A Project report

on

STOCK PRICE PREDICTION

Submitted By

Batina Koushik	19bcs019
Burra Manikanta	19bcs026
Dasari Rishikesh	19bcs035
Mankena Roja	19bcs067

Under the guidance of

Dr. B. Jayalakshmi

Asst. Professor, Computer Science & Engineering

Dr. Deepak K T

Asst. Professor, Electronics & Communication Engineering

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
DHARWAD

Certificate

This is to certify that the work contained in the project report titled **STOCK PRICE PREDICTION** by **Batina Koushik, Burra Manikanta, Dasari Rishikesh, and Mankena Roja** was completed during the VI semester - III Year as a Machine Learning course project under the guidance of **Dr. B. Jayalakshmi,** Asst. Professor, CSE dept. and **Dr. Deepak K T**, Asst. Professor, ECE dept.

Signature of the supervisor

Dr. B. Jayalakshmi

Asst. Professor, Computer Science & Engineering

Dr. Deepak K T

Asst. Professor, Electronics & Communication Engineering

Abstract

Stock market investment strategies are complex and rely on an evaluation of vast amounts of data. In recent years, machine learning techniques have increasingly been examined to assess whether they can improve market forecasting when compared with traditional approaches. The objective for this study is to identify directions for future machine learning stock market prediction. The predictions were made by using five different algorithms and they are linear regression, k-nearest neighbor, auto-ARIMA, prophet, and LSTM. These models give accurate results as they go by. We came up with a comparison of output models to analyze which is the best model to predict the market prices. In this study the datasets are taken of different companies and were went around by the models.

Keywords*: Stock Predictions, Linear Regression, K- Nearest Neighbors, Auto-ARIMA, Prophet, LSTM*

Table of Contents

Chapter 1: Introduction **05**

Chapter 2: Review of Literature **06**

Chapter 3: Problem Statement and Datasets **09-11**

3.1 Problem Statement 09

3.2 Objective function 09

3.3 Datasets 10

3.4 Feature Selection 11

Chapter 4: Models **12-16**

4.1 Linear Regression 12

4.2 KNN 13

4.3 Auto Arima 15

4.4 Prophet 15

4.5 Long Short Term Memory 16

Chapter 5: Implementation, Results and Analysis **17-34**

5.1 Tata Global Beverages Dataset 17-27

5.1.1 Linear Regression Implementation, Results and Analysis 18

5.1.2 KNN Implementation, Results and Analysis 20

5.1.3 Auto Arima implementation, Results and Analysis 22

5.1.4 Prophet Implementation, Results and Analysis 24

5.1.5 LSTM Implementation, Results and Analysis 26

5.2 Tesla Stock Dataset 28-30

5.2.1 Linear Regression Implementation, Results and Analysis` 30

5.3 Google Datasets(Train and Test datasets) 31-34

5.3.1 LSTM Implementation, Results and Analysis 32

Conclusion **35**

References **36**

Acknowledgement **37**

Chapter 1: Introduction

Stocks are basically an equity investment that represents part ownership in a corporation or a company. It entitles you to a part of the company's earnings and assets. A stock market is a public market where you can buy and sell shares for publicly listed companies. The stocks, also known as equities, represent ownership in the company. The stock exchange is the mediator that allows the buying and selling of shares. Stock markets help companies to raise capital. It helps generate personal wealth. Stock markets serve as an indicator of the state of the economy. It is a widely used source for people to invest money in companies with high growth potential.

Predictions of market values is great importance to help you in maximizing the profits of your stock option purchase while keeping the risk at low and this is important because you need to invest your money in stock which is going to increase in value over and not decrease. There is a lot of complicated financial indicators and also the fluctuators of the stock market is very very violent. As the technology is getting advanced the opportunity to gain study fortune from the stock market is increased and it also helps experts to find out the most informative indicators to make better predictions. Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behavior, and so on. All these factors combine to make share prices dynamic and volatile. This makes it very difficult to predict stock prices with high accuracy. Using features like the latest announcements about an organization, their quarterly revenue results, etc., machine learning techniques have the potential to unearth patterns and insights we didn't see before, and these can be used to make unerringly accurate predictions.

Chapter 2: Review of Literature

A systematic literature review methodology is used to identify relevant peer-reviewed journal articles from the past twenty years and categorize studies that have similar methods and contexts.

Each of the articles fits into one of the following four categories:
(a) artificial neural network studies,
(b) support vector machine studies,
(c) studies using genetic algorithms with other techniques, and
(d) studies using hybrid or other artificial intelligence approaches.

Studies in each category are reviewed to identify common findings, unique findings, limitations, and areas that need further investigation. The final section provides overall conclusions and directions for future research.

Studies using Artificial Neural Network(ANN)

ANNs are computational models based on biological neural networks. In the network, sets of nodes are grouped into layers starting with an input layer and ending with an output layer. Signals are transmitted (propagated) through the connected nodes as they learn based on examples and attempt to reduce the level of prediction error. As the system is working to improve its performance, weights are adjusted for the signals between connected nodes.

> Jasic and Wood (2004) developed an artificial neural network to predict daily stock market index returns using data from several global stock markets. The focus is on trying to support profitable trading. A method is introduced based on univariate neural networks using untransformed data inputs to provide short-term stock market index return predictions. The prediction performance for the neural network is evaluated against a benchmark linear autoregressive model and prediction improvement is confirmed when applied to the S&P 500 and DAX indices.
>
> Chong, Han and Park (2017) analyze deep learning networks for stock market analysis and prediction. Deep learning networks extract features from a large set of raw data without relying on prior knowledge of predictors which makes it useful for high frequency stock market prediction. Empirical results suggest that deep neural networks can extract additional information from the residuals of the autoregressive model and improve predictive performance.

Studies using Support Vector Machine(SVM)

An SVM model represents the examples as points in a space with the goal of creating a gap between the categories that is as wide as possible. New examples are classified based on the category in which they most likely belong.

