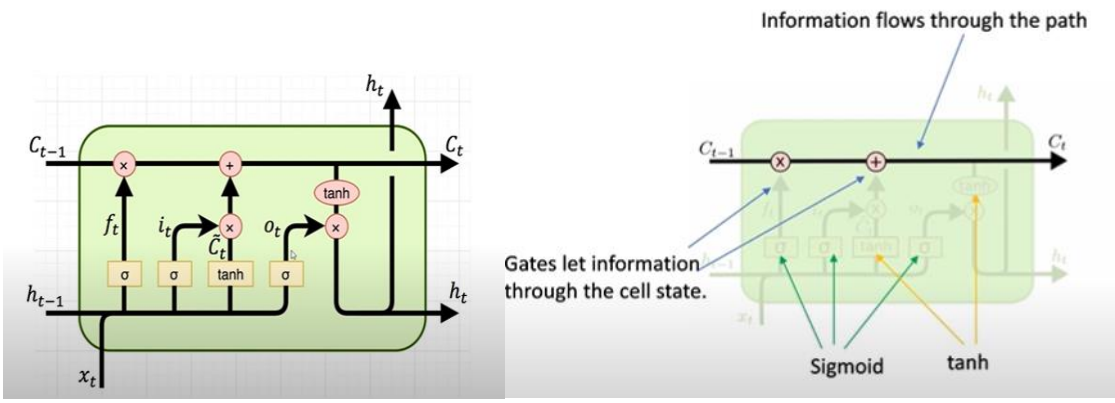


Fig [8] Sigmoid and tanh functions of LSTM

An LSTM has 3 gates Fig [9] for protecting and controlling the cell state:

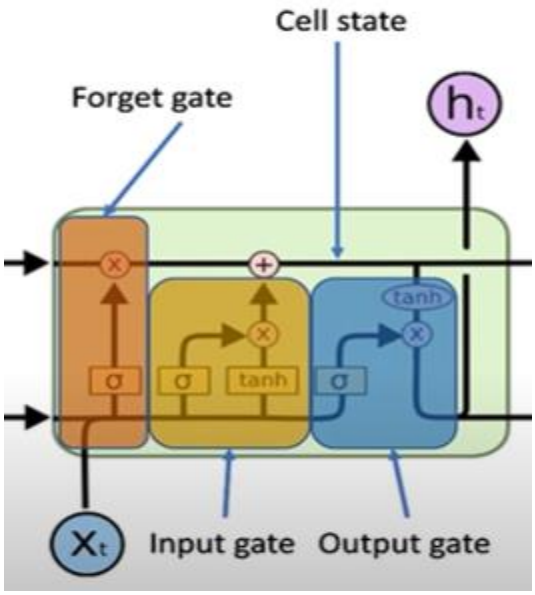
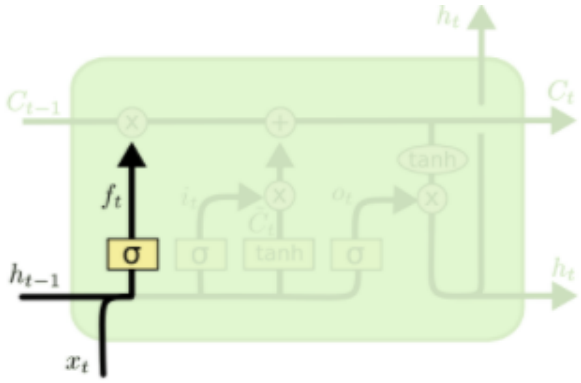


Fig [9] Three gates in LSTM cell

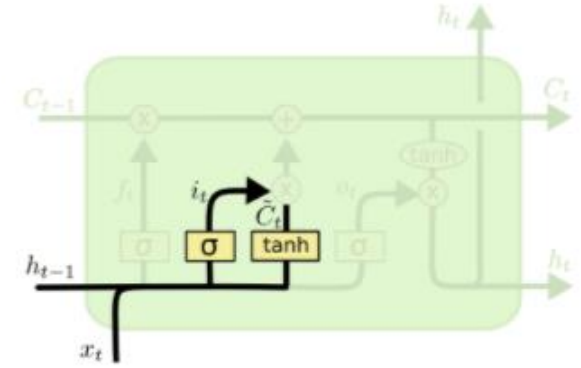
1. **Forget gate:** It is used to decide what information we need to throw out from the cell state, this decision is made by the sigmoid layer. It outputs a number between 0 and 1. A value of 0 represent to "throw away everything" and a value of 1 represent "keep everything"



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

Fig [10] First Step of LSTM

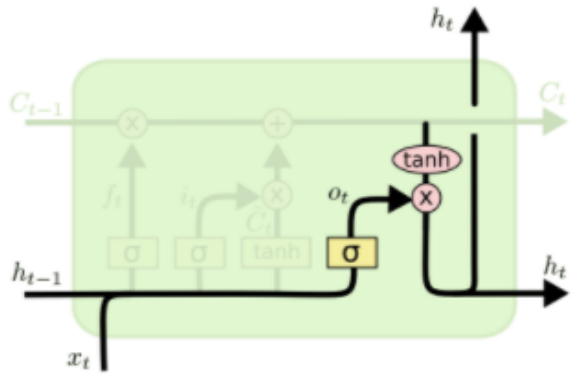
2. **Input gate:** This gate decides what new information we are going to store in the cell. It consists of two layers. Sigmoid layer which decides which value to be updated and tanh layer, which creates a vector of new candidates. Finally, the two results are combined to be passed to the next cell.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig [11] Second step of LSTM

3. **Output gate:** Finally, the output gates decide what part of current cell makes to the output this is decided by first running sigmoid layer. Then the cell state is passed through tanh to push the values between -1 and 1.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fig [12] Third step of LSTM

4.2.3 LSTM Implementation

1. Data Preparation and Scaling

The input data for the LSTM model is prepared by collecting the historical stock price from January 2010- December 2020 for three specific tickers (Microsoft, Google, and Amazon) by calling API endpoints from Tingo (https://api.tiingo.com/), TradingView, and Yahoo Finance. The input for the LSTM model will consider only stocks closing and opening price. LSTM is sensitive to the scale of the data, so the input will be transformed using MinMax scaler between [0,1] with the help of the NumPy library.

2. Dataset Splitting and Model Construction

The input data from the previous step will be further split into training and test data. Initially, multiple training and test will be taken into consideration to find the best possible split. To obtain better accuracy during model fitting different values for parameters will be taken into consideration for example Epoch keeps the count of how many times the data is passed through the neural network. In this project since the data is huge the epoch value is taken as 40. Also, the project will make use of the Adams optimizer which is mostly used for building RNN architecture. The detailed flow of model construction is shown in fig [13]

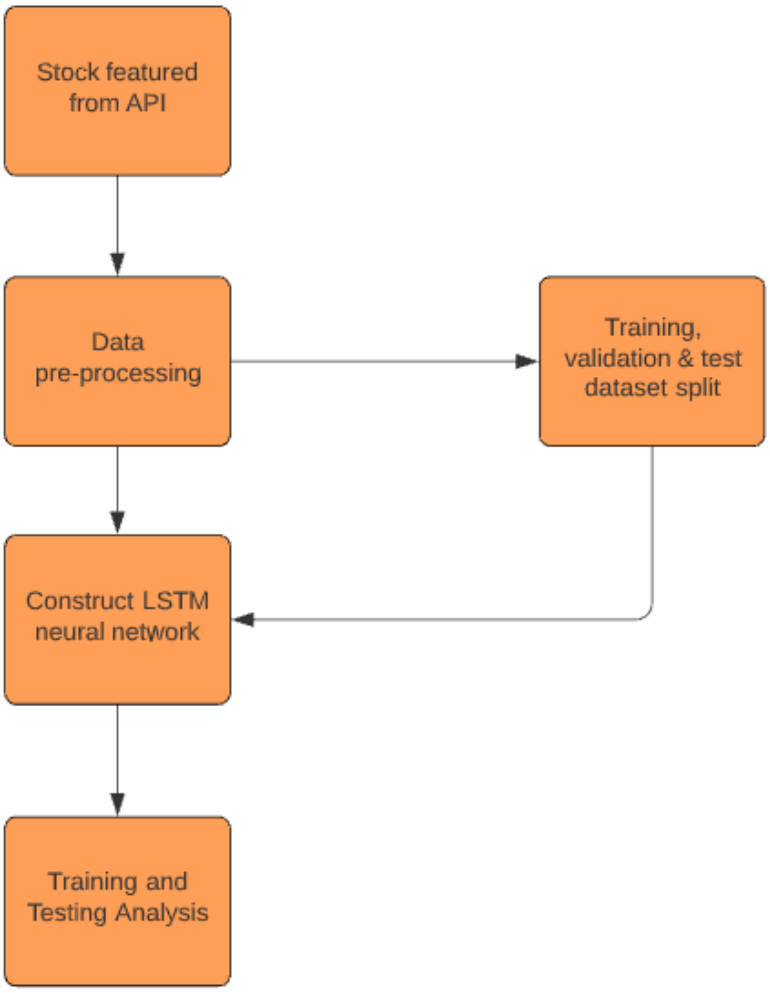


Fig [13] Model construction process

3. Data Pre-Processing and Model Evaluation

This phase will consider taking custom timesteps for hyperparameter tuning of LSTM Model and converting train and test data to independent (x-train, y-train) and dependent (x-test, y-test) features. The train and test data will be evaluated separately by running the LSTM model. The project will use Root Mean Squared Error (RMSE) as the key performance index to evaluate train and test data. The lower the difference between rmse_train and rmse_test the better the model will be. Finally, the output of the train and test data will be plot using Matplotlib which will show the prediction graph of the stock price for test data.

