GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Stock Market Prediction Using Machine Learning

Final report of project 2

Submitted by

KISHAN GIRI

(1613101333/16SCSE101616)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

M. Vivek Anand

Assistant Professor.

APRIL/MAY-2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"STOCK MARKET PREDICTION USING MACHINE LEARNING"** is the bonafide work of **"KISHAN GIRI(1613101333)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

M.VIVEK ANAND , M.Tech.
Assistant Professor
School of Computing Science & Engineering

Abstract:

The main objective of this project is to find the best model to predict the value of the stock market. During the process Of considering various techniques and variables that must be taken into account, we found out that techniques like random forest, support vector machine were not exploited fully. In, this paper we are going to present and review a more feasible method to predict the stock movement with higher accuracy. The first thing we have taken into account is the dataset of the stock market prices from previous year. The dataset was pre-processed and tuned up for real analysis. Hence, our paper will also focus on data preprocessing of the raw dataset. Secondly, after pre- processing the data, we will review the use of random forest, support vector machine on the dataset and the outcomes it generates. In addition, the proposed paper examines the use of the prediction system in real-world settings and issues associated with the accuracy of the overall values given. The paper also presents a machine-learning model to predict the longevity of stock in a competitive market. The successful prediction of the stock will be a great asset for the stock market institutions and will provide real-life solutions to the problems that stock investors face. Case description: Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are widely used for prediction of stock prices and its movements. Every algorithm has its way of learning patterns and then predicting. Artificial Neural Network (ANN) is a popular method which also incorporate technical analysis for making predictions in financial markets.

Discussion and evaluation: Most common techniques used in the forecasting of financial time series are Support Vector Machine (SVM), Support Vector Regression (SVR) and Back Propagation Neural Network (BPNN). In this article, we use neural networks based on three different learning algorithms, i.e., Levenberg-Marquardt, Scaled Conjugate Gradient and Bayesian Regularization for stock market prediction based on tick data as well as 15-min data of an Indian company and their results compared.

Conclusion: All three algorithms provide an accuracy of 99.9% using tick data. The accuracy over 15-min dataset drops to 96.2%, 97.0% and 98.9% for LM, SCG and Bayesian Regularization respectively which is significantly poor in comparison with that of results obtained using tick data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 SCOPE OF THE PROJECT

The stock market is basically an aggregation of various buyers and sellers of stock. A stock (also known as shares more commonly) in general represents ownership claims on business by a particular individual or a group of people. The attempt to determine the future value of the stock market is known as a stock market prediction. The prediction is expected to be robust, accurate and efficient. The system must work according to the real-life scenarios and should be well suited to real-world settings. The system is also expected to take into account all the variables that might affect the stock's value and performance. There are various methods and ways of implementing the prediction system like Fundamental Analysis, Technical Analysis, Machine Learning, Market Mimicry, and Time series aspect structuring. With the advancement of the digital era, the prediction has moved up into the technological realm. The most prominent and [3] promising technique involves the use of Artificial Neural Networks, Recurrent Neural Networks, that is basically the implementation of machine learning. Machine learning involves artificial intelligence which empowers the system to learn and improve from past experiences without being programmed time and again. Traditional methods of prediction in machine learning use algorithms like Backward Propagation, also known as Backpropagation errors. Lately, many researchers are using more of ensemble learning techniques. It would use low price and time lags to predict future highs while another network would use lagged highs to predict future highs. These predictions were used to form stock prices.

Stock market price prediction for short time windows appears to be a random process. The stock price movement over a long period of time usually develops a linear curve. People tend to buy those stocks whose prices are expected to rise in the near future. The uncertainty in the stock market refrain people from investing in stocks. Thus, there is a need to accurately predict the stock market which can be used in a real-life scenario. The methods used to predict the stock market includes a time series forecasting along with technical analysis, machine learning modeling and predicting the variable stock market. The datasets of the stock market prediction model include details like the closing price opening price, the data and various other variables that are needed to predict the object variable which is the price in a given day. The previous model used traditional methods of prediction like multivariate analysis with a prediction time series model. Stock market prediction outperforms when it is treated as a regression problem but performs well when treated as a classification. The aim is to design a model that gains from the market information utilizing machine learning strategies and gauge the future patterns in stock value development. The Support Vector Machine (SVM) can be used for both classification and regression. It has been observed that SVMs are more used in classification based problem like ours. The SVM technique, we plot every single data component as a point in n- dimensional space (where n is the number of features of the dataset available) with the value of feature being the value of a particular coordinate and, hence classification is performed by hyperplane that differentiates the two classes explicitly. A stock market is a platform for trading of a company's stocks and derivatives at an agreed price. Supply and demand of shares drive the stock market. In any country stock market is one of the most emerging sectors. Nowadays, many people are indirectly or dir- ectly related to this sector. Therefore, it becomes essential to know about market trends. Thus, with the development of the stock market, people are interested in forecasting stock price. But, due to dynamic nature and liable to quick changes in stock price, predic- tion of the stock price becomes a challenging task. Stock markets are mostly a non-parametric, non-linear, noisy and deterministic chaotic system (Ahangar et al. 2010). As the technology is increasing, stock traders are moving towards to use Intelligent Trading Systems rather than fundamental analysis for predicting prices of stocks, which helps them to take immediate investment decisions. One of the main aims of a trader is to predict the stock price such that he can sell it before its value decline, or buy the stock before the price rises. The efficient market hypothesis states that it is not possible to predict stock prices and that stock behaves in the random walk. It seems to be very difficult to replace the professionalism of an experienced

trader for predicting the stock price. But because of the availability of a remarkable amount of data and technological advancements we can now formulate an appropriate algorithm for pre- diction whose results can increase the profits for traders or investment firms. Thus, the accuracy of an algorithm is directly proportional to gains made by using the algorithm.

1.2 PROBLEM STATEMENT

- The existing system fails when there are rare outcomes or predictors, as the algorithm is based on bootstrap sampling .
- The previous results indicate that the stock price is unpredictable when the traditional classifier is used.
- The existence system reported highly predictive values, by selecting an appropriate time period for their experiment to obtain highly predictive scores.
- The existing system does not perform well when there is a change in the operating environment.
- It doesn't focus on external event in the environment , like news events or social media.
- It exploits only one data source, thus highly biased.

CHAPTER 2

Overview of the methodes used

2.1.Data Collection

Data collection is a very basic module and the initial step towards the project. It generally deals with the collection of the right dataset. The dataset that is to be used in the market prediction has to be used to be filtered based on various aspects. Data collection also complements to enhance the dataset by adding more data that are external. Our data mainly consists of the previous year stock prices. Initially, we will be analyzing the Kaggle dataset and according to the accuracy, we will be using the model with the data to analyze the predictions accurately.

2.1.1.Preprocessing

Data pre-processing is a part of data mining, which involves transforming raw data into a more coherent format. Raw data is usually, inconsistent or incomplete and usually contains many errors. The data pre-processing involves checking out for missing values, looking for categorical values, splitting the data-set into training and test set and finally do a feature scaling to limit the range of variables so that they can be compared on common environs.