Lee (2009) developed a prediction model based on a support vector machine with a hybrid feature selection method to predict the trend of stock markets. This proposed hybrid feature selection method, named F-score and Supported Sequential Forward Search (F_SSFS), combines the advantages of filter methods and wrapper methods to select the optimal feature subset from the original feature set. Experimental results show that the proposed SVM-based model combined with F_SSFS has the highest level of predictive accuracy and generalization in comparison with the other three feature selection methods.

Machine learning model using data from the National Stock Exchange (NSE) of India Limited for the time period from January 1, 2007 to December 31, 2010. Das and Padhy (2012) use two machine learning techniques: backpropagation (BP) and SVM to predict future prices in the Indian stock market. The performance of these techniques are compared and it is observed that SVM provides better results when compared with the results from the BP technique.

Studies using Genetic Algorithms(GA) with other techniques

A genetic algorithm is a form of evolutionary algorithm. The evolutionary process begins with a set of randomly generated problem solutions. In each iterative generation, the fitness of each solution is measured by an objective function. The solutions with higher fitness are retained and combined with other high fitness solutions to create a new generation of solutions. Parent solutions combine to create a new child solution that retains some characteristics from both original solutions. This process continues until a certain number of generations has been created or the population of solutions reaches a satisfactory fitness level.

Kim and Han (2000), they propose a genetic algorithm approach to feature discretization and the determination of connection weights for artificial neural networks to predict the value of a stock price index. In this study, the GA is employed not only to improve the learning algorithm, but also to reduce the complexity in the feature space. The research data used is the daily Korea stock price index (KOSPI) from January 1989 to December 1998. Experimental results show that the GA approach incorporated into the feature discretization model outperforms the other two conventional models.

A dynamic fuzzy model is proposed by Chiu and Chen (2009) in combination with a SVM to explore stock market dynamics. A GA adjusts the influential degree of each input variable dynamically. Stock and futures market data from January 2003 to December 2004 is taken from the Taiwan Stock Exchange Corporation. The experimental results show that the model is more accurate when compared with alternative prediction methods.

Studies using hybrid or Artificial Intelligence approaches

This final category describes studies that have used other unique, or multi-method, artificial intelligence techniques in this problem domain**.**

O, Lee, Lee and Zhang (2006) present a new stock trading method that incorporates dynamic asset allocation in a reinforcement-learning framework. The proposed asset allocation strategy, called meta policy (MP), is designed to utilize the temporal information from both stock recommendations and the ratio of the stock fund over the asset. Experimental results using Korean stock market (KOSPI) index data from 1998 to 2003 show that the proposed MP method outperforms other fixed asset-allocation strategies and reduces the risks inherent for local traders.

Chapter 3: Problem statement and Datasets

3.1 Problem statement

Time Series forecasting & modeling plays an important role in data analysis. Time Series is being widely used in analytics & data science. Stock prices are volatile in nature and price depends on various factors. The main aim of this project is to predict stock prices using different models and analyze the best models.

3.2 Objective function

Broadly the stock market analysis is divided into two parts. They are,

- Fundamental Analysis involves analyzing the company's future profitability on the basis of its current business environment and financial performance.
- Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

our objective is on the technical analysis part. We implement a mix of machine learning algorithms to predict the future stock prices of company starting with simple algorithms like linear regression, KNN, and then move on to advanced techniques like Auto-ARIMA and LSTM.

The core idea behind this project is to showcase how these algorithms were implemented.

3.3 Datasets

There are three different datasets used for the implementation of models.

We using

1. Tata Global Beverages from Quandl (from 08-10-2013 to 08-10-2018)
2. Tesla from the kaggle. (from 06-07-2010 to 08-03-2019)
3. Google data sets from the yahoo finance including the train dataset(from 01-03-2012 to 12-30-2016) and test dataset(from 13-08-2018 to 13-08-2019) separately

E.g. Tata global beverages dataset sample

Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
08-10-2018	208	222.25	206.85	216	215.15	4642146	10062.83
05-10-2018	217	218.6	205.9	210.25	209.2	3519515	7407.06
04-10-2018	223.5	227.8	216.15	217.25	218.2	1728786	3815.79
03-10-2018	230	237.5	225.75	226.45	227.6	1708590	3960.27
01-10-2018	234.55	234.6	221.05	230.3	230.9	1534749	3486.05
28-09-2018	234.05	235.95	230.2	233.5	233.75	3069914	7162.35
27-09-2018	234.55	236.8	231.1	233.8	233.25	5082859	11859.95
26-09-2018	240	240	232.5	235	234.25	2240909	5248.6
25-09-2018	233.3	236.75	232	236.25	236.1	2349368	5503.9
24-09-2018	233.55	239.2	230.75	234	233.3	3423509	7999.55