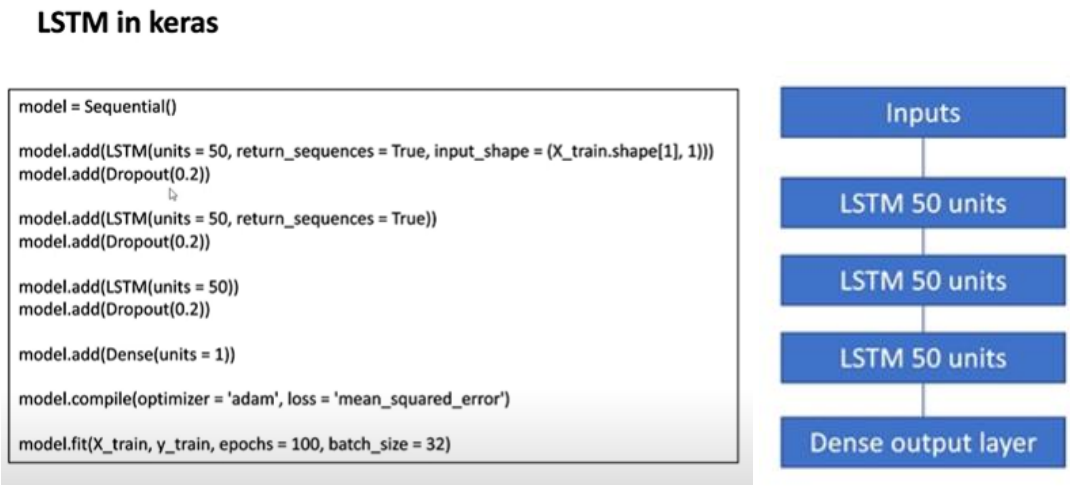


Fig [14] LSTM proposed blueprint

4.3 Phase 3: Combining Sentiment Analysis and Long Short-Term Memory (LSTM)

1. Data Collection and Pre-Processing

The data collected from sentiment analysis from phase 1 using news headlines and VADER (Valency Aware Dictionary and Sentiment Reasoner) library will be used to calculate the compounded score. For calculating the sentiments of articles/news content another Python library Text Blob [10] will be used which will generate a polarity score ranging from [-1, +1] where +1 represent the positive sentiments and -1 represent negative sentiments. The combination of both compound score and polarity score will help improve the accuracy of sentiment analysis.

Finally, the historical stock data prepared for LSTM model during phase 2 will be combined with sentiment analysis data. The new data set constructed will then be transformed and scaled to be used as input for new LSTM Model.

2. Model Construction and Evaluation

The LSTM model constructed during this phase will be similar to the LSTM model constructed during phase 2 with the key difference of combined input data from phase 1 and phase 2. The combined output from Sentiment analysis and LSTM will help improve the accuracy of the LSTM model constructed during this phase. Fig [15] shows the detailed flow of model construction.

For evaluating LSTM Model, the dataset prepared from the previous step will be split into train (65%) and test (35%) dataset for model training and prediction analysis purposes. The Key Performance Indicator (KPI) for phase 3 will be RMSE (Root Mean Squared Error) or MAE (Mean Absolute Error). Root mean squared error will represent the average difference between predicted and actual values. Smaller RMSE and MAE will show that

the stock prices predicted are closer to actual stock prices. Finally, the model with smaller RMSE will provide better prediction accuracy.

The comparison will be made using LSTM Model constructed (phase 2) without using sentiment score and the LSTM model constructed (phase 3) using sentiment score. The RMSE score for both phase 2 and phase 3 will be compared. This will further prove the correlation of sentiment analysis on stock prices.

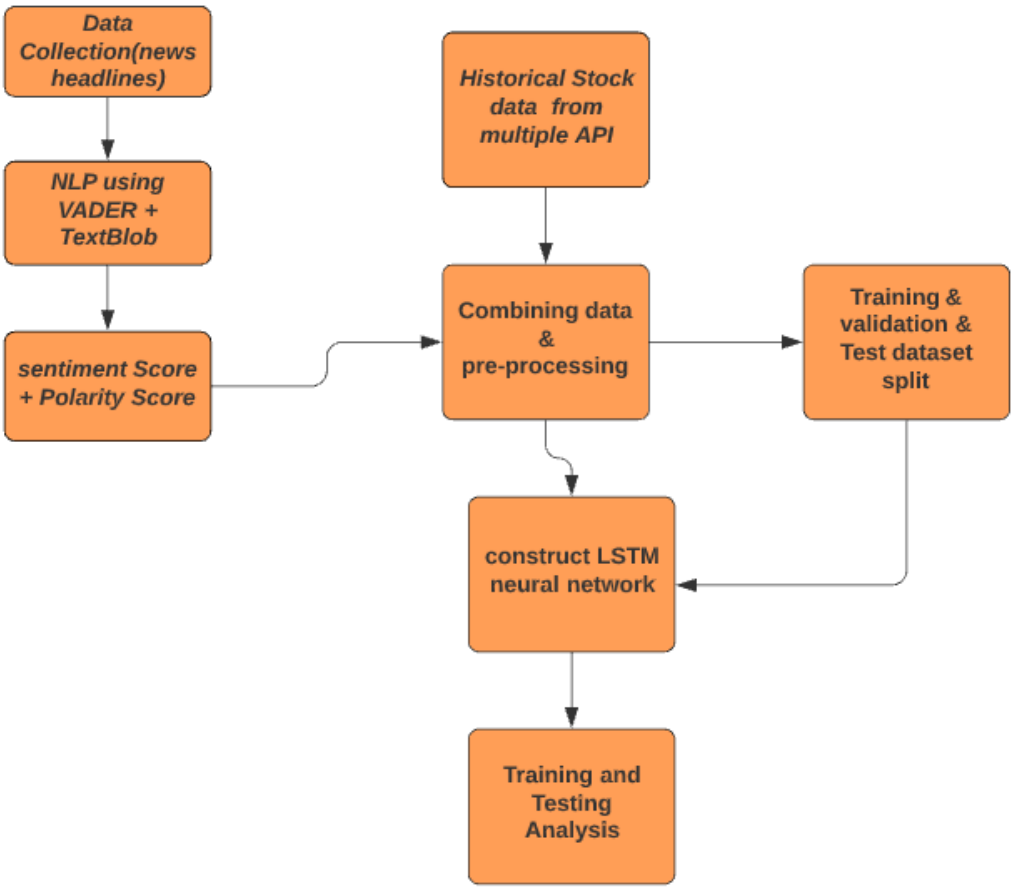


Fig [15] LSTM Model with Sentiment analysis

4.4 Phase 4: Predicting 30 days Stock Price

Once the RNN(LSTM) model is successfully created. It will be first trained using X_train data and the output of the training will then be stored in X_test which will further hold the predicted values for a particular stock. Initially the timestamp for 100 days is taken into consideration. After fitting the data with training data set (65%), the remaining Y_train and Y_test will be used to evaluate the model accuracy. The 35% test data will be used to validate the LSTM stacked model and predict the closing price of a particular stock.

4.5 Phase 5: Data Visualization

The dataset prepared during phase 1, phase 2, and phase 3 will be visualized using the python matplotlib library. The LSTM model evaluation (RMSE) during phase 2 and phase 3 for training and test data will also be visualized using matplotlib. The visualization will help analyze the difference between phase 2 and phase 3 LSTM models. Finally, a dynamic dashboard will be presented using Python Streamlit to further visualize the stock(ticker), historical price, and predicted price of 30 days. Based on the price momentum shown in the dynamic dashboard an informed decision can be made to buy or sell a particular stock.

5 RESULTS:

5.1 Sentiment Analysis (Phase 1)

In this section, previous seven days stock sentiments are shown. This phase shows how fundamental analysis can impact stock prediction. It also shows how the sentiment for a particular stock in a market can impact the performance of the stock. The stock ticker selected for this phase are Apple (AAPL), Tesla (TSLA) and Meta (FB)/Facebook. The data is scraped using python Beautifulsoup and specific new articles related to specific ticker are extracted. Fig [16] below shows the Ticker, Date-Time and News Articles.

0	FB	Nov-21-21	05:28PM	Australian mining billionaire to help publishe...
1	FB	Nov-21-21	04:17PM	These Are The 5 Best Stocks To Buy And Watch Now
2	FB	Nov-21-21	11:15AM	Top 10 Stock Picks of Leon Lowensteins Lionsto...
3	FB	Nov-21-21	07:00AM	3 Metaverse Stocks to Buy Right Now
4	FB	Nov-20-21	09:09PM	SHAREHOLDER ALERT: Levi & Korsinsky, LLP Notif...
..	...	...	...	...
295	TSLA	Nov-17-21	09:13AM	Tesla is 'the king' of EV transition: Ross Gerber
296	TSLA	Nov-17-21	08:49AM	The Big Winner From Elon Musks Stock Options P...
297	TSLA	Nov-17-21	08:15AM	Dow Jones Futures Fall As Tesla Rises But Rivi...
298	TSLA	Nov-17-21	06:11AM	Retooling auto plants for EVs will cost billio...
299	TSLA	Nov-17-21	06:00AM	Retooling auto plants for EVs will cost billio...

Fig [16] News Articles for FB, Tesla

The articles are used to calculate the polarity score on a scale of [-1, +1]. Where +1 shows that the news sentiments are positive and -1 show the news sentiment are negative. The polarity score for a particular stock is then combined into a single compounded score using Python NLTK Vader Library. The result for the compounded scores are shown below in Fig [17]

	ticker	date	time	title	compound
0	FB	2021-11-21	05:28PM	Australian mining billionaire to help publishe...	0.2960
1	FB	2021-11-21	04:17PM	These Are The 5 Best Stocks To Buy And Watch Now	0.6369
2	FB	2021-11-21	11:15AM	Top 10 Stock Picks of Leon Lowensteins Lionsto...	0.2023
3	FB	2021-11-21	07:00AM	3 Metaverse Stocks to Buy Right Now	0.0000
4	FB	2021-11-20	09:09PM	SHAREHOLDER ALERT: Levi & Korsinsky, LLP Notif...	0.2577
..	...	...	...	...	...
295	TSLA	2021-11-17	09:13AM	Tesla is 'the king' of EV transition: Ross Gerber	0.0000
296	TSLA	2021-11-17	08:49AM	The Big Winner From Elon Musks Stock Options P...	0.5267
297	TSLA	2021-11-17	08:15AM	Dow Jones Futures Fall As Tesla Rises But Rivi...	-0.4215
298	TSLA	2021-11-17	06:11AM	Retooling auto plants for EVs will cost billio...	0.4019
299	TSLA	2021-11-17	06:00AM	Retooling auto plants for EVs will cost billio...	0.4019

Fig [17] Compounded Scores

The compounded scores are grouped together using stock ticker, date and finally plotted using Matplotlib to finally visualize the new sentiments. The bar graph shows the stock price moment based on the news articles polarity score calculated.