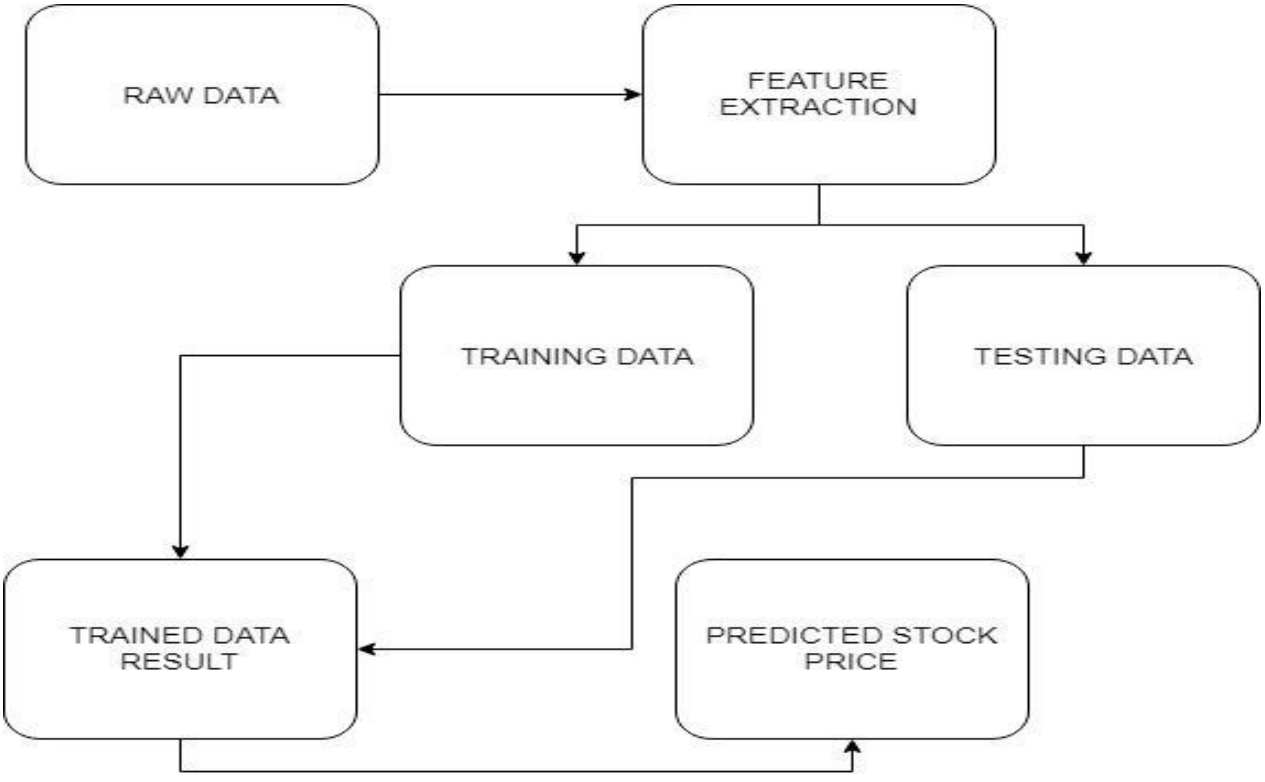


Fig 2.1.1

2.1.2 Training the Machine

Training the machine is similar to feeding the data to the algorithm to touch up the test data. The training sets are used to tune and fit the models. The test sets are untouched, as a model should not be judged based on unseen data. The training of the model includes cross-validation where we get a well-grounded approximate performance of the model using the training data. Tuning models are meant to specifically tune The hyper parameters like the number of trees in a random forest. We perform the entire cross-validation loop on each set of hyper parameter values.
Finally, we will calculate a cross-validated score, for individual sets of hyper parameters. Then, we select the best hyper parameters. The idea behind the training of the model is that we some initial values with the dataset and then optimize the parameters which we want to in the model. This is kept on repetition until we get the optimal values. Thus, we take the predictions from the trained model on the inputs from the test dataset. Hence, it is divided in the ratio of 80:20 where 80% is for the training set and the rest 20% for a testing set of the data.

2.1.3 Data Scoring

The process of applying a predictive model to a set of data is referred to as scoring the data. The technique used to process the dataset is the Random Forest Algorithm. Random forest involves an ensemble method, which is usually used, for classification and as well as regression. Based on the learning models, we achieve interesting results. The last module thus describes how the result of the model can help to predict the probability of a stock to rise and sink based on certain parameters. It also shows the vulnerabilities of a particular stock or entity. The user authentication system control is implemented to make sure that only the authorized entities are accessing the results.

2.1.4 Proposed system

In this proposed system, we focus on predicting the stock values using machine learning algorithms like Random Forest and Support Vector Machines. We proposed the system “Stock market price prediction” we have predicted the stock market price using the random forest algorithm. In this proposed system, we were able to train the machine from the various data points from the past to make a future prediction. We took data from the previous year stocks to train the model. We majorly used two machine-learning libraries to solve the problem. The first one was numpy, which was used to clean and manipulate the data, and getting it into a form ready for analysis. The other was sickest, which was used for real analysis and prediction. The data set we used was from the previous years stock markets collected from the public database available online, 80 % of data was used to train the machine and the rest 20 % to test the data. The basic approach of the supervised learning model is to learn the patterns and relationships in the data from the training set and then reproduce them for the test data. We used the python pandas library for data processing which combined different datasets into a data frame. The tuned up data frame allowed us to prepare the data for feature extraction. The data frame features were date and the closing price for a particular day. We used all these features to train the machine on random forest model and predicted the object variable, which is the price for a given day. We also quantified the accuracy by using the predictions for the test set and the actual values. The proposed system touches different areas of research including data pre-processing, random forest, and soon.

2.2 Methodology(Using LSTM)

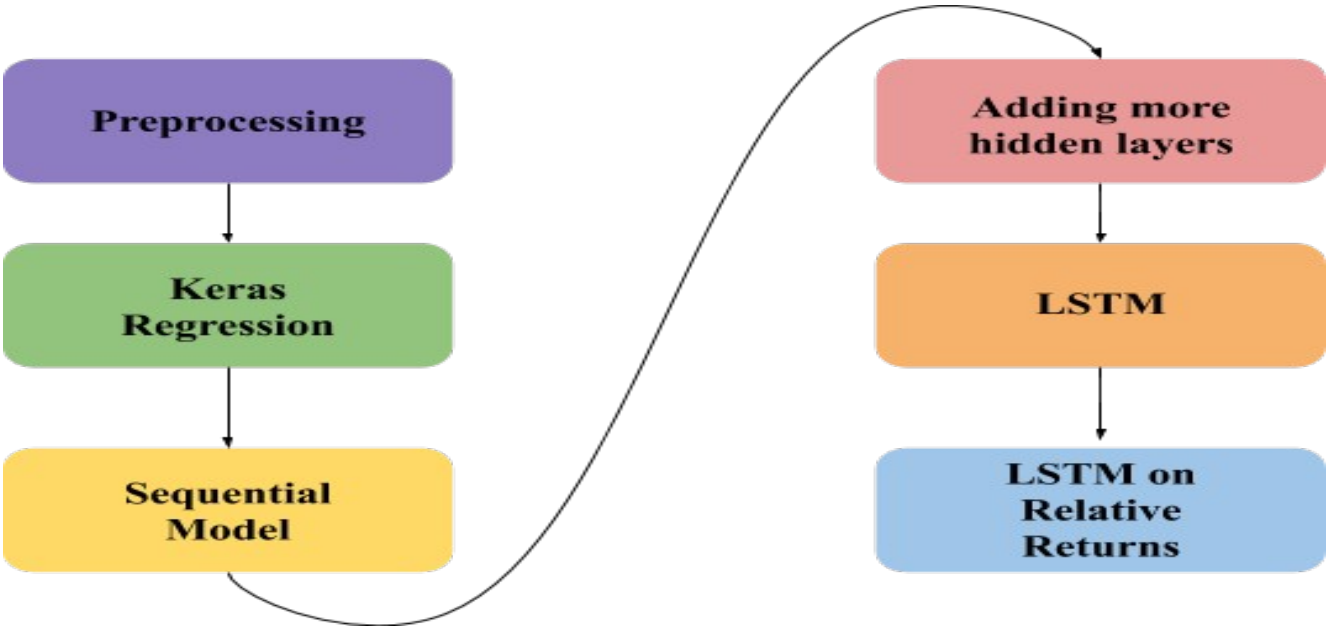


Fig 2.2.1

The first step involves preprocessing the given training data. This includes the following three steps:

1. **Replacing missing data**

   The data points missing in the training data were replaced with the mean of the data in the respective column.