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1235 entries, 0 to 1234
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  1235 non-null   object
 1   Open                  1235 non-null   float64
 2   High                  1235 non-null   float64
 3   Low                   1235 non-null   float64
 4   Last                  1235 non-null   float64
 5   Close                 1235 non-null   float64
 6   Total Trade Quantity  1235 non-null   int64
 7   Turnover (Lacs)       1235 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 77.3+ KB
```

3.4 Feature Selection

There are multiple variables in the dataset – date, open, high, low, last, close, total_trade_quantity, and turnover.

- The columns *Open* and *Close* represent the starting and final price at which the stock is traded on a particular day.
- *High*, *Low* and *Last* represent the maximum, minimum, and last price of the share for the day.
- *Total Trade Quantity* is the number of shares bought or sold in the day and *Turnover (Lacs)* is the turnover of the particular company on a given date.

Another important thing to note is that the market is closed on weekends and public holidays.

The profit or loss calculation is usually determined by the closing price of a stock for the day, hence we will consider the closing price as the target variable.

Chapter 4: Models

Initially, regression models were considered like Linear Regression, then going to time series forecasting models like Auto-ARIMA, Prophet, and Long Short Term Memory(LSTM).

4.1 Linear regression

The linear regression model is an analysis that is used to predict the value of a variable based on another variable. It performs the task to predict a dependent variable based on independent variables.

Equation:

$$Y=a+bX+e$$

Where,

a=intercept of line

b=slope of the line

e=error term

This equation can be used to predict the value of a target variable based on given predictor values.

<u>Types of linear regression:</u>

a. Simple linear regression- It is a regression model used to estimate the relationship between one independent variable and one dependent variable.

b.Multiple linear regression- It is a regression model used to estimate the relationship between more than one independent variable and one dependent variable.

For our problem statement, we do not have a set of independent variables. We have only the dates instead. Let us use the date column to extract features like – day, month, year, mon/fri etc. and then fit a linear regression model.

4.2 KNN

KNN is a supervised machine learning algorithm that can be used to solve both classification and regression problem statements. It is a non-parametric algorithm, which means it does not make assumptions on underlying data.

KNN is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset.

KNN classification:

Attempts to predict the class to which the output variable belongs by computing the local probability.

KNN Regression:

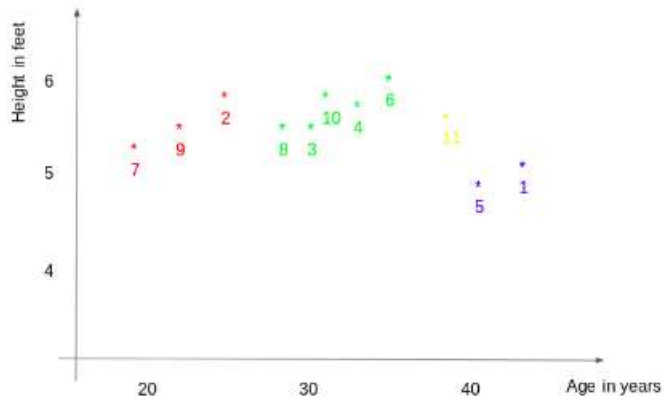
This algorithm tries to predict the value of the output variable by using a local average.

- KNN can be used for regression problem statements, the predicted value is given by the average of the values of its K nearest neighbors.

Let me explain this with a simple example.

Consider the height and age for 11 people. On the basis of given features (‘Age’ and ‘Height’), the table can be represented in a graphical format as shown below:

ID	Age	Height	Weight
1	45	5	77
2	26	5.11	47
3	30	5.6	55
4	34	5.9	59
5	40	4.8	72
6	36	5.8	60
7	19	5.3	40
8	28	5.8	60
9	23	5.5	45
10	32	5.6	58
11	38	5.5	?



To determine the weight for ID #11, kNN considers the weight of the nearest neighbors of this ID. The weight of ID #11 is predicted to be the average of its neighbors. If we consider three neighbors (k=3) for now, the weight for ID#11 would be = (77+72+60)/3 = 69.66 kg.

ID	Height	Age	Weight
1	5	45	77
5	4.8	40	72
6	5.8	36	60

4.3 Auto Arima

An Autoregresion integrated moving average or ARIMA, is a statistical analysis model that uses time-series data to better understand the data set or predict future trends.
This model is used to understand past data or predict future data in a series

Equation:

$$y'(t) = c + \phi 1 * y'(t-1) + \cdots + \phi p * y'(t-p) + \theta 1 * \varepsilon(t-1) + \cdots + \theta q * \varepsilon(t-q) + \varepsilon t.$$

Where,

p= number of lag observations in the model=lag order

d=the number of times that raw observations are differenced= degree of differencing

q= size of moving average= order of moving average.

Parameter tuning for ARIMA consumes a lot of time. So we will use auto ARIMA which automatically selects the best combination of (p,q,d) that provides the least error.

4.4 Prophet

There are a number of time series techniques that can be implemented on the stock prediction dataset, but most of these techniques require a lot of data preprocessing before fitting the model. Prophet, designed and pioneered by Facebook, is a time series forecasting library that requires no data preprocessing and is extremely simple to implement. The input for Prophet is a data frame with two columns: date and target (ds and y).

Prophet tries to capture the seasonality in the past data and works well when the dataset is large.

4.5 Long Short Term Memory

LSTM is used in the field of deep learning. It is a variety of recurrent neural networks that are capable of learning long-term dependencies, especially in sequence prediction problems.

LSTM is a time series forecasting model that can predict future values based on previous, sequential data. It uses a series of 'gates' which control how the information in a sequence of data comes into, is stored, and leaves the networks.

LSTM has three gates:

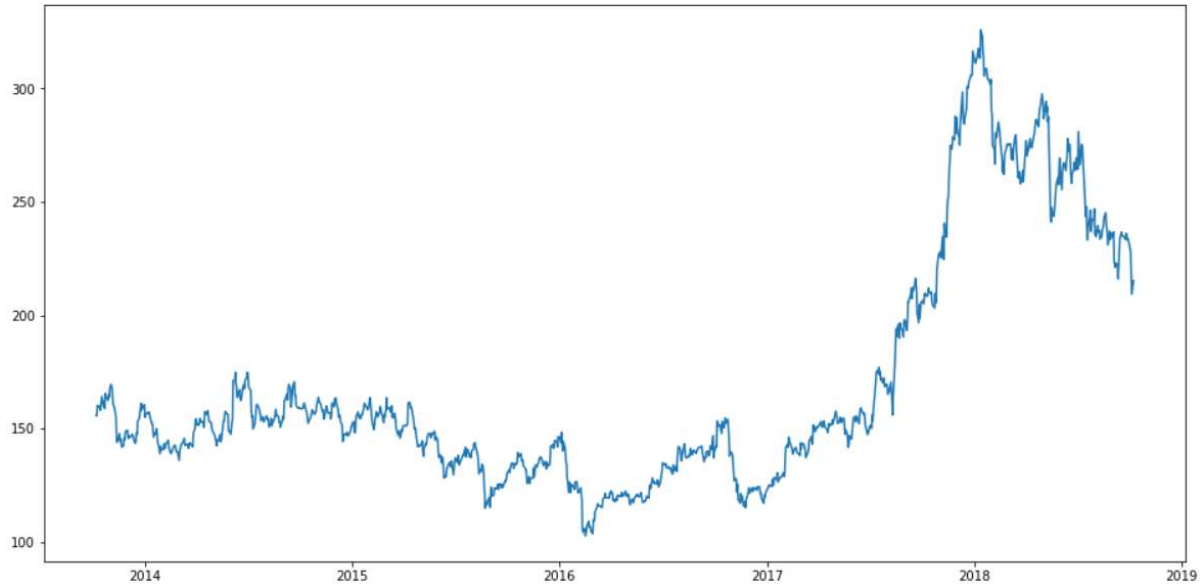
- The input gate: The input gate adds information to the cell state
- The forget gate: It removes the information that is no longer required by the model
- The output gate: Output Gate at LSTM selects the information to be shown as output

Chapter 5: Implementation, Results and Analysis

5.1 Tata Global Beverages dataset

We will first sort the dataset in ascending order and then create a separate dataset so that any new feature created does not affect the original data.

The graph has been plotted for the given dates and closing values of the dataset.



5.1.1 Linear Regression Implementation, Results and analysis

We will first sort the dataset in ascending order and then create a separate dataset so that any new feature created does not affect the original data.

We will now split the data into train and validation sets to check the performance of the model.