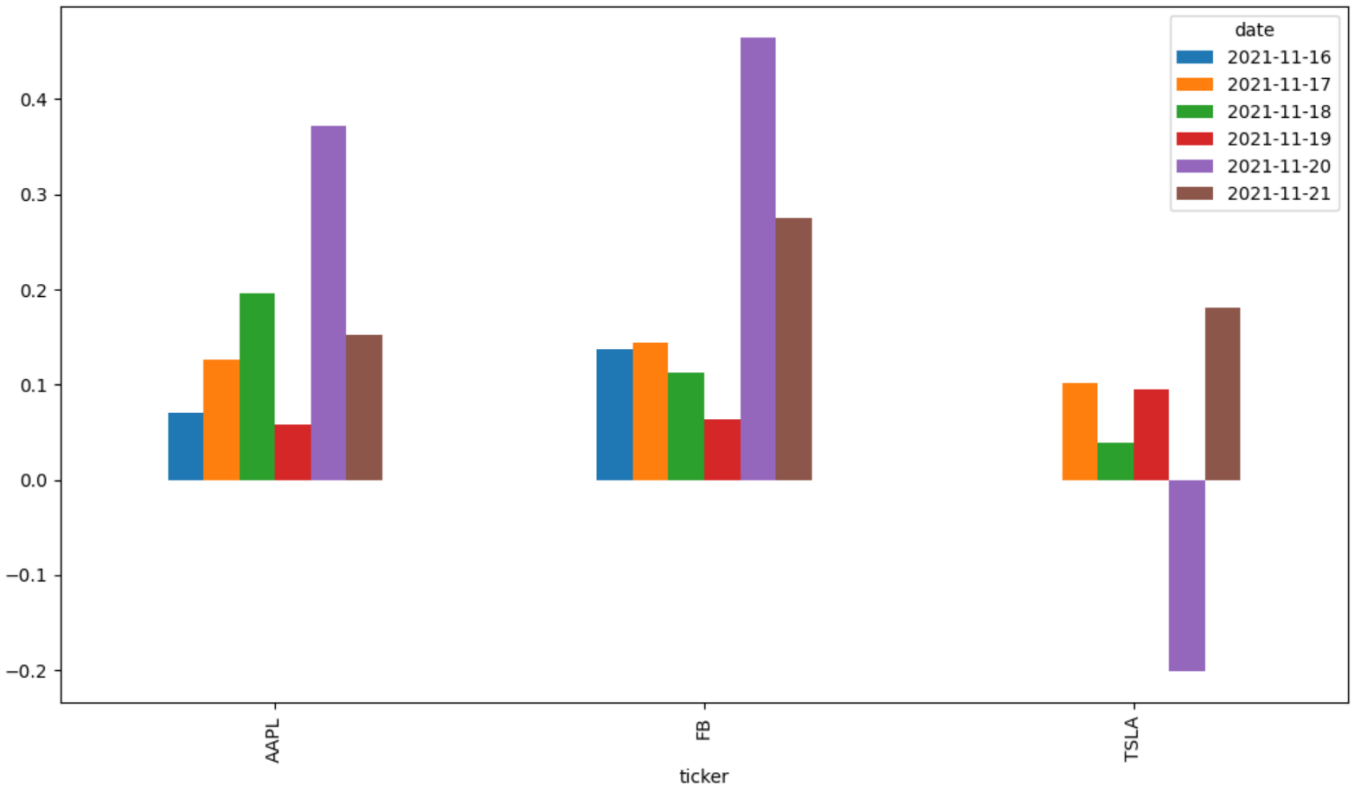


Fig [18] Sentiment Scores of Apple, FB and Tesla of Seven Days.

5.2 Long-Short Term Memory (RNN) Predictive Analysis (Phase2, Phase 3)

In this section multiple steps are performed to analyze stock market data and finally predict the stock closing price for future 30 days. Multiple graphs are provided with technical indicators which further indicate the importance of analyzing stock prices. An improved Recurrent Neural Network (RNN) architecture i.e., Long-Short Term Memory is used to overcome the drawbacks of traditional neural network. Generally, RNN encounter a major problem of vanishing gradient due to which the long-term data learning process is impacted. When it comes to time-series data it is important to keep track of past data.

Technical Indicators for Stock Price Prediction (Moving Averages):

Technical indicators are widely used for predicting the momentum of any stock price along with fundamental and technical analysis. Technical indicator plays an important role to understand the trends of a stock based on previous values. Moving average is a widely used technical indicator used for capturing stock momentum. In this example, the graph of the company FB(Meta) and Tesla is shown. The graph shows the year-on-year change in the closing prices.

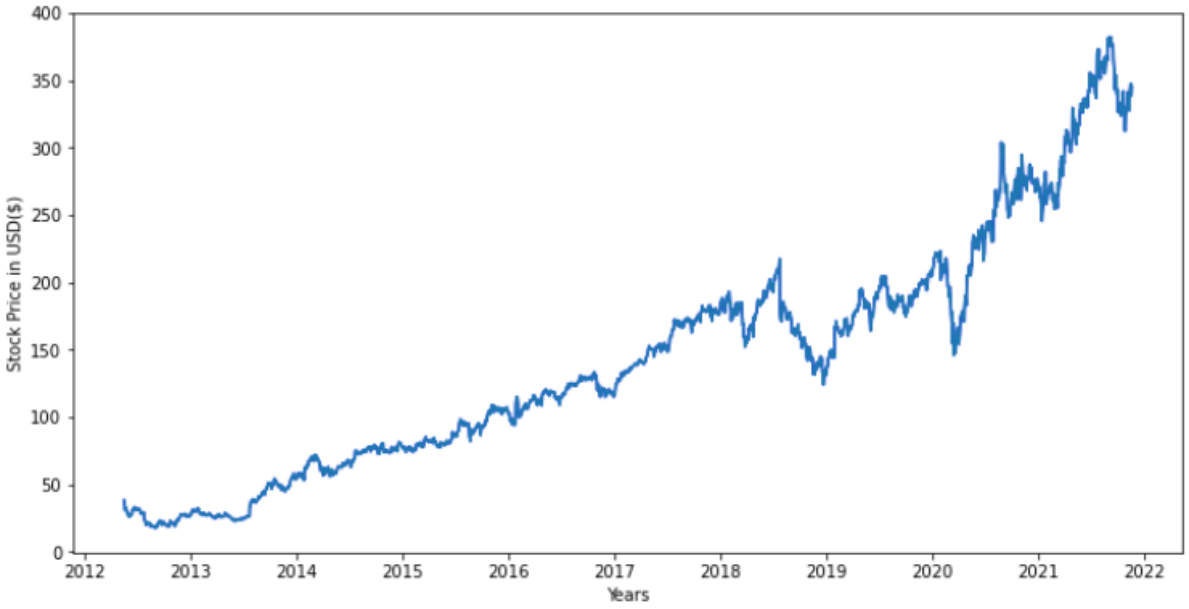


Fig [19] Year-on-Year Stock Price of Facebook

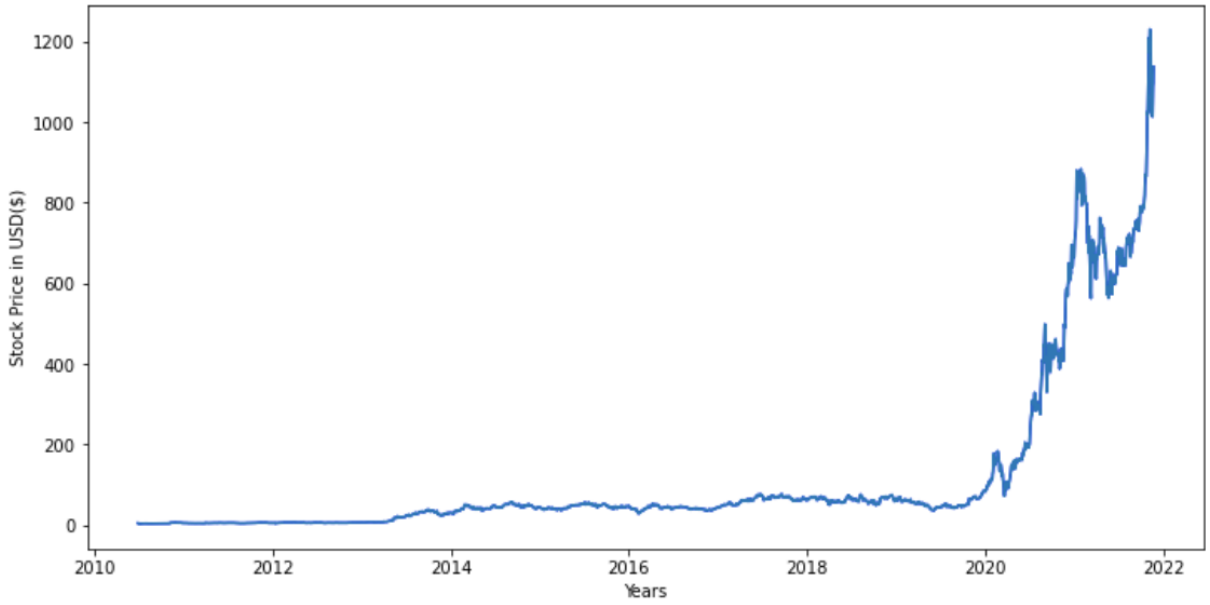


Fig [20] Year-on-Year Stock Price of Tesla

Moving Average of 100 & 200 days:

100- and 200-days moving average takes an average of previous 100- and 200-days closing price and the value calculated can be plotted on the original graph. Technical Analyst in the market uses a strategy that if 100 days moving average crosses above the 200 days moving average then there is an uptrend or if 100 days moving average crosses below the 200 days moving average then it is the starting point of down trend.