2. **Splitting the data**

   The training data was split into input and output data.

3. **Normalizing the input data**

   Finally, the input data (represented by X) was normalized using the MinMaxScaler Method. The formula used in this method is given below:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

The different approaches adopted in this project utilize regression using Neural Networks to make future predictions of stock returns. The initial approach involved using the **Keras Regression Library and Random Forest Model**. After splitting and normalizing the data, the above-mentioned Regression model was trained on it. This model proved to be overfitting and the loss was 0%. Then the **Sequential Model** was tried. Initially, it

gave a large Mean Square Error. On trying deeper topology, different number of epochs, and varied batch sizes this error, as well as loss, was reduced.

Fig 2.2.2

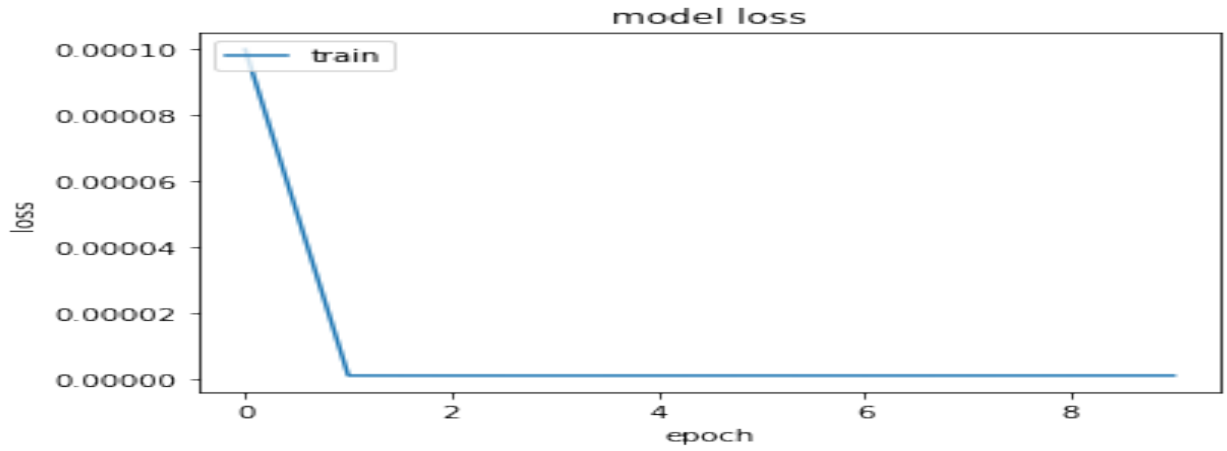


Fig 2.2.3

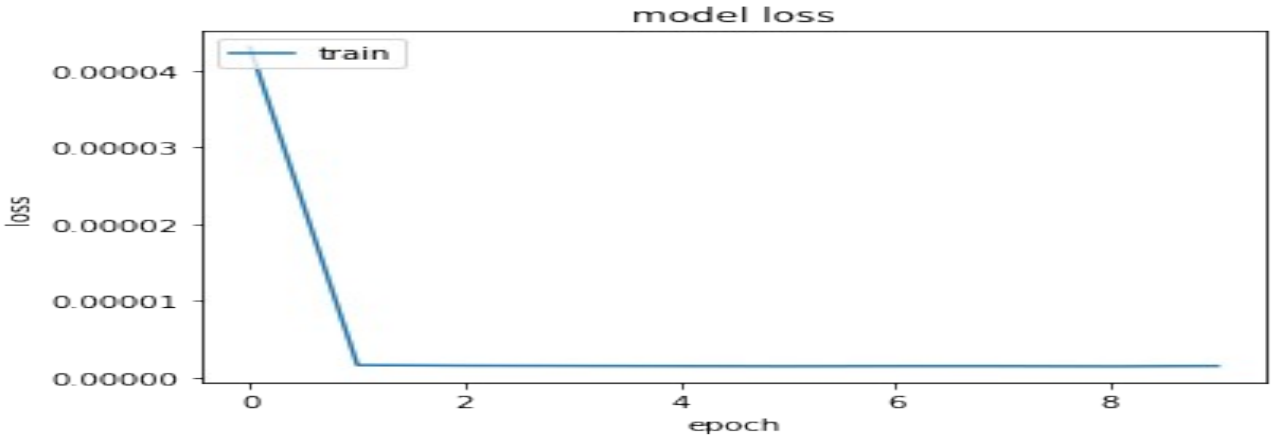


Fig 2.2.3

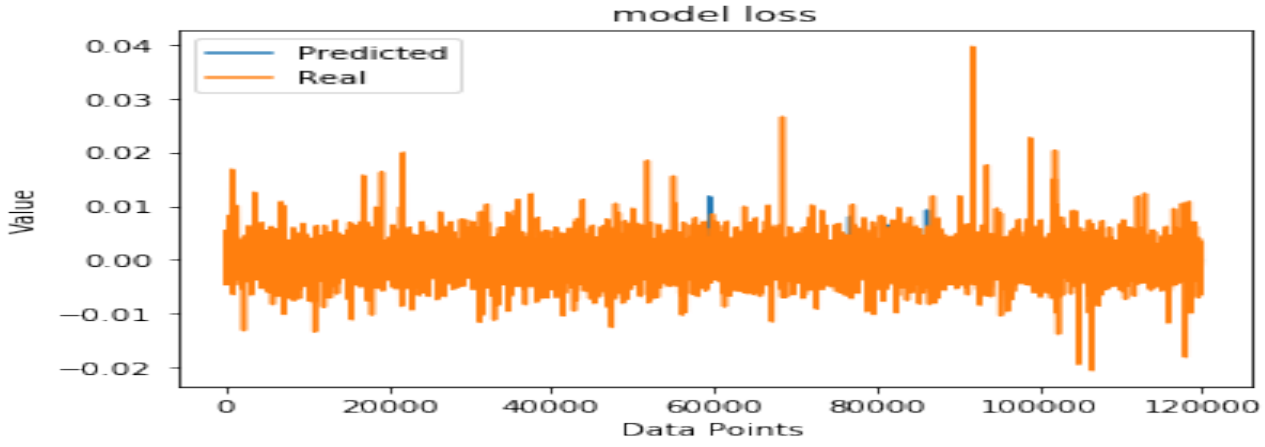
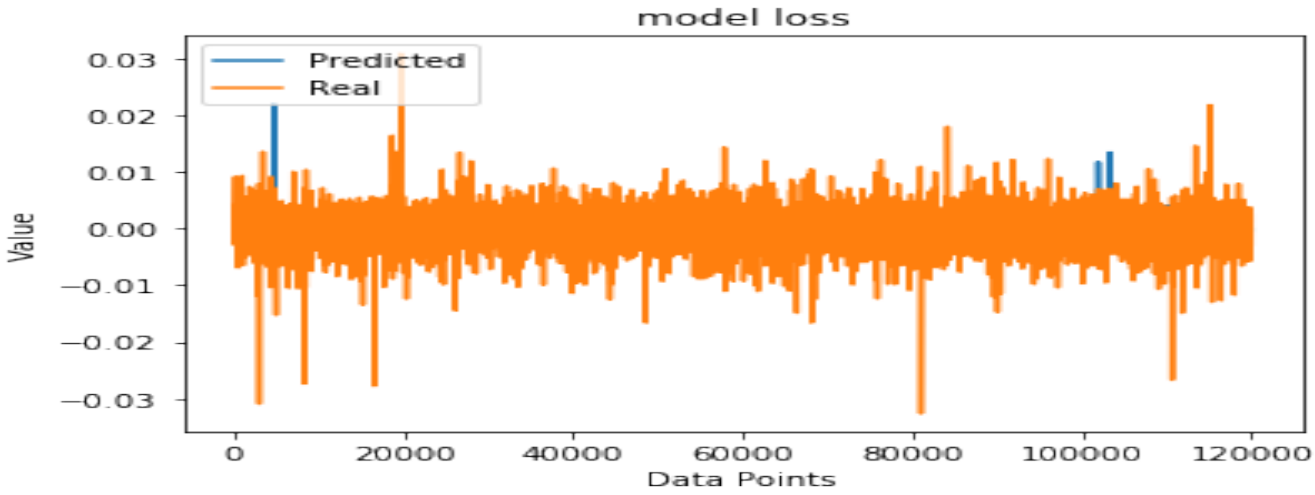


Fig 2.2.4



The final approach used **Long Short Term Memory networks** that are a special kind of Recurrent Neural Networks. This model was trained on the data to give an acceptable loss and MSE.