```
#split into train and validation
train = new_data[:987]
valid = new_data[987:]

x_train = train.drop('Close', axis=1)
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

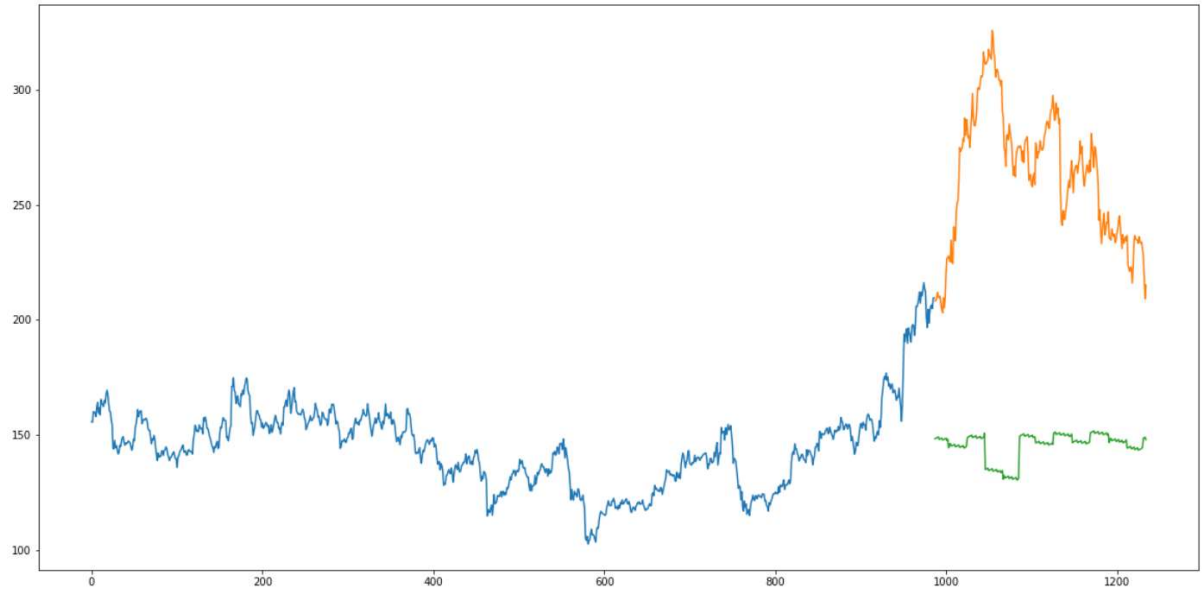
```
LinearRegression()
```

Now we acquire the results using following:

```
#make predictions and find the rmse
preds = model.predict(x_valid)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

```
121.16291596522808
```

The RMSE value is higher than the previous technique, which clearly shows that linear regression has performed poorly. Let's look at the plot and understand why linear regression has not done well:



Linear regression is a simple technique and quite easy to interpret, but there are a few obvious disadvantages. One problem in using regression algorithms is that the model overfits to the date and month column. Instead of taking into account the previous values from the point of prediction, the model will consider the value from the same *date* a month ago, or the same *date/month* a year ago.

5.1.2 KNN Implementation, Results and analysis

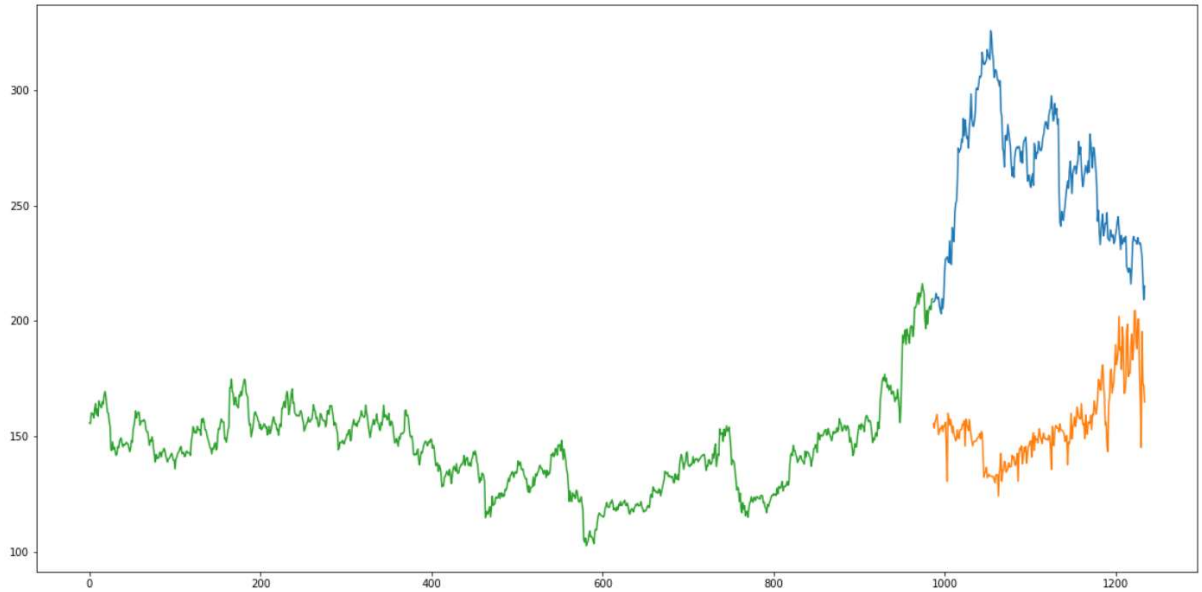
Importing the required libraries of KNN and using the same train and validation set from the linear regression model implementation.