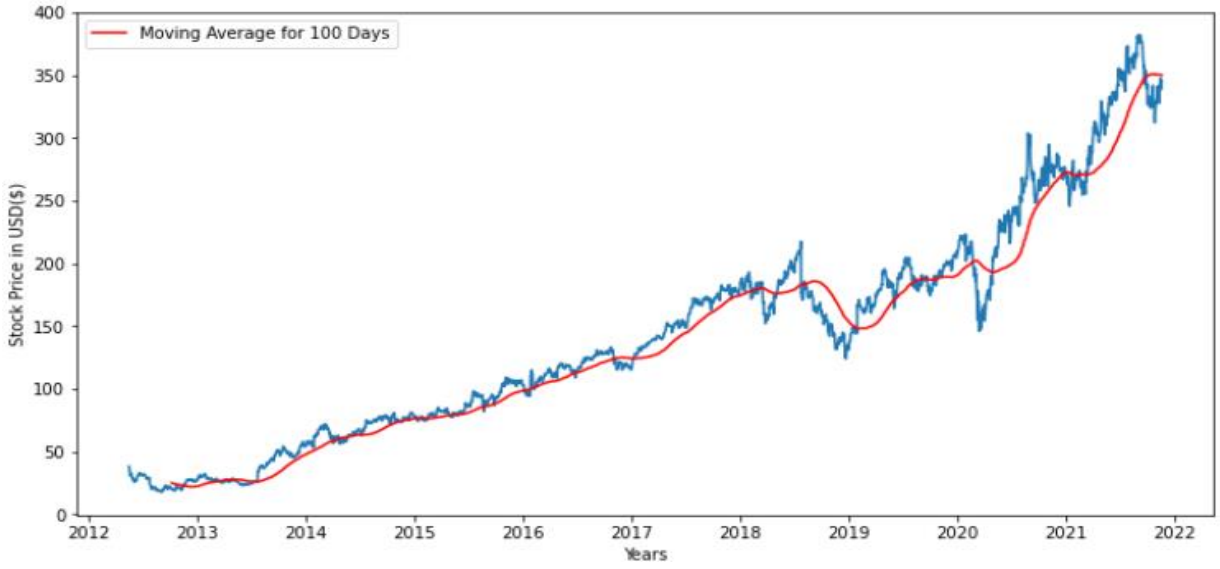


Fig [21] FB Moving Average of 100 days

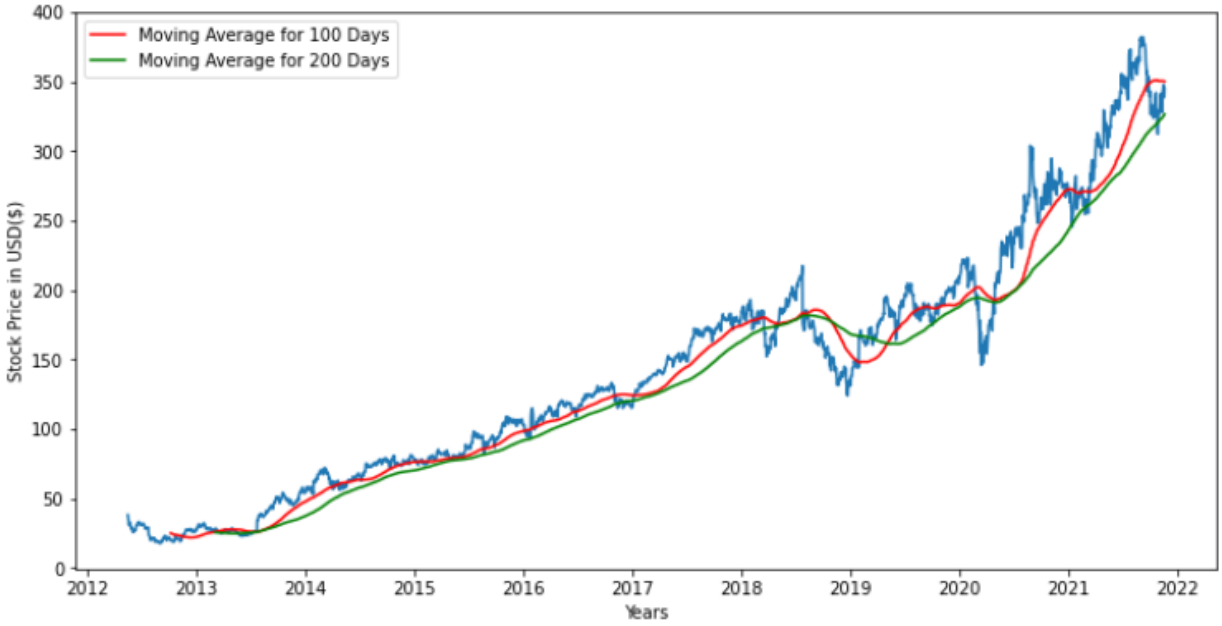


Fig [22] 100- & 200-Days Moving Averages FB.

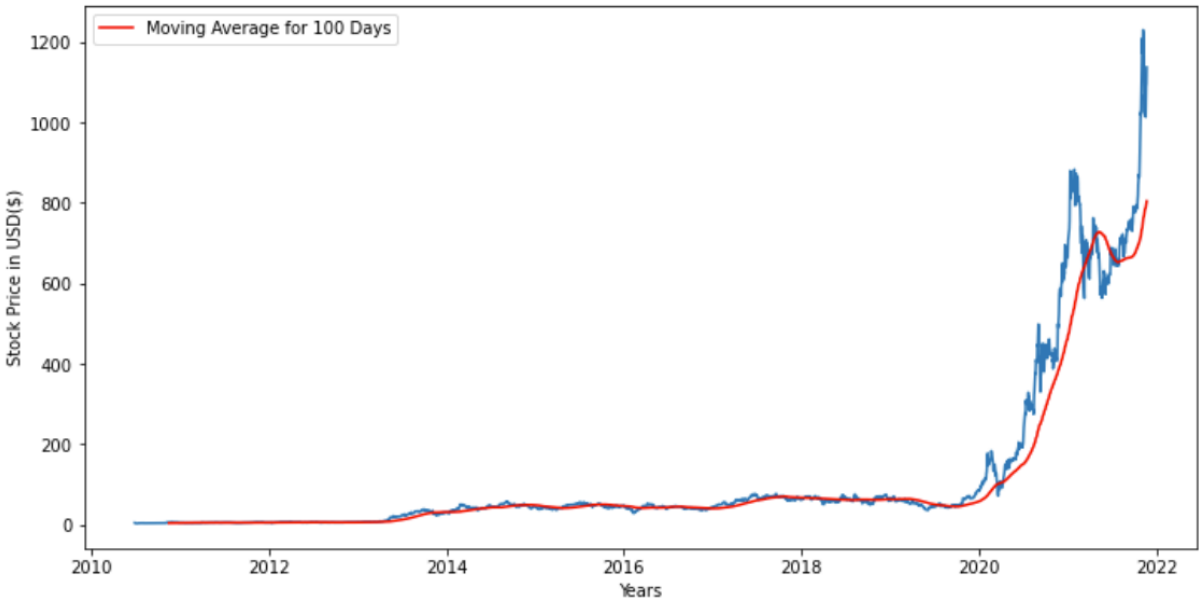


Fig [23] Tesla Moving Average of 100 days

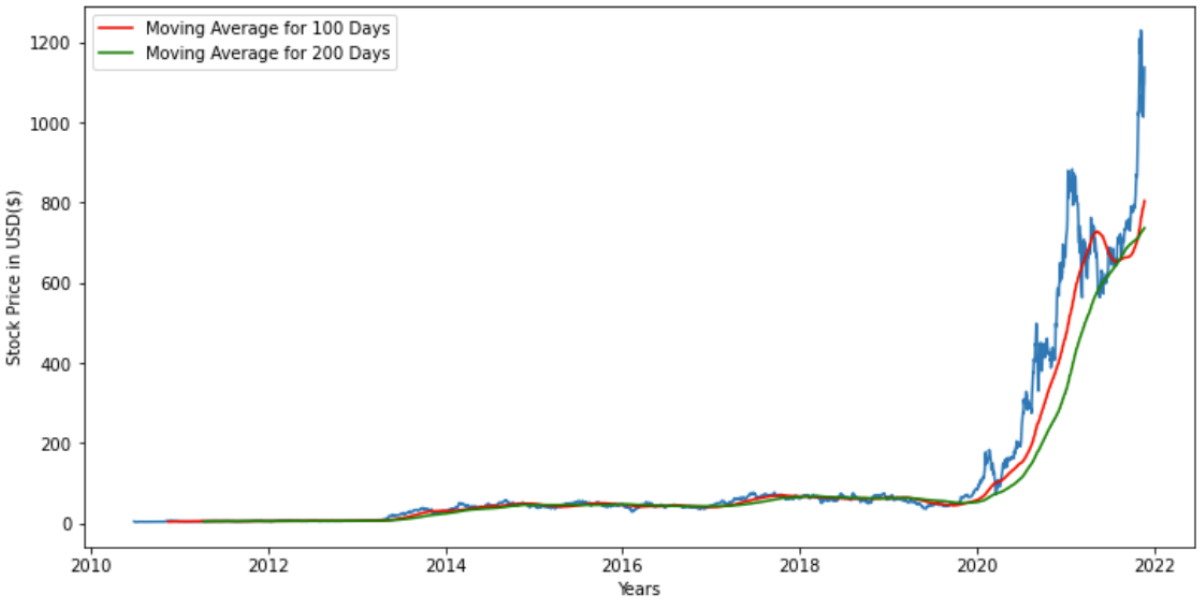


Fig [24] Tesla Moving Average of 100 days

Step 1: Data Collection:

The data collection is done from Yahoo finance API, Here the use of Pandas data reader function is done to collect stock prices for FB and TESLA stock from 2010. There are multiples columns in the data extracted, However, in this case only closing price is taken into consideration. In future work more prices can also be taken into consideration and a multivariate LSTM model can be used to process those values. Note: Facebook was listed in 2012 hence the earliest data available is from 2012.

```
### Getting data from Yahoo finance
start = '2010-01-01'
end = '2021-11-20'
df= pdr.DataReader('FB','yahoo',start,end)
```

```
### Getting data from Yahoo finance
start = '2010-01-01'
end = '2021-11-20'
df= pdr.DataReader('TSLA','yahoo',start,end)
```

Fig [25] Yahoo Finance as data source

```
df.head()
```

Date	High	Low	Open	Close	Volume	Adj Close
2012-05-18	45.000000	38.000000	42.049999	38.230000	573576400	38.230000
2012-05-21	36.660000	33.000000	36.529999	34.029999	168192700	34.029999
2012-05-22	33.590000	30.940001	32.610001	31.000000	101786600	31.000000
2012-05-23	32.500000	31.360001	31.370001	32.000000	73600000	32.000000
2012-05-24	33.209999	31.770000	32.950001	33.029999	50237200	33.029999

```
df.tail()
```

Date	High	Low	Open	Close	Volume	Adj Close
2021-11-15	353.649994	343.200012	344.339996	347.559998	25076600	347.559998
2021-11-16	346.649994	340.869995	343.829987	342.959991	18181100	342.959991
2021-11-17	347.299988	340.100006	344.239990	340.769989	13602800	340.769989
2021-11-18	342.459991	335.299988	339.720001	338.690002	17487200	338.690002
2021-11-19	352.100006	339.899994	342.200012	345.299988	26441000	345.299988

Fig [26] Open, Close Prices for FB.