Fig 5:Ret_60

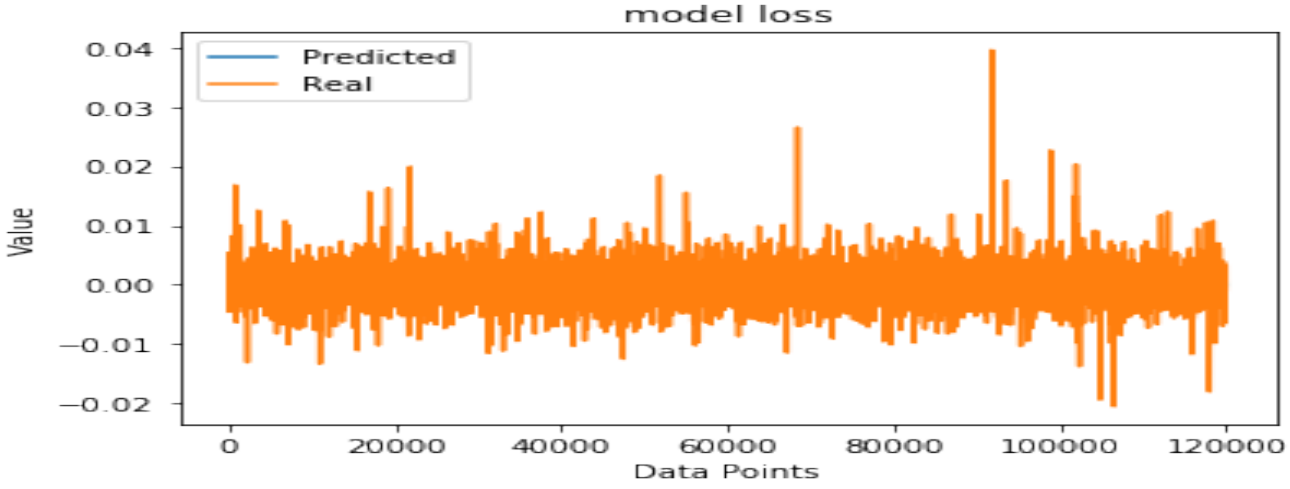
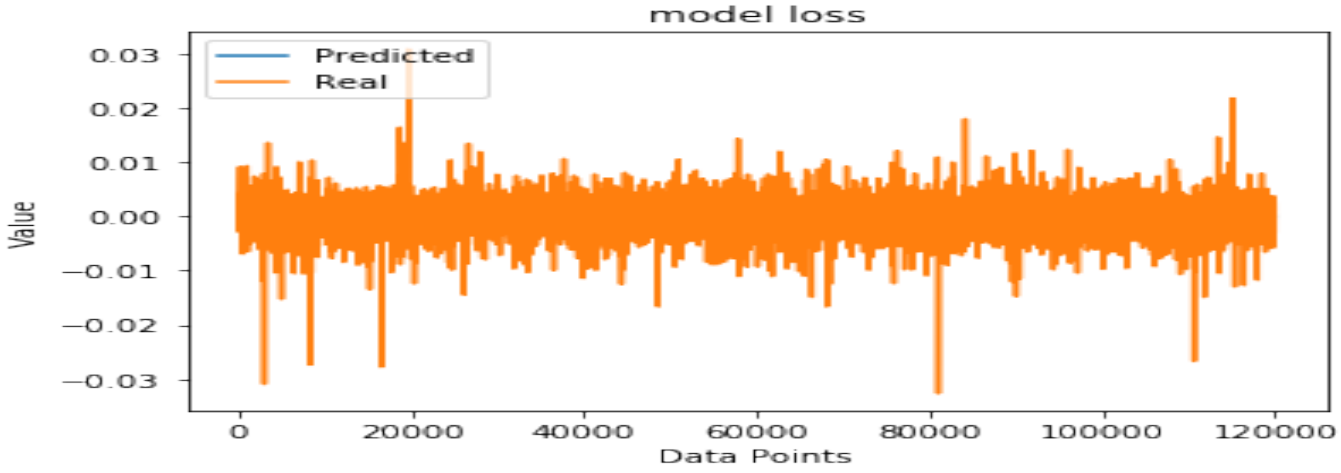


Fig 2.2.4



Further, the LSTM Model was also trained on **relative returns**. The data was processed to give the stock returns relative to the first day instead of the day previous to the respective data point. This approach also gave

an acceptable loss and MSE.

Fig 2.2.5

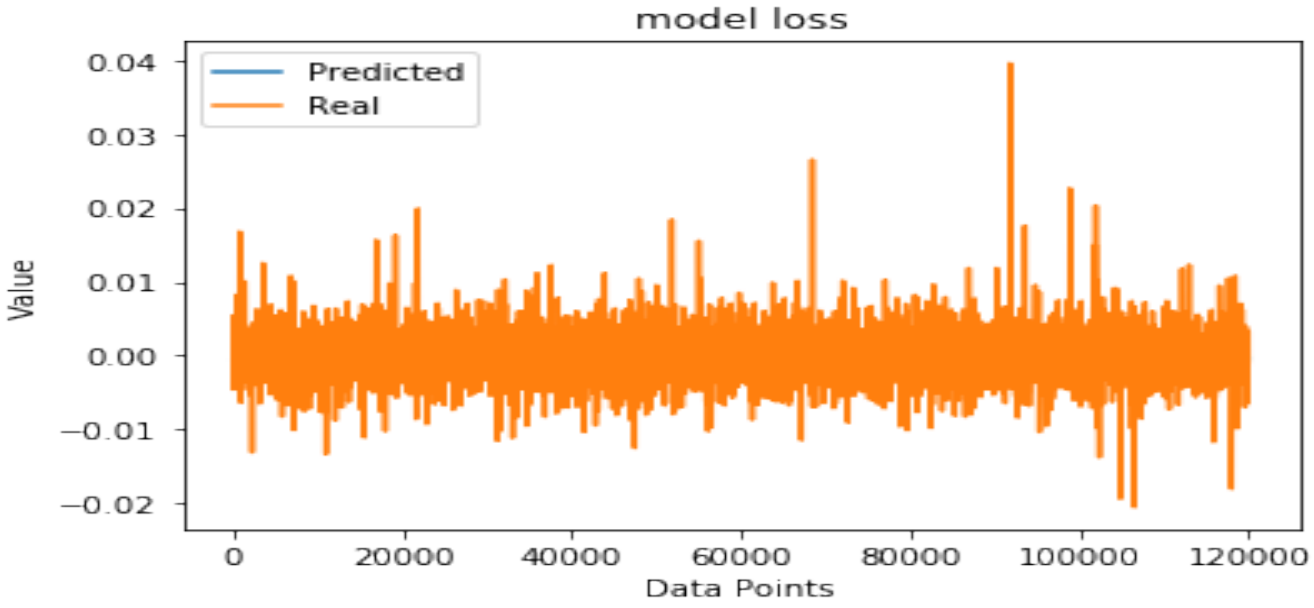
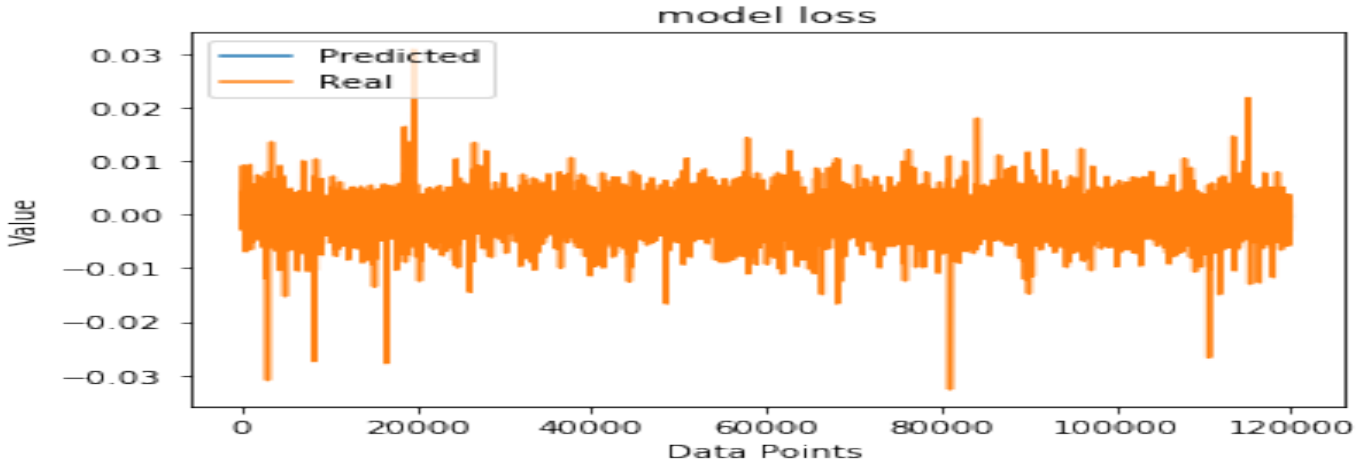