```
#importing libraries
from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

#scaling data
x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x_valid)
x_valid = pd.DataFrame(x_valid_scaled)

#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)

#fit the model and make predictions
model.fit(x_train,y_train)
preds = model.predict(x_valid)
```

Interpreting the results as fpllows:

```
#rmse
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms

115.17086550026721
```

Plotting the graph for KNN model:



The RMSE value is almost similar to the linear regression model and the plot shows the same pattern. Like linear regression, kNN also identified a drop in January 2018 since that has been the pattern for the past years. We can safely say that regression algorithms have not performed well on this dataset.

5.1.3 Auto-ARIMA Implementation, Results and analysis

Importing the Auto-ARIMA model libraries and making the predictions.

```
from pmdarima.arima import auto_arima
data = df.sort_index(ascending=True, axis=0)

train = data[:987]
valid = data[987:]

training = train['Close']
validation = valid['Close']

model = auto_arima(training, start_p=1, start_q=1,max_p=3, max_q=3, m=12,start_P=0, seasonal=True,d=1, D=1, trace=True,error_action='ignore',suppress_warnings=True)
model.fit(training)

forecast = model.predict(n_periods=248)
forecast = pd.DataFrame(forecast,index = valid.index,columns=['Prediction'])
```

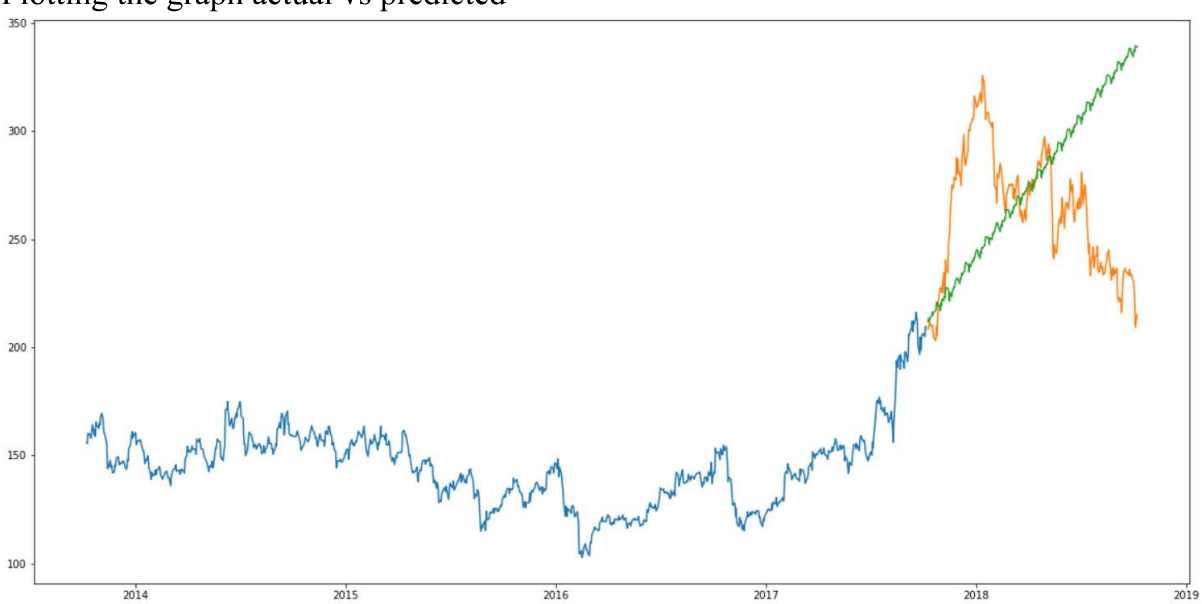
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=inf, Time=7.34 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=5477.259, Time=0.10 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=5168.276, Time=1.19 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=5.77 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=5479.164, Time=0.13 sec
 ARIMA(1,1,0)(2,1,0)[12]             : AIC=5076.280, Time=2.25 sec
 ARIMA(1,1,0)(2,1,1)[12]             : AIC=inf, Time=21.97 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=inf, Time=7.72 sec
 ARIMA(0,1,0)(2,1,0)[12]             : AIC=5074.398, Time=1.51 sec
 ARIMA(0,1,0)(1,1,0)[12]             : AIC=5166.277, Time=0.66 sec
 ARIMA(0,1,0)(2,1,1)[12]             : AIC=inf, Time=9.99 sec
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=inf, Time=2.20 sec
 ARIMA(0,1,1)(2,1,0)[12]             : AIC=5076.285, Time=1.12 sec
 ARIMA(1,1,1)(2,1,0)[12]             : AIC=5077.935, Time=2.36 sec
 ARIMA(0,1,0)(2,1,0)[12] intercept   : AIC=5076.382, Time=3.42 sec

Best model:  ARIMA(0,1,0)(2,1,0)[12]
Total fit time: 67.818 seconds
```

Getting the results as follows:

```
rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-np.array(forecast['Prediction'])),2)))
rms
```

```
54.54322744872866
```

Plotting the graph actual vs predicted



As we saw earlier, an auto ARIMA model uses past data to understand the pattern in the time series. Using these values, the model captured an increasing trend in the series. Although the predictions using this technique are far better than that of the previously implemented machine learning models, these predictions are still not close to the real values.

As its evident from the plot, the model has captured a trend in the series, but does not focus on the seasonal part.