Date	High	Low	Open	Close	Volume	Adj Close
2010-06-29	5.000	3.508	3.800	4.778	93831500.0	4.778
2010-06-30	6.084	4.660	5.158	4.766	85935500.0	4.766
2010-07-01	5.184	4.054	5.000	4.392	41094000.0	4.392
2010-07-02	4.620	3.742	4.600	3.840	25699000.0	3.840
2010-07-06	4.000	3.166	4.000	3.222	34334500.0	3.222

```
df.tail()
```

Date	High	Low	Open	Close	Volume	Adj Close
2021-11-15	1031.979980	978.599976	1017.630005	1013.390015	34775600.0	1013.390015
2021-11-16	1057.199951	1002.179993	1003.309998	1054.729980	26542400.0	1054.729980
2021-11-17	1119.640015	1055.500000	1063.510010	1089.010010	31445400.0	1089.010010
2021-11-18	1112.000000	1075.020020	1106.550049	1096.380005	20898900.0	1096.380005
2021-11-19	1138.719971	1092.699951	1098.869995	1137.060059	21642300.0	1137.060059

Fig [27] Open, Close Prices for TESLA.

Step 2: Data Pre-Processing

1. **Scaling of Data**
   After successfully collecting the data, the next step is to pre-process the existing prices as LSTM Model is sensitive to the input provided. There is a need for scaling the data. Here use of sklearn MinMax Scaler is used to normalize the data from [0,1].

```
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
print(df1)
```

```
[[0.05624914]
 [0.04472493]
 [0.03641103]
 ...
 [0.88637672]
 [0.88066953]
 [0.89880641]]
```

Fig[28] Scaling closing prices for LSTM Input

2. **Splitting the data into train and test data**

```
##splitting dataset into training and testing
training_size=int(len(df1)*0.65)
test_size=len(df1)-training_size
train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]
```

```
training_size,test_size
```

```
(1556, 838)
```

```
##splitting dataset into training and testing
training_size=int(len(df1)*0.80)
test_size=len(df1)-training_size
train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]
```

Fig [29] Training and Test data set for FB and Tesla

In this case 65% of the data is taken as training data and 35% [1][5][10][12] of data is taken as test data. Also, in this model as a hyperparameter tuning the timestamp used is of 100 days, meaning 100 past days data will be used to calculate 101 day data, creating independent and dependent feature using the 100 days timestamp. The process is repeated for both training and test data. Since LSTM model require data in 3 dimension the existing training and test data need to be reshaped inorder to be used as an input for LSTM.

```
# reshape
time_step = 100
x_train, y_train = create_dataset(train_data, time_step)
x_test, y_test = create_dataset(test_data, time_step)
```

```
# reshape input to be [samples, time steps, features] which is required for LSTM
x_train =x_train.reshape(x_train.shape[0],x_train.shape[1] , 1)
x_test = x_test.reshape(x_test.shape[0],x_test.shape[1] , 1)
```

Fig [30] Reshaping train and test data

Step 3: Stacked LSTM Model Creation

In this step a sequential LSTM model is created using TensorFlow keras. The first and second layer uses 50 units and an activation function caller 'relu' used for improving the efficiency of stacked LSTM. Finally, a dense layer of 1 unit is used to as an output 'adam' optimizer. Also, to check the efficiency of the model mean squared error is used as a performance metric.

```
### Create Stacked LSTM model and importing models
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

```
##adding more nodes and layers can increase the efficiency
model=Sequential()
model.add(LSTM(units = 50,activation='relu', return_sequences=True,input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50,activation='relu',return_sequences=True))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_17 (LSTM)              (None, 100, 50)           10400

 lstm_18 (LSTM)              (None, 100, 50)           20200

 lstm_19 (LSTM)              (None, 50)                20200

 dense_5 (Dense)             (None, 1)                 51

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

Fig [31] LSTM Model Summary

Once the model is created it needs to be fitted using x_train and y_train(training data set). The use of validation data i.e x_test and y_test is also used. Epochs of 100 is used and a batch size of 64 is taken as a starting values. From the below fig [32] it can be seen that with every epoch the validation losses are reduced.

```
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100,batch_size=64,verbose=1)
validation_data=(x_test,y_test)
```

```
Epoch 1/100
23/23 [==============================] - 8s 208ms/step - loss: 0.0207 - val_loss: 0.0434
Epoch 2/100
23/23 [==============================] - 4s 183ms/step - loss: 0.0011 - val_loss: 0.0030
Epoch 3/100
23/23 [==============================] - 6s 246ms/step - loss: 2.8925e-04 - val_loss: 0.0019
Epoch 4/100
23/23 [==============================] - 6s 256ms/step - loss: 1.9370e-04 - val_loss: 0.0024
Epoch 5/100
23/23 [==============================] - 6s 246ms/step - loss: 1.6826e-04 - val_loss: 0.0022
Epoch 6/100
23/23 [==============================] - 5s 235ms/step - loss: 1.6346e-04 - val_loss: 0.0026
Epoch 7/100
23/23 [==============================] - 6s 251ms/step - loss: 1.6537e-04 - val_loss: 0.0023
Epoch 8/100
23/23 [==============================] - 7s 292ms/step - loss: 1.7381e-04 - val_loss: 0.0025
```

```
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100,batch_size=32,verbose=1)
validation_data=(x_test,y_test)
```

```
Epoch 1/100
69/69 [==============================] - 15s 177ms/step - loss: 7.2063e-05 - val_loss: 0.0073
Epoch 2/100
69/69 [==============================] - 10s 139ms/step - loss: 9.1700e-06 - val_loss: 0.0070
Epoch 3/100
69/69 [==============================] - 10s 144ms/step - loss: 9.2219e-06 - val_loss: 0.0099
Epoch 4/100
69/69 [==============================] - 11s 154ms/step - loss: 7.7426e-06 - val_loss: 0.0075
Epoch 5/100
69/69 [==============================] - 11s 157ms/step - loss: 9.3625e-06 - val_loss: 0.0084
Epoch 6/100
69/69 [==============================] - 10s 139ms/step - loss: 7.1815e-06 - val_loss: 0.0093
Epoch 7/100
69/69 [==============================] - 10s 145ms/step - loss: 6.7531e-06 - val_loss: 0.0084
Epoch 8/100
69/69 [==============================] - 10s 146ms/step - loss: 6.3586e-06 - val_loss: 0.0084
Epoch 9/100
69/69 [==============================] - 11s 164ms/step - loss: 5.9084e-06 - val_loss: 0.0077
Epoch 10/100
```

Fig [32] Model fitting

To obtain best results, multiple epochs and batch sizes can be taken into consideration.

Step 4: Predict and Test Data

After training the model prediction needs to be performed using first x_train data and then using x_test data. The next step will be to reverse the normalization performed earlier. In this case use of scaler.inverse_transform() is done to get the original prices of the stock back. The performance metric used to evaluate the model is Root Mean Squared Error(RMSE) using sklearn. The parameters used for RMSE are y_train and y_predict(test_predict).

```
### Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,x_predict))
```

```
0.009866275018928219
```

```
### Test Data RMSE
math.sqrt(mean_squared_error(y_test,y_predict))
```

```
0.0456594593844303
```

Fig [33] RMSE train and test

Finally, the visualization is done using matplotlib library showing the original price, trained price, and test price. As we can see from the below graphs that the training and testing completely overlap it indicated the model build is fairly accurate.