Fig 2.2.5



EXPERIMENTAL RESULTS

1) **Table 1.1**

S.No.	Model	No. of Epochs	Loss % (MSE)	Error (MAE)
1	Keras Regression(& Random Forest Model)	10	0	0
2	Sequential Model	10	8.99	5.64
3	LSTM	5	9.48	5.74
4	LSTM on relative returns	5	9.60	5.85

2) **Table 1.2**

S.No.	Model	No. of Epochs	Loss % (MSE)	Error (MAE)
1	Keras Regression(& Random Forest Model)	10	0	0
2	Sequential Model	10	1.49	6.60
3	LSTM	5	1.13	6.27
4	LSTM on relative returns	5	1.13	6.29

2.3 Methodology 2:

In this study, we have used variations of ANN to predict the stock price. But the efficiency of forecasting by ANNs depends upon the learning algorithm used to train the ANN. This paper compares three algorithms, i.e., Levenberg-Marquardt (LM), Scaled Conjugate Gradient and Bayesian Regularization. As shown in Fig. 1, neural net- works with 20 hidden layers and a delay of 50 data points are used. Thus, each prediction is made using the last 50 values.

Theory of Levenberg-Marquardt

The Levenberg-Marquardt algorithm was developed to approximate the second-order training speed to avoid the computation of the Hessian matrix, and used for solving a non-linear least The gradient is calculated in (2), which is first order derivative of total error function and used for updating weights in (4) square problem. The Hessian matrix can be estimated if the perform- ance function is in the form of a sum

of squares by

$$H=J^T J$$

Equation (1) is used to avoid heavy computation of hessian matrix as it can be calculated using Jacobian matrix.

$$G= J^T e$$

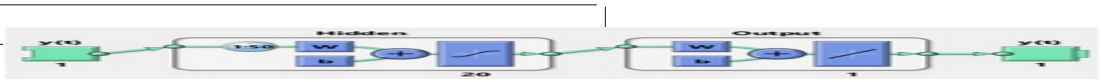


Fig. 2.3.1Neural Network Structure

Fig. 2 .3. 2 Flow chart of LM algorithm

where J is the Jacobian matrix and e is a vector of network errors. All the first derivatives which correspond to the network errors with respect to biases and weights contained in J. Keeping in mind the end goal to ensure that the approximated Hessian matric J^TJ is invertible, Levenberg– Marquardt calculation acquaints another approximation of Hessian matrix:

$$H=J^T J+\mu I$$

If the value of the scalar μ is zero, this algorithm will be similar to Newton's method which uses Hessian matrix approximation. If the scalar μ becomes large, this algo- rithm will be similar to gradient descent with small step size. But, Newton's method is much closer and faster near an error minimum. So, the primary object- ive is to shift toward Newton's method as fast as possible. Thus, decreasing μ after each successful step leads to trimming of the performance function. μ will increase only when there is any improvement in performance function at any tentative step as shown in Fig. 2. Therefore, at each iteration, the performance function is always reduced.
One of the significant merits of the LM approach is that it performs similarly to gradient search and Newton method for large values of μ and small values of μ respect- ively. The LM algorithm merges the best attributes of the steepest-descent algorithm

and the Gauss-Newton technique. Also, many of their limitations avoided. Specific- ally, this algorithm handles the problem of slow convergence efficiently (Hagan & Menhaj, 1994).

2.3.1SOFTWARE SPECIFICATION REQUIREMENTS:
Operating System:

- Windows 10.

Tools Used: Python version 3.7.

Chapter 3

3.1 Understanding the Problem Statement

We'll dive into the implementation part of this article soon, but first it's important to establish what we're aiming to solve. Broadly, stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

- 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.

As you might have guessed, our focus will be on the technical analysis part. We'll be using a dataset from Quandl (you can find historical data for various stocks here) and for this particular project, I have used the data for 'Tata Global Beverages'. Time to dive in!

Note: Here is the dataset I used for the code.

We will first load the dataset and define the target variable for the problem:

3.2 CODE:-

```
#import packages

import pandas as pd

import numpy as np



#to plot within notebook

import matplotlib.pyplot as plt

%matplotlib inline
```

```
#setting figure size

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 20,10



#for normalizing data

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))



#read the file

df = pd.read_csv('NSE-TATAGLOBAL(1).csv')



#print the head

df.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
1	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79
3	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590.0	3960.27
4	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749.0	3486.05

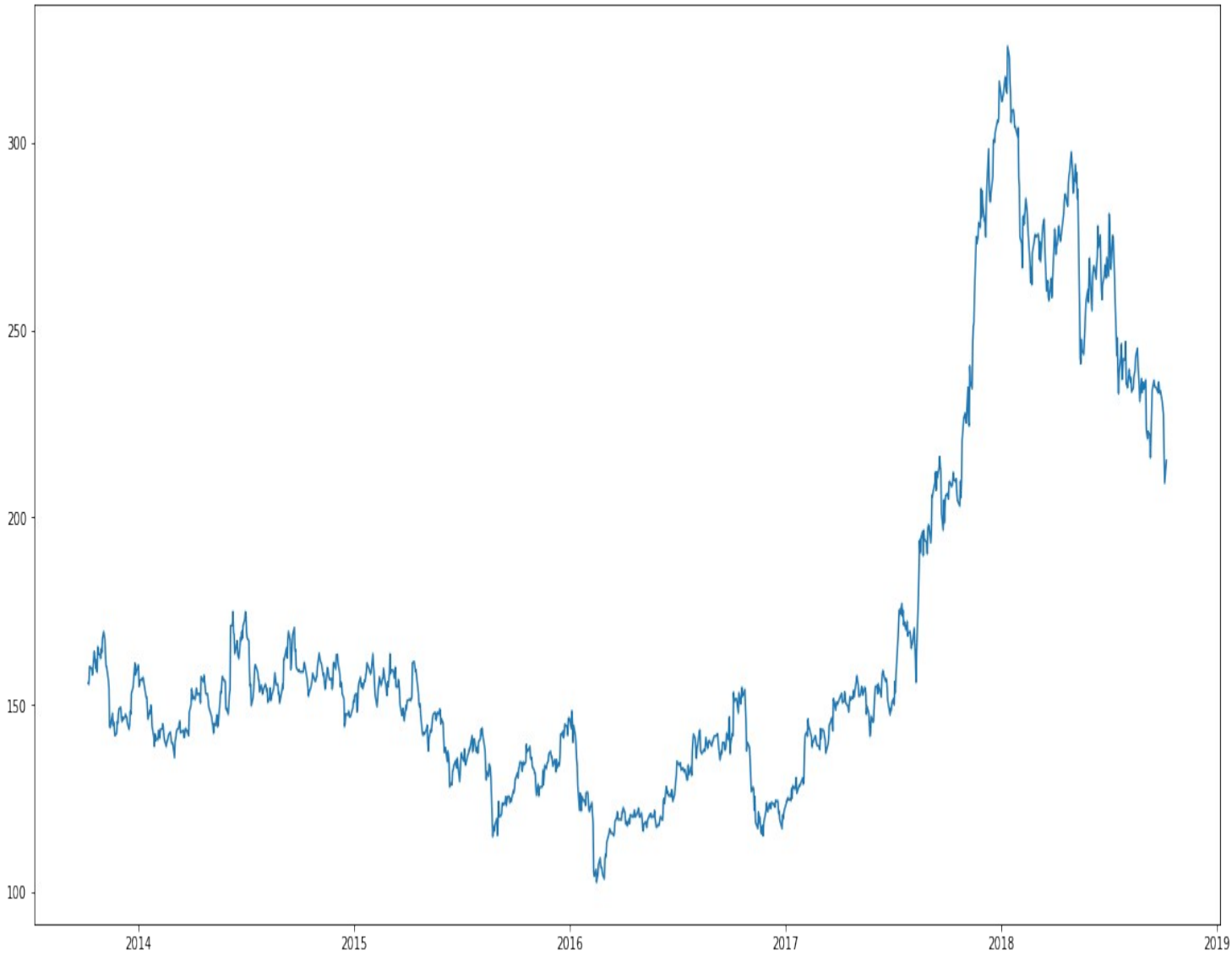
There are multiple variables in the dataset – date, open, high, low, last, close, total_trade_quantity, and turnover.

- 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.Notice the above table again, some date values are missing – 2/10/2018, 6/10/2018, 7/10/2018. Of these dates, 2nd is a national holiday while 6th and 7th fall on a weekend.

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. Let's plot the target variable to understand how it's shaping up in our data:

```
#setting index as date

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

df.index = df['Date']




#plot

plt.figure(figsize=(16,8))

plt.plot(df['Close'], label='Close Price history')
```



In the upcoming sections, we will explore these variables and use different techniques to predict the daily closing price of the stock.

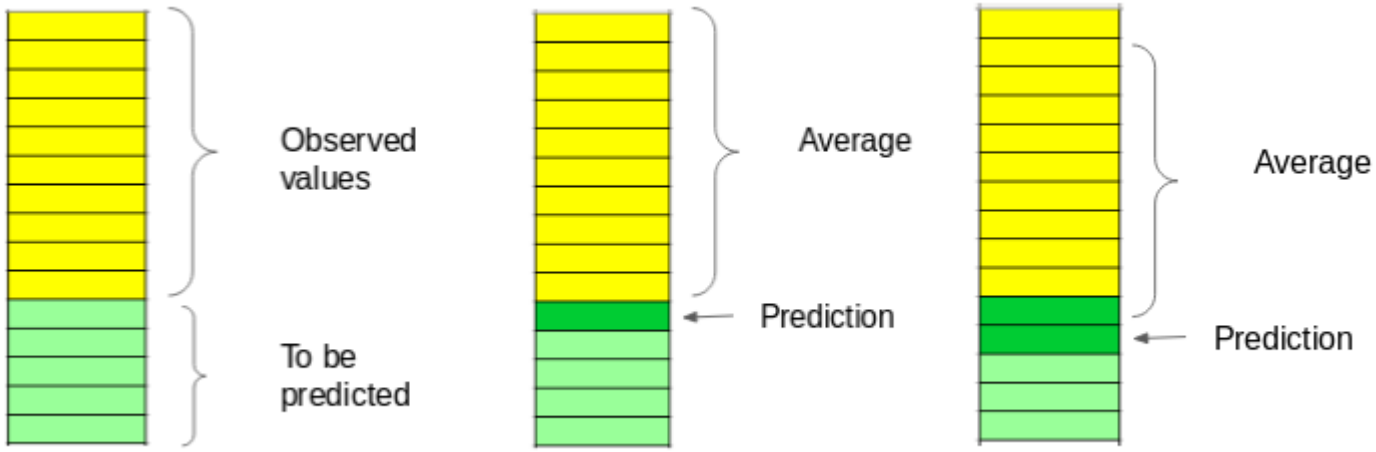
Moving Average

Introduction

'Average' is easily one of the most common things we use in our day-to-day lives. For instance, calculating the average marks to determine overall performance, or finding the average temperature of the past few days to get an idea about today's temperature – these all are routine tasks we do on a regular basis. So this is a good starting point to use on our dataset for making predictions.

The predicted closing price for each day will be the average of a set of previously observed values. Instead of using the simple average, we will be using the moving average technique which uses the latest set of values for each prediction. In other words, for each subsequent step, the predicted values are taken into consideration while

removing the oldest observed value from the set. Here is a simple figure that will help you understand this with more clarity.

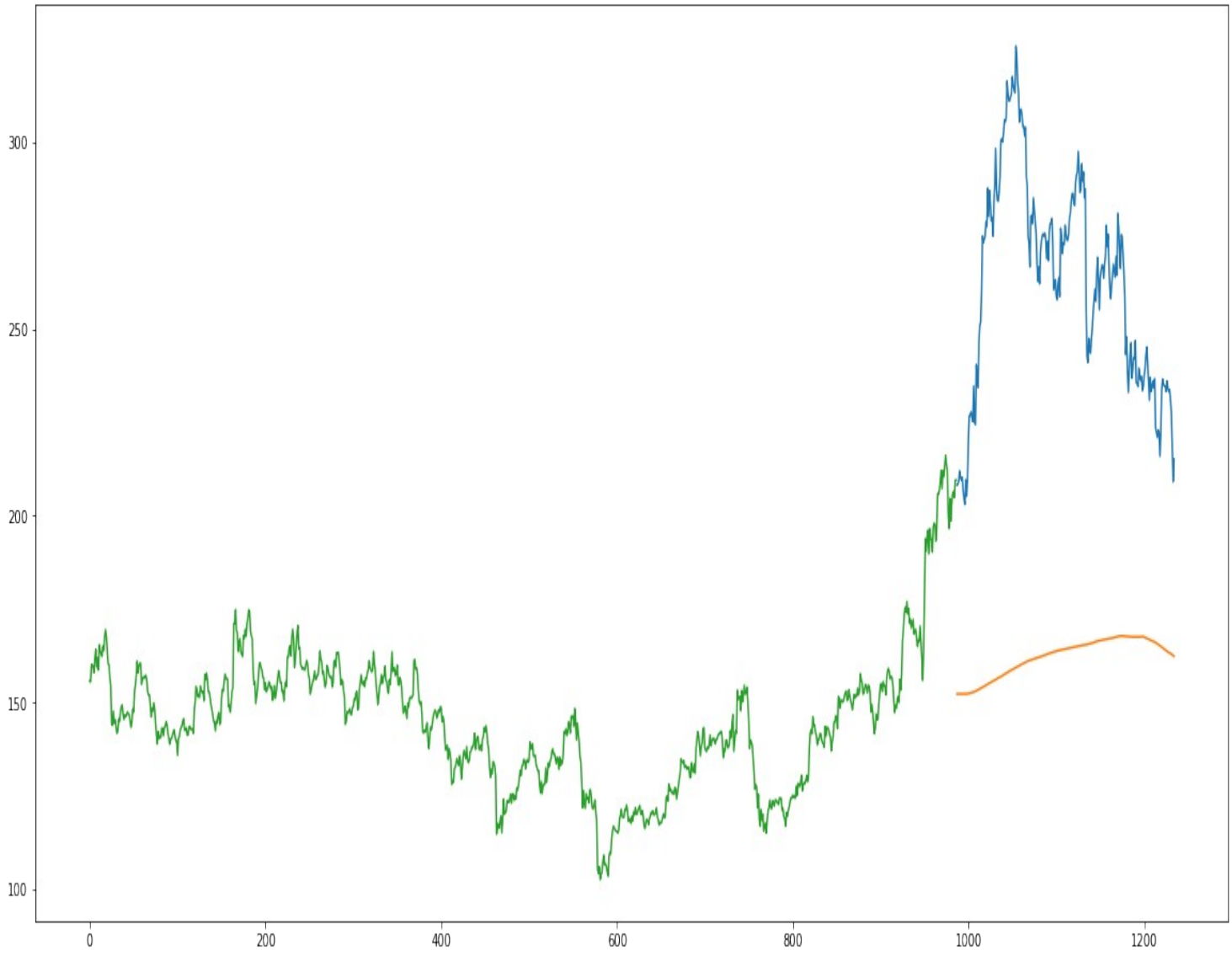


We will implement this technique on our dataset. The first step is to create a dataframe that contains only the *Date* and *Close* price columns, then split it into train and validation sets to verify our predictions.

Implementation

Just checking the RMSE does not help us in understanding how the model performed. Let's visualize this to get a more intuitive understanding. So here is a plot of the predicted values along with the actual values.