5.1.4 Prophet Implementation, Results and analysis

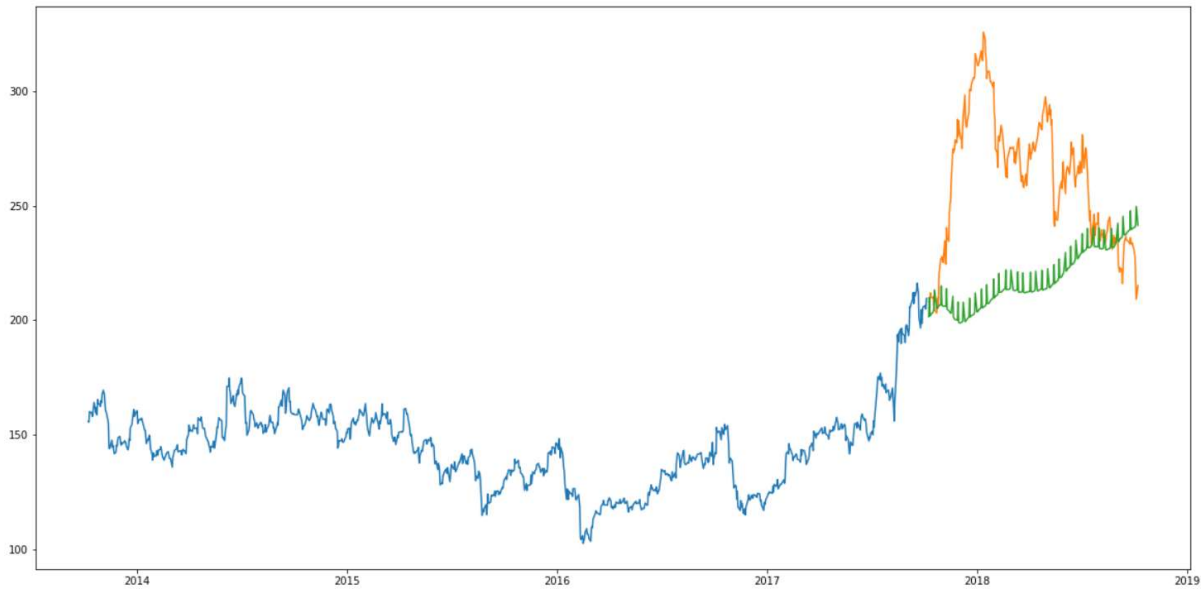
We will implement a time series model that takes both trend and seasonality of a series into account in the prophet model.

```
#importing prophet
from fbprophet import Prophet

#creating dataframe
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]

new_data['Date'] = pd.to_datetime(new_data.Date,format='%Y-%m-%d')
new_data.index = new_data['Date']

#preparing data
new_data.rename(columns={'Close': 'y', 'Date': 'ds'}, inplace=True)

#train and validation
train = new_data[:987]
valid = new_data[987:]

#fit the model
model = Prophet()
model.fit(train)

#predictions
close_prices = model.make_future_dataframe(periods=len(valid))
forecast = model.predict(close_prices)
```

The results were interpreted as follows:

```
#rmse
forecast_valid = forecast['yhat'][987:]
rms=np.sqrt(np.mean(np.power((np.array(valid['y'])-np.array(forecast_valid)),2)))
rms

57.31515213754124
```

The plot is drawn as below:



Prophet (like most time series forecasting techniques) tries to capture the trend and seasonality from past data. This model usually performs well on time-series datasets, but fails to live up to its reputation in this case.

As it turns out, stock prices do not have a particular trend or seasonality. It highly depends on what is currently going on in the market and thus the prices rise and fall. Hence forecasting techniques like ARIMA, SARIMA and Prophet would not show good results for this particular problem.

5.1.5 LSTM Implementation, Results and analysis

Let us go ahead and try another advanced technique – Long Short Term Memory (LSTM).

```
#importing required libraries
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM

#creating dataframe
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]

#setting index
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)

#creating train and test sets
dataset = new_data.values

train = dataset[0:987,:]
valid = dataset[987:,:]

#converting dataset into x_train and y_train
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)

#predicting 246 values, using past 60 from the train data
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)

X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)

X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

The Results of the LSTM is as

```
rms=np.sqrt(np.mean(np.power((valid-closing_price),2)))
rms

11.393317415942564
```

The Plot that got from LSTM model is

The LSTM model can be tuned for various parameters such as changing the number of LSTM layers, adding dropout value or increasing the number of epochs. This model predicted the values very accurate than other four models

5.2 Tesla Stock Dataset

The dataset Tata Global Beverages is implemented over Linear Regression.

Here is the top-5 rows from the dataset

```
#print the head
tesla.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	06-07-2010	20.000000	20.000000	15.830000	16.110001	16.110001	6866900
1	07-07-2010	16.400000	16.629999	14.980000	15.800000	15.800000	6921700
2	08-07-2010	16.139999	17.520000	15.570000	17.459999	17.459999	7711400
3	09-07-2010	17.580000	17.900000	16.549999	17.400000	17.400000	4050600
4	12-07-2010	17.950001	18.070000	17.000000	17.049999	17.049999	2202500

The total number of rows and columns, datatypes of variables and checking whether any null values are present

```
tesla.info()
```