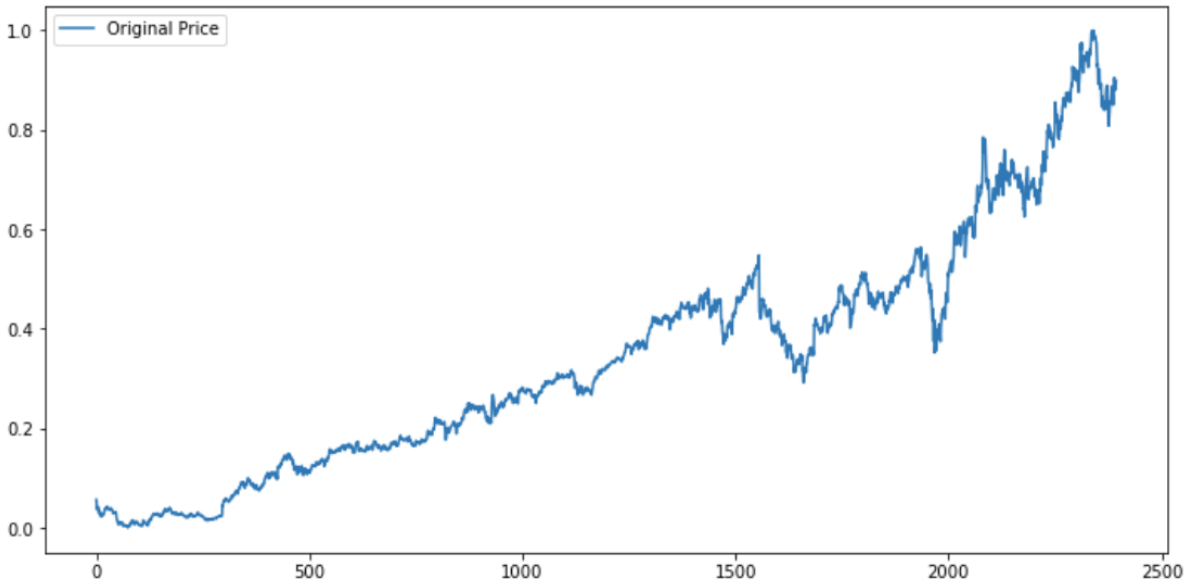


Fig [34] Original price graph for FB

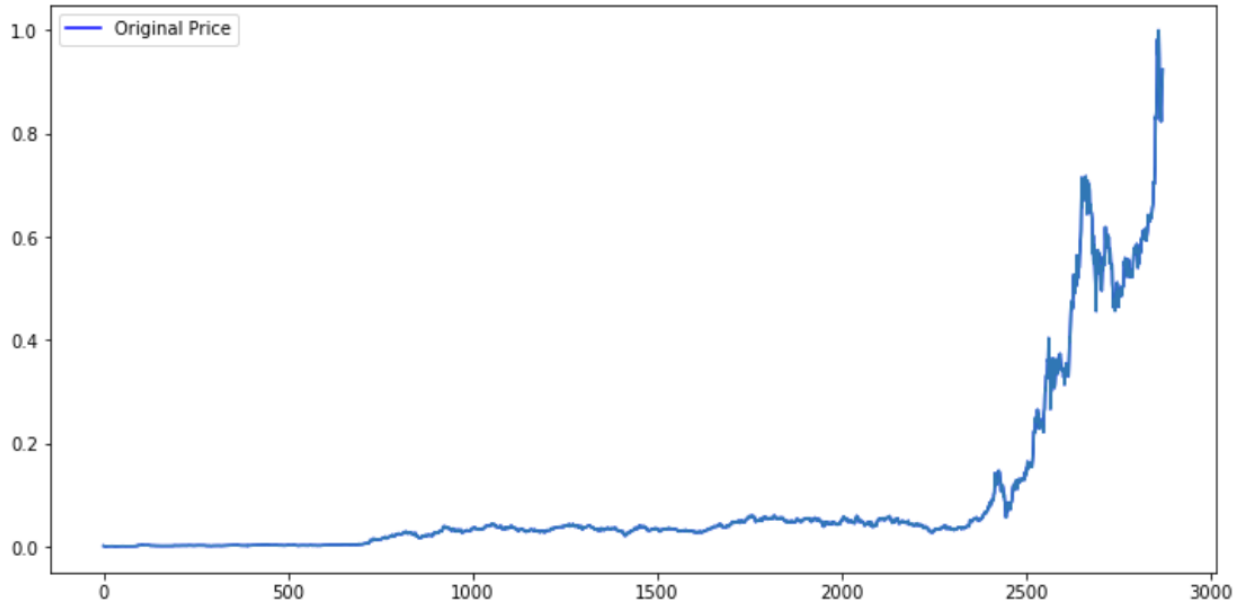


Fig [35] Original price graph for TESLA

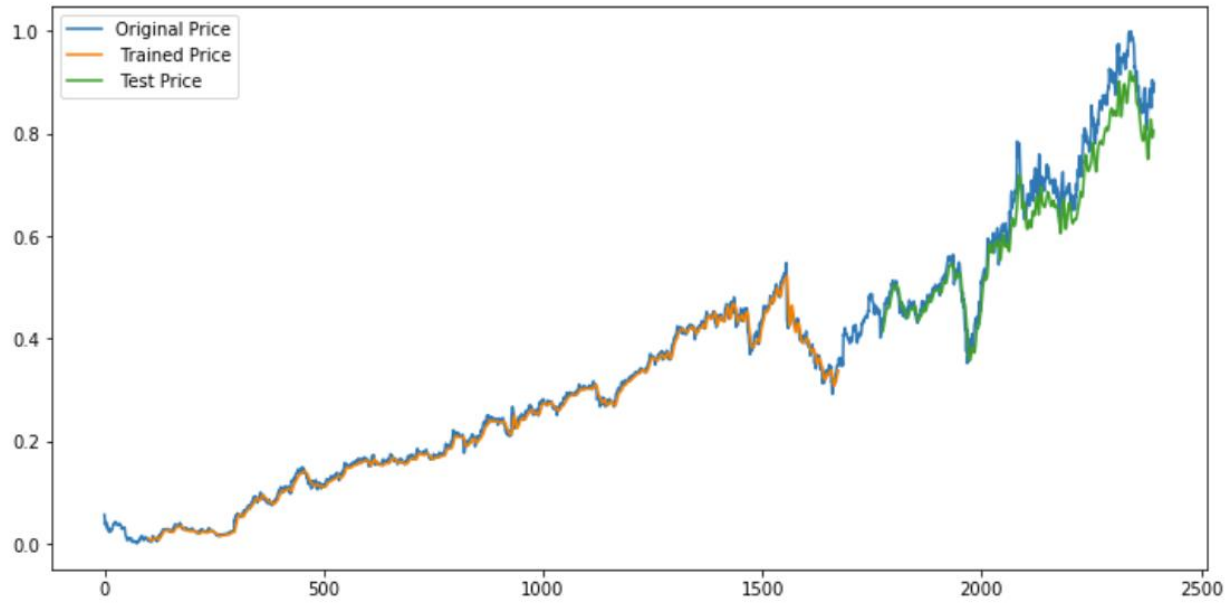


Fig [36] Training and Testing Prices for FB

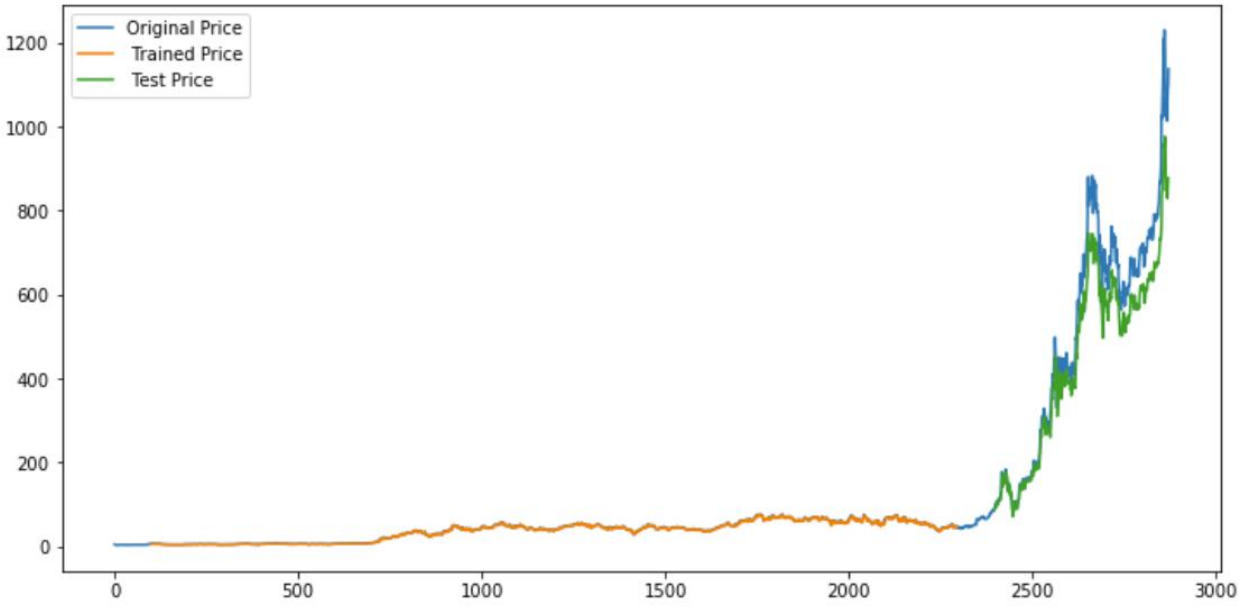


Fig [37] Training and Testing Prices for TESLA

Step 5: Predicting Future 30 Days Stock Prices and Momentum

Originally the test data build has 719 prices. To predict next 30 days data, we need to take previous 100 days data as it is the timestamp under consideration and use the existing LSTM model. Similarly, to feed data into LSTM the data need to be first reshaped so that it's 3 dimensional.

```
len(test_data)
```

```
719
```

```
x_input=test_data[len(test_data)-100:].reshape(1,-1)
x_input.shape
```

```
(1, 100)
```

```
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
len(temp_input)
```

```
100
```

Fig [38] Reshaping previous 100 days test data for LSTM model

To get the input for next 30 days, the 100 days data generated is passed to the model build, and prediction is made. A sliding window approach is used here where the values obtained from prediction are added to the final output and again added to the previous input. At this point there will be 101 values in the input and sliding is done with each iteration to obtain next 30 days data.

```
# demonstrate prediction for next 30 days
from numpy import array

lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(temp_input)>n_steps):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        #print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)
```

```
1 day output [[0.8130377]]
2 day output [[0.7982684]]
3 day output [[0.77922726]]
4 day output [[0.7595558]]
5 day output [[0.74169415]]
6 day output [[0.72696674]]
7 day output [[0.7157248]]
8 day output [[0.7076498]]
9 day output [[0.701958]]
10 day output [[0.69761616]]
11 day output [[0.6935402]]
12 day output [[0.68875015]]
13 day output [[0.68247384]]
14 day output [[0.67420906]]
15 day output [[0.6637546]]
16 day output [[0.6512178]]
17 day output [[0.63698757]]
18 day output [[0.62147087]]
19 day output [[0.6052769]]
20 day output [[0.5891919]]
21 day output [[0.5739871]]
22 day output [[0.5602738]]
23 day output [[0.5484161]]
24 day output [[0.5385001]]
25 day output [[0.53036565]]
26 day output [[0.5237037]]
27 day output [[0.5180598]]
28 day output [[0.5129288]]
29 day output [[0.5078312]]
[[0.8189530372619629], [0.8130376935005188], [0.7982683777809143], [0.7792272567749023], [0.7595558166503906], [0.7416941523
551941], [0.7269667387008667], [0.7157248258590698], [0.707649827003479], [0.7019580006599426], [0.6976161599159241], [0.693
5402154922485], [0.688750147819519], [0.6824738383293152], [0.6742090582847595], [0.6637545824050903], [0.6512178182601929],
[0.636987566947937], [0.6214708685874939], [0.605276882648468], [0.589191913604736 3], [0.5739871263504028], [0.5602738261222
839], [0.5484160780906677], [0.5385000705718994], [0.5303656458854675], [0.5237036943435669], [0.5180597901344299], [0.51292
87838935852], [0.5078312158584595]]
```

Fig [39] Next 30 days data

Finally, the graph for next 30 days is plotted using python Matplotlib library which can be seen in the fig below.