```
#plot

valid['Predictions'] = 0

valid['Predictions'] = preds

plt.plot(train['Close'])

plt.plot(valid[['Close', 'Predictions']])
```



Inference

The RMSE value is close to 105 but the results are not very promising (as you can gather from the plot). The predicted values are of the same range as the observed values in the train set (there is an increasing trend initially and then a slow decrease).

In the next section, we will look at two commonly used machine learning techniques – Linear Regression and kNN, and see how they perform on our stock market data.

Linear Regression

Introduction

The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable.

The equation for linear regression can be written as:

$$Y = \theta_1 X_1 + \theta_2 X_2 + \ldots \theta_n X_n$$

Here, x_1, x_2,….x_n represent the independent variables while the coefficients θ_1, θ_2, …. θ_n represent the weights. You can refer to the following article to study linear regression in more detail:

- A comprehensive beginners guide for Linear, Ridge and Lasso Regression.

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.

Implementation

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.

```
#setting index as date values

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

df.index = df['Date']



#sorting

data = df.sort_index(ascending=True, axis=0)



#creating a separate dataset

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]
```

```
#create features

from fastai.structured import  add_datepart

add_datepart(new_data, 'Date')

new_data.drop('Elapsed', axis=1, inplace=True)  #elapsed will be the time stamp
```

This creates features such as:

'Year', 'Month', 'Week', 'Day', 'Dayofweek', 'Dayofyear', 'Is_month_end', 'Is_month_start', 'Is_quarter_end', 'Is_quarter_start', 'Is_year_end', and 'Is_year_start'.

Note: I have used add_datepart from fastai library. If you do not have it installed, you can simply use the command ***pip install fastai****. Otherwise, you can create these feature using simple for loops in python. I have shown an example below.*

Apart from this, we can add our own set of features that we believe would be relevant for the predictions. For instance, my hypothesis is that the first and last days of the week could potentially affect the closing price of the stock far more than the other days. So I have created a feature that identifies whether a given day is Monday/Friday or Tuesday/Wednesday/Thursday. This can be done using the following lines of code:

```
new_data['mon_fri'] = 0
```

```
for i in range(0,len(new_data)):
    if (new_data['Dayofweek'][i] == 0 or new_data['Dayofweek'][i] == 4):
        new_data['mon_fri'][i] = 1
    else:
        new_data['mon_fri'][i] = 0
```

If the day of week is equal to 0 or 4, the column value will be 1, otherwise 0. Similarly, you can create multiple features. *If you have some ideas for features that can be helpful in predicting stock price, please share in the comment section.*

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)
```

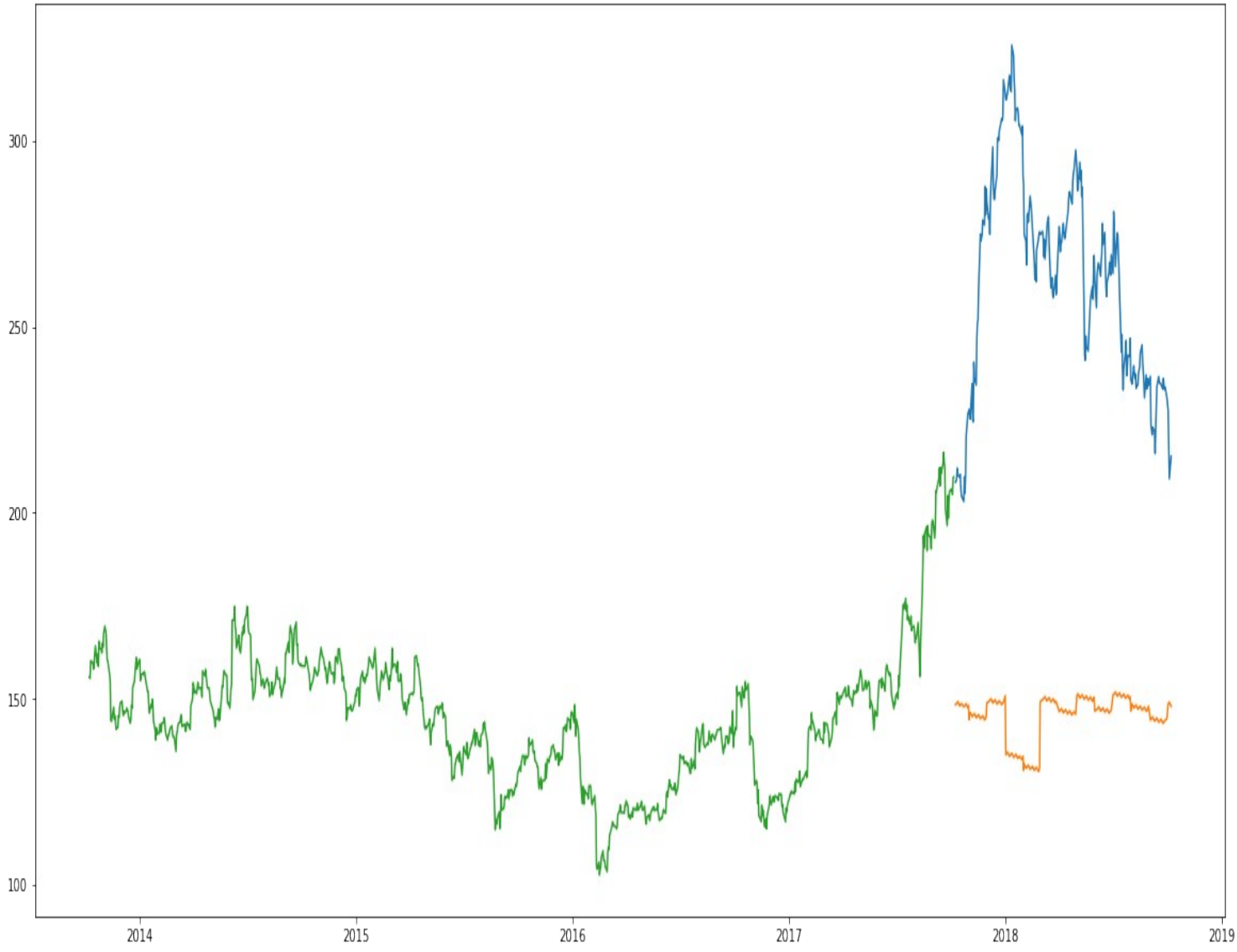
Results

```
#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

121.16291596523156
```

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:

```
#plot

valid['Predictions'] = 0

valid['Predictions'] = preds



valid.index = new_data[987:].index
```

```
train.index = new_data[:987].index



plt.plot(train['Close'])

plt.plot(valid[['Close', 'Predictions']])
```



Inference

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.