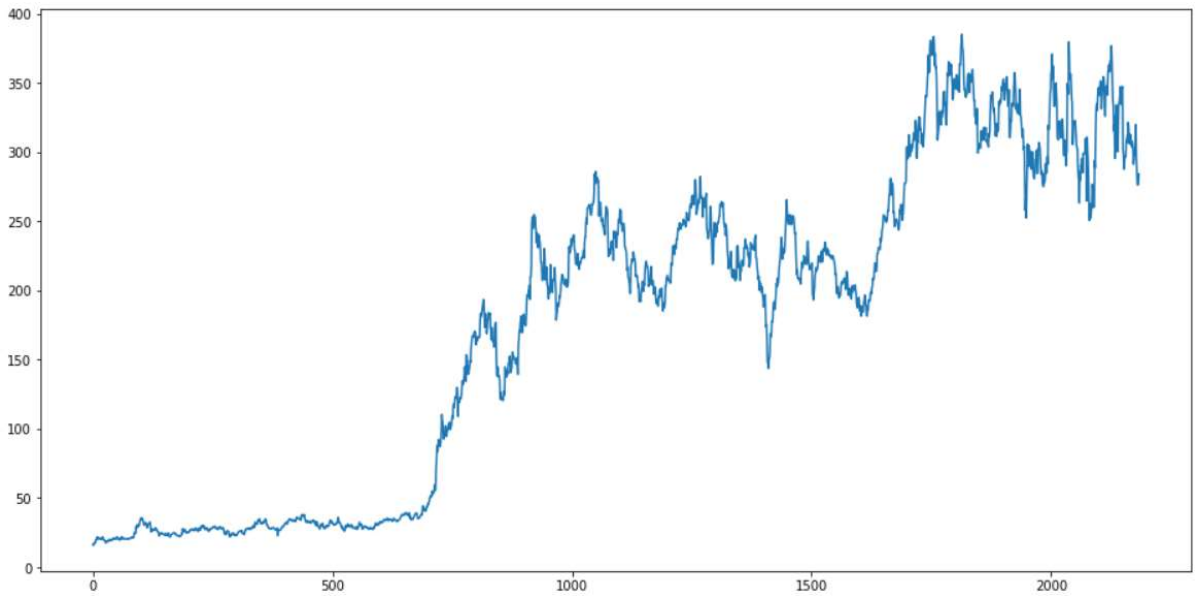
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2184 entries, 0 to 2183
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       2184 non-null   object
 1   Open       2184 non-null   float64
 2   High       2184 non-null   float64
 3   Low        2184 non-null   float64
 4   Close      2184 non-null   float64
 5   Adj Close  2184 non-null   float64
 6   Volume     2184 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 119.6+ KB
```

The summary statistics for the dataset is

```
tesla.describe()
```

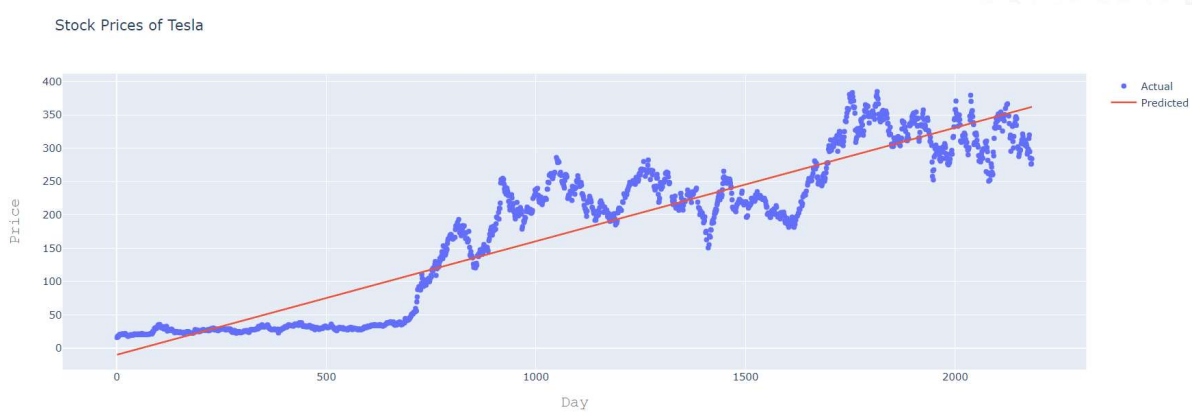
	Open	High	Low	Close	Adj Close	Volume
count	2184.000000	2184.000000	2184.000000	2184.000000	2184.000000	2.184000e+03
mean	175.679538	178.734757	172.441809	175.677550	175.677550	5.055842e+06
std	115.514138	117.308992	113.581785	115.511987	115.511987	4.531720e+06
min	16.139999	16.629999	14.980000	15.800000	15.800000	1.185000e+05
25%	33.127500	33.977500	32.500000	33.207500	33.207500	1.573200e+06
50%	204.965003	208.095001	201.659996	204.965003	204.965003	4.151200e+06
75%	261.905014	264.725006	256.074990	261.380005	261.380005	6.825525e+06
max	386.690002	389.609985	379.350006	385.000000	385.000000	3.716390e+07

The initial graph has been plotted for the given dates and closing values of the dataset.



5.2.1 Linear Regression Implementation, Results and analysis

Importing the mixmaxscalar and standardscaler from sci-kit learn library for preprocessing. We first split the dataset into train and test sets by giving the test size as 30%. We do feature scaling using the standardscalar function. Standardizing the dataset is required for any ML algorithm they might behave badly if the individual features not normally distributed. Then we fit the linear regression model into the plot the graphs and make results.



Calculating the MNSE, R2 score values

```
#calculate scores for model evaluation
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(Y_train, lm.predict(X_train))}\t{r2_score(Y_test, lm.predict(X_test))}
{'MSE'.ljust(10)}{mse(Y_train, lm.predict(X_train))}\t{mse(Y_test, lm.predict(X_test))}
'''

print(scores)
```

```
Metric           Train               Test
r2_score  0.8653903102262495    0.8629979614862864
MSE       1800.766851351372     1801.356966055652
```

5.3 Google Datasets (Train and Test)

Using both the different train and test datasets for the Google stock predictions.

The training dataset contains from the date 01-03-2012 to 12-30-2016. The testing dataset contains from the date 13-08-2018 to 13-08-2019.

The top five rows from the train dataset

```
data = pd.read_csv(r'/content/Google_train_data.csv')
data.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

getting the datatypes and rows and columns from the train dataset

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1258 non-null   object
 1   Open    1258 non-null   float64
 2   High    1258 non-null   float64
 3   Low     1258 non-null   float64
 4   Close   1258 non-null   object
 5   Volume  1258 non-null   object
dtypes: float64(3), object(3)
memory usage: 59.1+ KB
```

We do some preprocessing of data to transform the closing price variable to a numeric variable. Drop nonavailable data snd keep the close column for training data.

Now its changed from object to float as follows

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1149 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1149 non-null   object
 1   Open    1149 non-null   float64
 2   High    1149 non-null   float64
 3   Low     1149 non-null   float64
 4   Close   1149 non-null   float64
 5   Volume  1149 non-null   object
dtypes: float64(4), object(2)
memory usage: 62.8+ KB
```