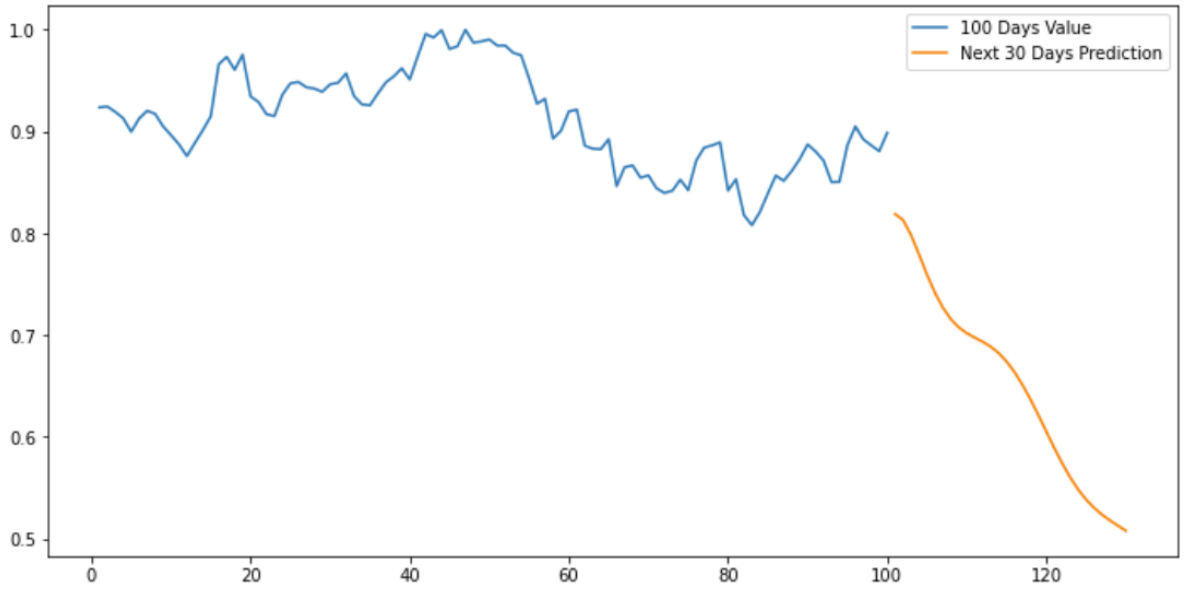


Fig [40] Next 30 days predicted momentum for FB

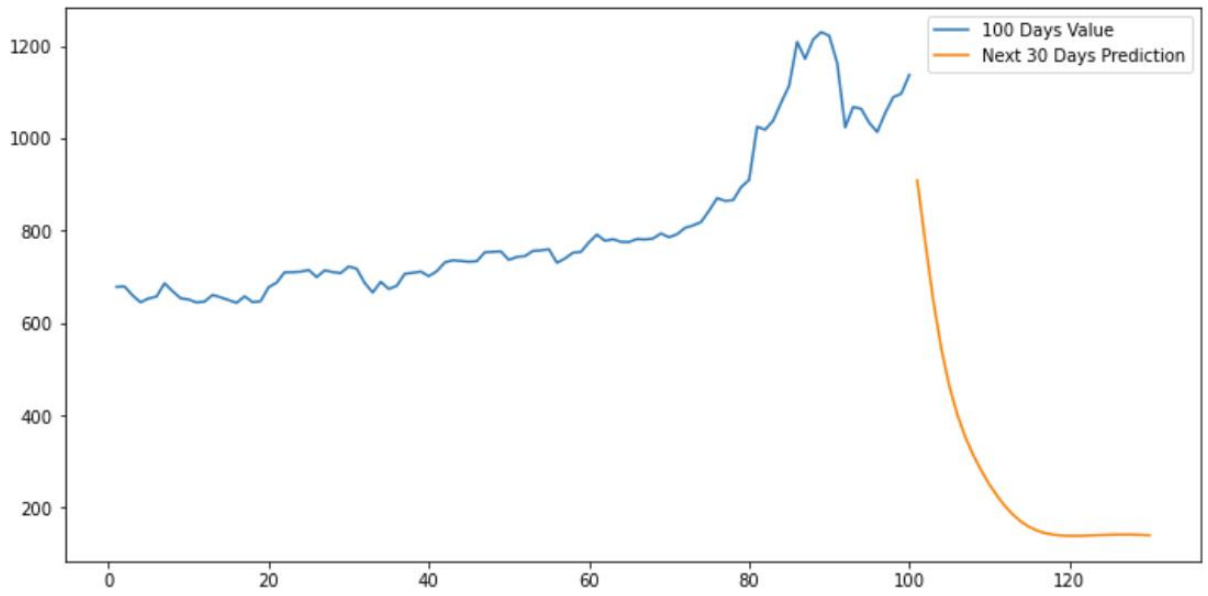


Fig [41] Next 30 days predicted momentum for TESLA

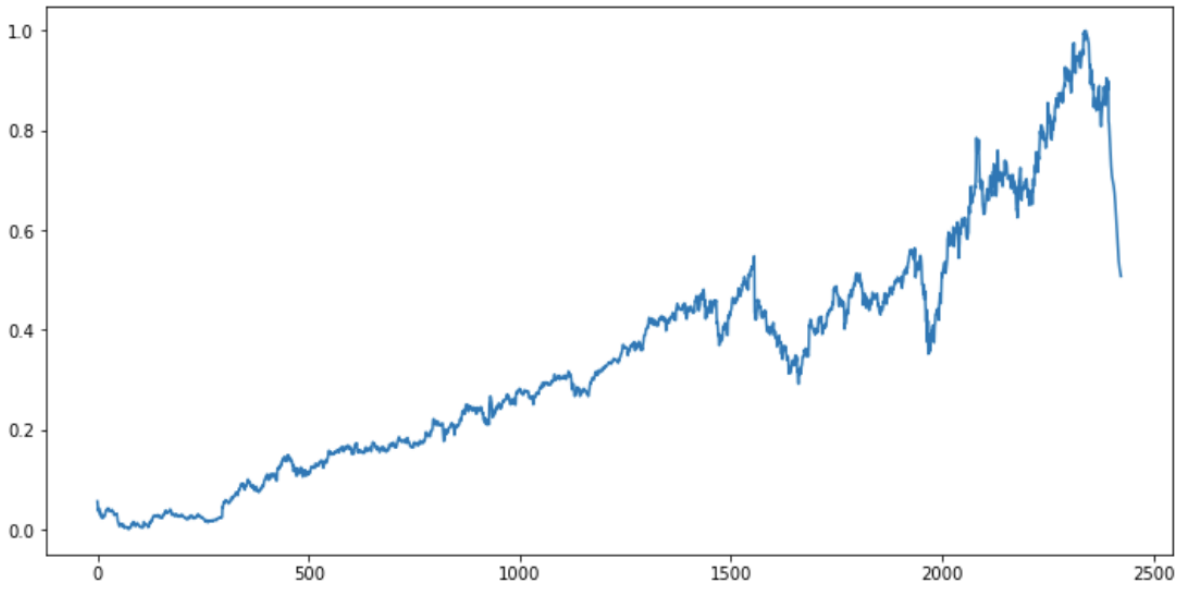


Fig [42] Final graph with smoothing

6 VALIDATING RESULTS WITH PREVIOUS STOCK PRICES:

Data Collection:

The data collection is done from Yahoo finance API, Here the use of Pandas data reader function is done to collect stock prices for TESLA stock from 2010. There are multiples columns in the data extracted, However, in this case only closing price is taken into consideration. In future work more prices can also be taken into consideration and a multivariate LSTM model can be used to process those values. Note: Tesla was listed in 2010 hence the earliest data available is from 2010.

```
### Getting data from Yahoo finance
start = '2004-01-01'
end = '2019-11-20'
df= pdr.DataReader('TSLA','yahoo',start,end)
```

Fig [43] Data source from yahoo finance

Date	High	Low	Open	Close	Volume	Adj Close
2010-06-29	5.000	3.508	3.800	4.778	93831500	4.778
2010-06-30	6.084	4.660	5.158	4.766	85935500	4.766
2010-07-01	5.184	4.054	5.000	4.392	41094000	4.392
2010-07-02	4.620	3.742	4.600	3.840	25699000	3.840
2010-07-06	4.000	3.166	4.000	3.222	34334500	3.222

```
df.tail()
```

Date	High	Low	Open	Close	Volume	Adj Close
2019-11-14	70.767998	68.582001	69.222000	69.870003	32324500	69.870003
2019-11-15	70.559998	69.671997	70.127998	70.433998	24045000	70.433998
2019-11-18	70.629997	69.220001	70.584000	69.998001	22002000	69.998001
2019-11-19	71.998001	69.559998	70.349998	71.903999	38624000	71.903999
2019-11-20	72.239998	69.914001	72.000000	70.444000	33625500	70.444000

Fig [44] Open, Close Prices for TESLA

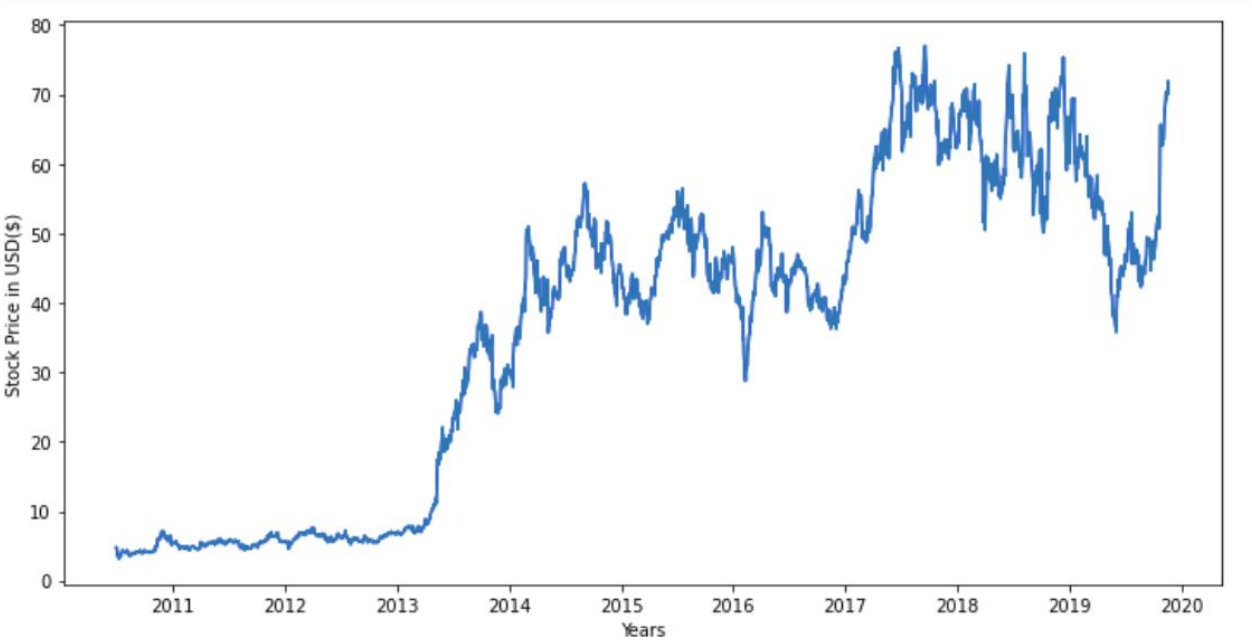


Fig [45] Year-on-Year Stock Price of TESLA

Predicting and Training

After training the model prediction needs to be performed using first x_train data and then using x_test data. The next step will be to reverse the normalization performed earlier. In this case use of scaler.inverse_transform() is done to get the original prices of the stock back. The performance metric used to evaluate the model is Root Mean Squared Error(RMSE) using sklearn. The parameters used for RMSE are y_train and y_predict(test_predict).

```
### Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,x_predict))
```

```
38.24919453709721
```

```
### Test Data RMSE
math.sqrt(mean_squared_error(y_test,y_predict))
```

```
57.21479056434584
```

Fig [46] RMSE train and test

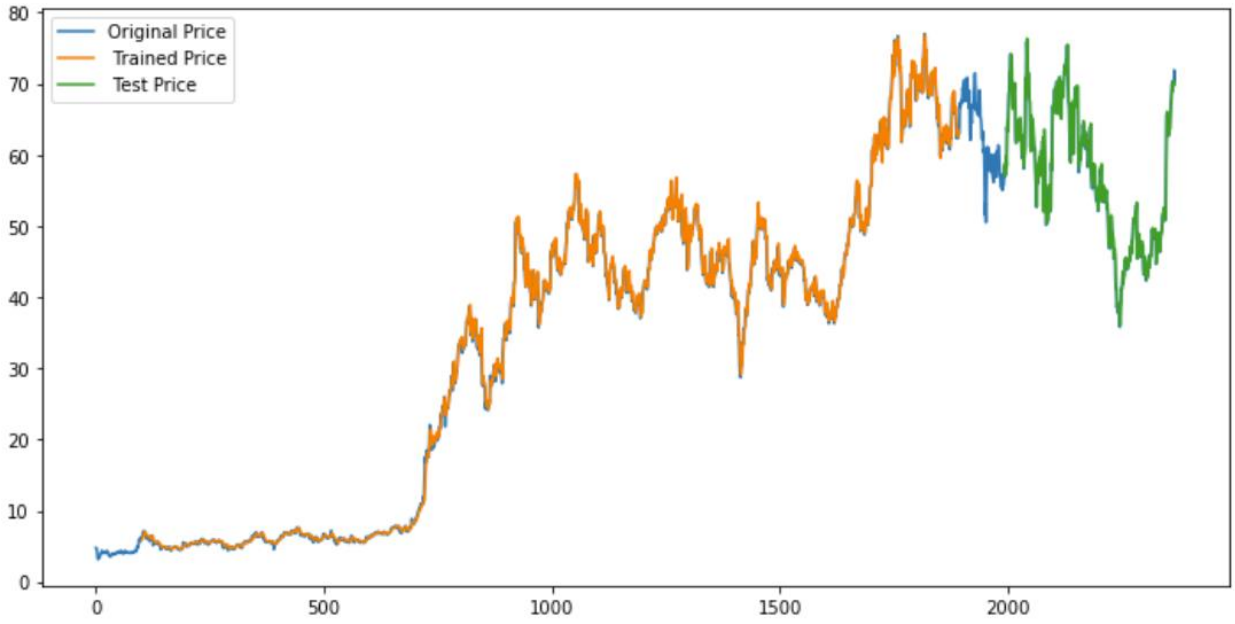


Fig [47] Training and Testing Prices for TESLA

Predicting Future 30 days Stock Prices and Momentum

Here 30 days data is taken from the past to validate the accuracy of model prediction. The original model uses data from January 2012- November 2019. In this case the model aims to predict the momentum for the month of December 2019. As we can see from Fig [49] right side that the prices originally go up from the November 2019. This is exactly what the model predicts that the momentum of prices in the month of December is rising.

```
start1 = '2019-11-20'
end2 = '2019-12-20'
df5= pdr.DataReader('TSLA','yahoo',start1,end2)
```

```
df5.head()
```

Fig [48] Data source from yahoo finance

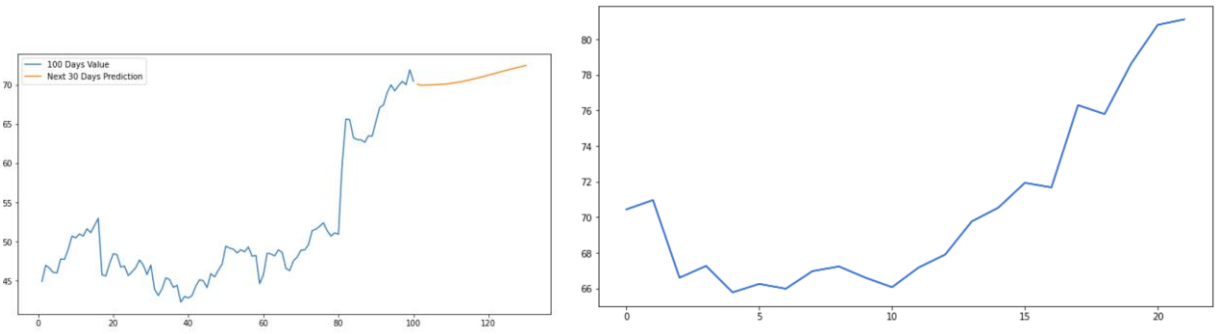


Fig [49] Verifying TESLA stock price momentum

7 FUTURE WORK:

Predicting stock market is a challenging task given the number of parameters involved. The project uses both technical and fundamental analysis to build a RNN model and predict the 30 days future prices and momentum. The LSTM model build is a single variate stacked LSTM in future more work can be done to use multivariate LSTM and taking into consideration Open and adjusted prices. Also, the model uses unit of 50 nodes in, and activation as 'relu'. In future better activation and optimizer can be used to improve the accuracy of the model. Also, for analyzing the sentiment of news articles better tokenization techniques such as bigram and n-gram can be used to improve the polarity score of an article.

8 CONCLUSION:

In this project, predictive analysis is performed on stock market data using both sentiment and time series analysis. The project uses multiple news sources to perform sentiment analysis using the VADER library in addition to other lexicon approaches. The benefit of using such a hybrid approach is that it can work easily on live stream data and is suitable for the versatile analysis of data. The final prediction will show a strong correlation between the news articles, social media posts, and stock prices. The LSTM model build will further verify the significance of public sentiment for stock market prediction. The proposed method can predict almost 80 % of market variance using simple sentiment analysis and LSTM. In the future, the proposed method and model build can be further extended to the forecast of cryptocurrency since research shows that there is a huge change in the momentum of cryptocurrency with news sentiments.

9 REFERENCES:

[1] Hegde, M. S., Krishna, G., & Srinath, R. (2018). An Ensemble Stock Predictor and Recommender System. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1981–1985. https://doi.org/10.1109/ICACCI.2018.8554424

[2] Shah, D., Isah, H., & Zulkernine, F. (2018). Predicting the Effects of News Sentiments on the Stock Market. *2018 IEEE International Conference on Big Data (Big Data)*, 4705–4708. https://doi.org/10.1109/BigData.2018.8621884

[3] Pasupulety, U., Abdullah Anees, A., Anmol, S., & Mohan, B. R. (2019). Predicting Stock Prices using Ensemble Learning and Sentiment Analysis. *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 215–222. https://doi.org/10.1109/AIKE.2019.00045

[4] Alostad, H., & Davulcu, H. (2015). Directional Prediction of Stock Prices Using Breaking News on Twitter. *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 523–530. https://doi.org/10.1109/WI-IAT.2015.82

[5] Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. *2015 IEEE International Conference on Big Data (Big Data)*, 2823–2824. https://doi.org/10.1109/BigData.2015.7364089

[6] Nikou, M., Mansourfar, G., & Bagherzadeh, J. (2019). Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, *26*(4), 164–174. https://doi.org/10.1002/isaf.1459

[7] Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. *2015 IEEE International Conference on Big Data (Big Data)*, 2823–2824. https://doi.org/10.1109/BigData.2015.7364089

[8] Sarkar, A., Sahoo, A. K., Sah, S., & Pradhan, C. (2020). LSTMSA: A Novel Approach for Stock Market Prediction Using LSTM and Sentiment Analysis. *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 1–6. https://doi.org/10.1109/ICCSEA49143.2020.9132928

[9] Gupta, R., & Chen, M. (2020). Sentiment Analysis for Stock Price Prediction. *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 213–218. https://doi.org/10.1109/MIPR49039.2020.00051

[10] Guo, Y. (2020). Stock Price Prediction Based on LSTM Neural Network: The Effectiveness of News Sentiment Analysis. *2020 2nd International Conference on Economic Management and Model Engineering (ICEMME)*, 1018–1024. https://doi.org/10.1109/ICEMME51517.2020.00206

[11] Aasi, B., Imtiaz, S. A., Qadeer, H. A., Singarajah, M., & Kashef, R. (2021). Stock Price Prediction Using a Multivariate Multistep LSTM: A Sentiment and Public Engagement Analysis

Model. *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 1–8. https://doi.org/10.1109/IEMTRONICS52119.2021.9422526

[12] Bathla, G. (2020). Stock Price prediction using LSTM and SVR. *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 211–214. https://doi.org/10.1109/PDGC50313.2020.9315800

[13] Mehta, Y., Malhar, A., & Shankarmani, R. (2021). Stock Price Prediction using Machine Learning and Sentiment Analysis. *2021 2nd International Conference for Emerging Technology (INCET)*, 1–4. https://doi.org/10.1109/INCET51464.2021.9456376

[14] Mohan, S., Mullapudi, S., Sammeta, S., Vijayvergia, P., & Anastasiu, D. C. (2019). Stock Price Prediction Using News Sentiment Analysis. *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, 205–208. https://doi.org/10.1109/BigDataService.2019.00035

[15] Urolagin, S. (2017). Text Mining of Tweet for Sentiment Classification and Association with Stock Prices. *2017 International Conference on Computer and Applications (ICCA)*, 384–388. https://doi.org/10.1109/COMAPP.2017.8079788