5.3.1 LSTM Implementation, Results and analysis

we use fit_transform method to transform values and define the timesteps and batch size. Then build a LSTM model of four layers of the LSTM network and all followed by a dropout layer. On the top of it has final dense layer and compiling these using adam optimizer and mean square as loss function. We train the data using epochs and batch size having model.fit function.

```
sc = MinMaxScaler(feature_range=(0,1))
trainData = sc.fit_transform(trainData)
trainData.shape
```

```
(1149, 1)
```

```
X_train = []
y_train = []

for i in range (60,1149): #60 : timestep // 1149 : Length of the data
  X_train.append(trainData[i-60:i,0])
  y_train.append(trainData[i,0])

X_train,y_train = np.array(X_train),np.array(y_train)
```

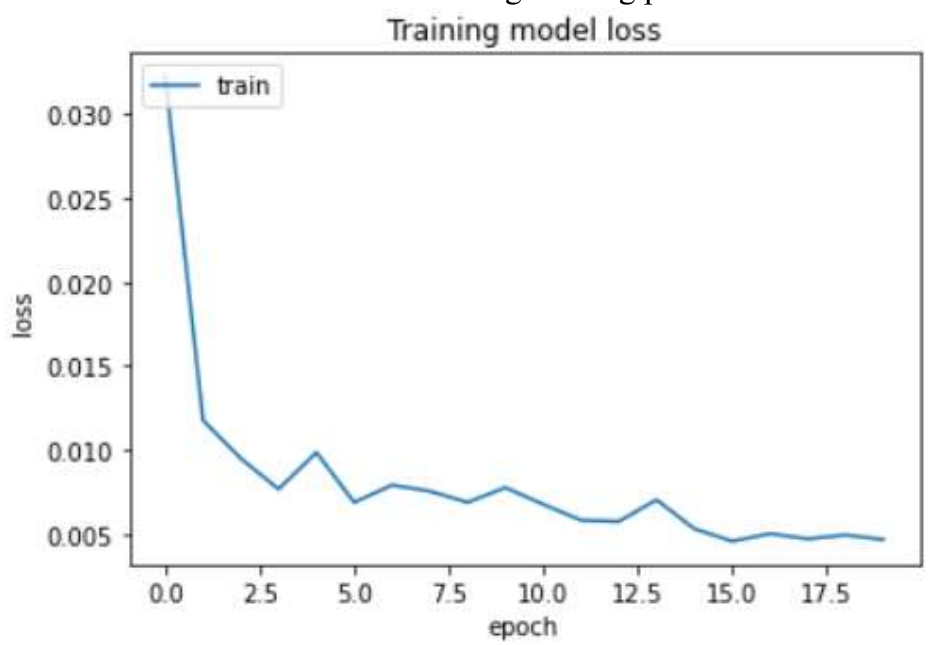
```
X_train = np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1)) #adding the batch_size axis
X_train.shape
```

```
(1089, 60, 1)
```

```
model = Sequential()

model.add(LSTM(units=100, return_sequences=True, input_shape=(X_train.shape[1],1)))
model.add(Dropout(0.2))

model.add(LSTM(units=100, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=100, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=100, return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(units=1))
model.compile(optimizer='adam',loss="mean_squared_error")
```

```
hist = model.fit(X_train, y_train, epochs = 20, batch_size=32, verbose=2)
```

Visualize the loss that occurred during training process



Now testing the new dataset takes place converting the close column to numeric and predicting the test data.

```
testData = pd.read_csv(r'/content/Google_test_data.csv')
testData['Close']=pd.to_numeric(testData.Close,errors='coerce')
testData = testData.dropna()
testData = testData.iloc[:,4:5]
y_test = testData.iloc[60:,0:].values
#input array for the model
inputClosing = testData.iloc[:,0:].values
inputClosing_scaled = sc.transform(inputClosing)
inputClosing_scaled.shape
X_test = []
length = len(testData)
timestep = 60
for i in range(timestep,length):
  X_test.append(inputClosing_scaled[i-timestep:i,0])
X_test = np.array(X_test)
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
X_test.shape
```

Then visualize the plot

Getting the results of it as follows

```
rms=np.sqrt(np.mean(np.power((y_test-predicted_price),2)))
rms

68.2313328556336
```

Conclusion

The popularity of the stock market is growing extremely fast which encourages researchers to find out new methods for the prediction using new techniques. The forecasting techniques not only help researchers but also helps investors or any person dealing with the stock market in order to help predict the stock indices forecasting model with good accuracy is required. The use of Machine Learning techniques in the prediction of algorithms has yielded promising results. Among all the five models the LSTM is giving the promising results and good accuracy. It has led to the conclusion that it is possible to predict the stock market with more accuracy and efficiency using Machine Learning models. Furthermore, other models could be studied to check for the accuracy rate resulted by them.

References

[1] https://finance.yahoo.com/quote/GOOG/history/

[2] https://www.kaggle.com/datasets/timoboz/tesla-stock-data-from-2010-to-2020

[3] https://www.quandl.com/

[4]https://www.researchgate.net/publication/341482418_A_Survey_on_Stock_Market_Prediction_Using_Machine_Learning_Techniques

[5]https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1435&context=jitim

[6]https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning

[7]https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation

[8] https://www.youtube.com/watch?v=lncoLfue_Y4

Acknowledgment

It gives us immense pleasure and satisfaction in presenting this machine learning course project **'Stock Price Prediction'**.

We would like to express our deep sense of gratitude towards **Dr.B. Jayalakshmi and Dr.Deepak K T** for their continuous guidance and assistance.

We also wish to express our heartfelt gratitude to **Pradyoth Hegde** and all those who have played a crucial role in the research for this project, without their active cooperation the preparation of this project could not have been completed within the specified time limit.

Batina Koushik 19bcs019

Burra Manikanta 19bcs026

Dasari Rishikesh 19bcs035

Mankena Roja 19bcs067