As seen from the plot above, for January 2016 and January 2017, there was a drop in the stock price. The model has predicted the same for January 2018. A linear regression technique can perform well for problems such as Big Mart sales where the independent features are useful for determining the target value.

3.3 k-Nearest Neighbours

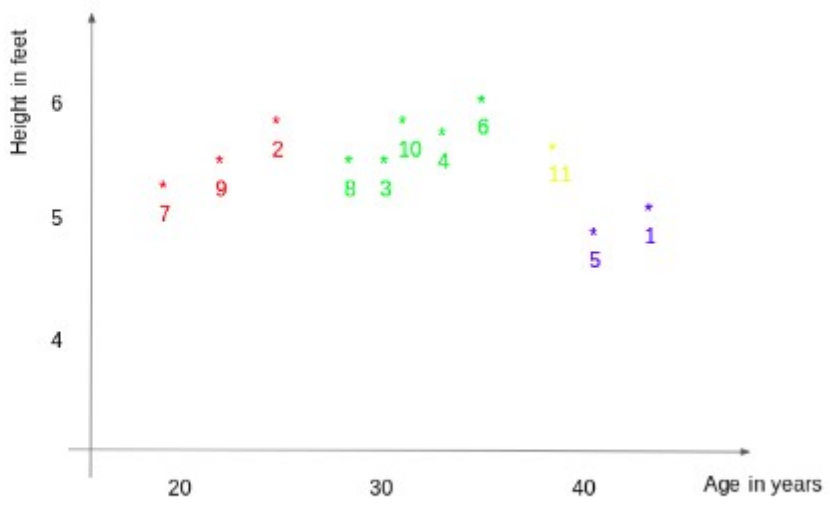
Introduction

Another interesting ML algorithm that one can use here is kNN (k nearest neighbours). Based on the independent variables, kNN finds the similarity between new data points and old data points. 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:

Table 3.3

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 it's neighbors. If we consider three neighbours (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

For a detailed understanding of kNN, you can refer to the following articles:

- Introduction to k-Nearest Neighbors: Simplified
- A Practical Introduction to K-Nearest Neighbors Algorithm for Regression

3.3 Implementation

```
#importing libraries

from sklearn import neighbors

from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
```

Using the same train and validation set from the last section:

```
#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)
```

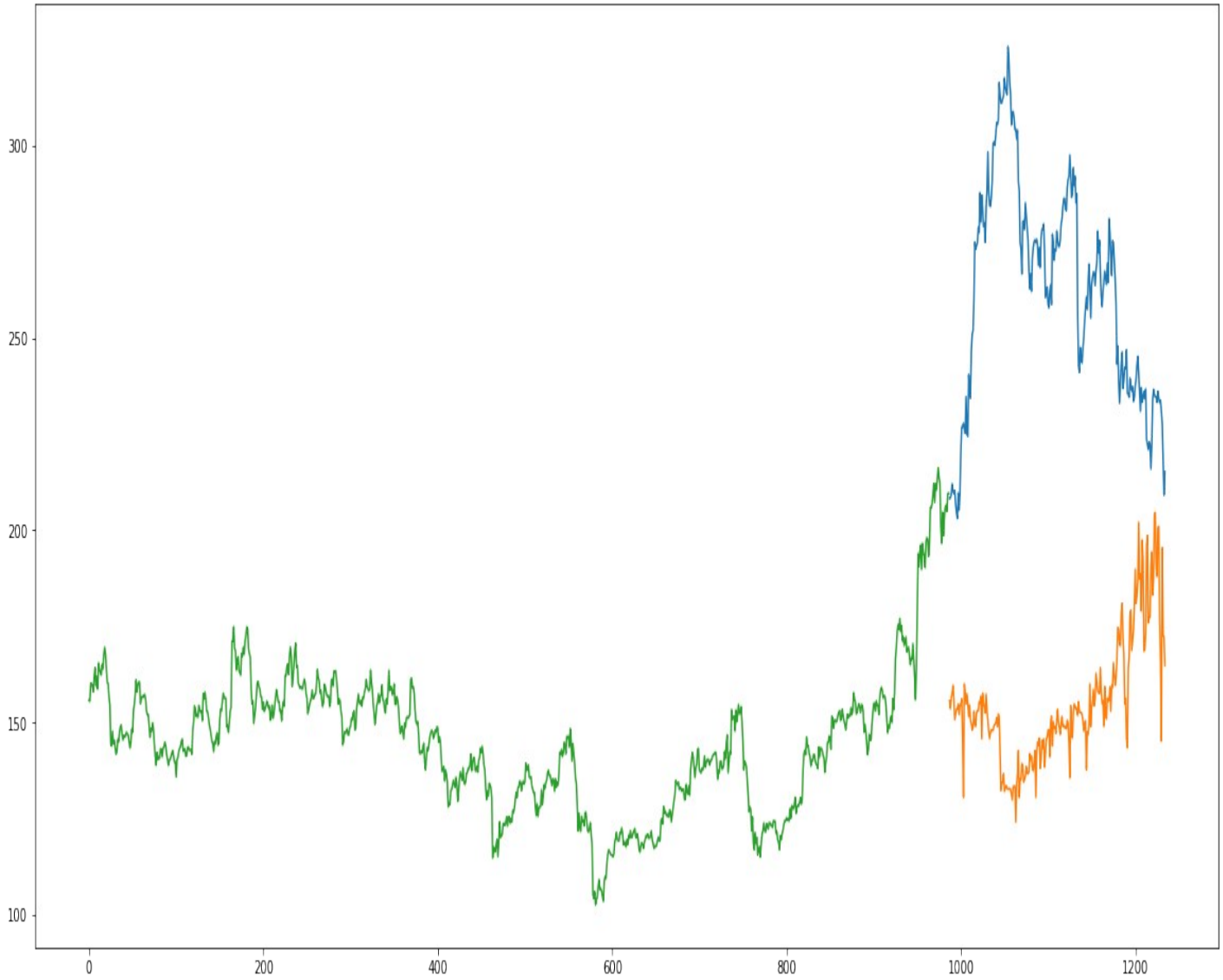
```
x_valid = pd.DataFrame(x_valid_scaled)



#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)
```

Results

```
#rmse

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

rms

115.17086550026721
```

There is not a huge difference in the RMSE value, but a plot for the predicted and actual values should provide a more clear understanding.

```
#plot

valid['Predictions'] = 0

valid['Predictions'] = preds

plt.plot(valid[['Close', 'Predictions']])

plt.plot(train['Close'])
```



Inference

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.

Let's go ahead and look at some time series forecasting techniques to find out how they perform when faced with this stock prices prediction challenge.

Auto ARIMA

Introduction

ARIMA is a very popular statistical method for time series forecasting. ARIMA models take into account the past values to predict the future values. There are three important parameters in ARIMA:

- p (past values used for forecasting the next value)
- q (past forecast errors used to predict the future values)
- d (order of differencing)

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. To read more about how auto ARIMA works, refer to this article:

- Build High Performance Time Series Models using Auto ARIMA

Implementation

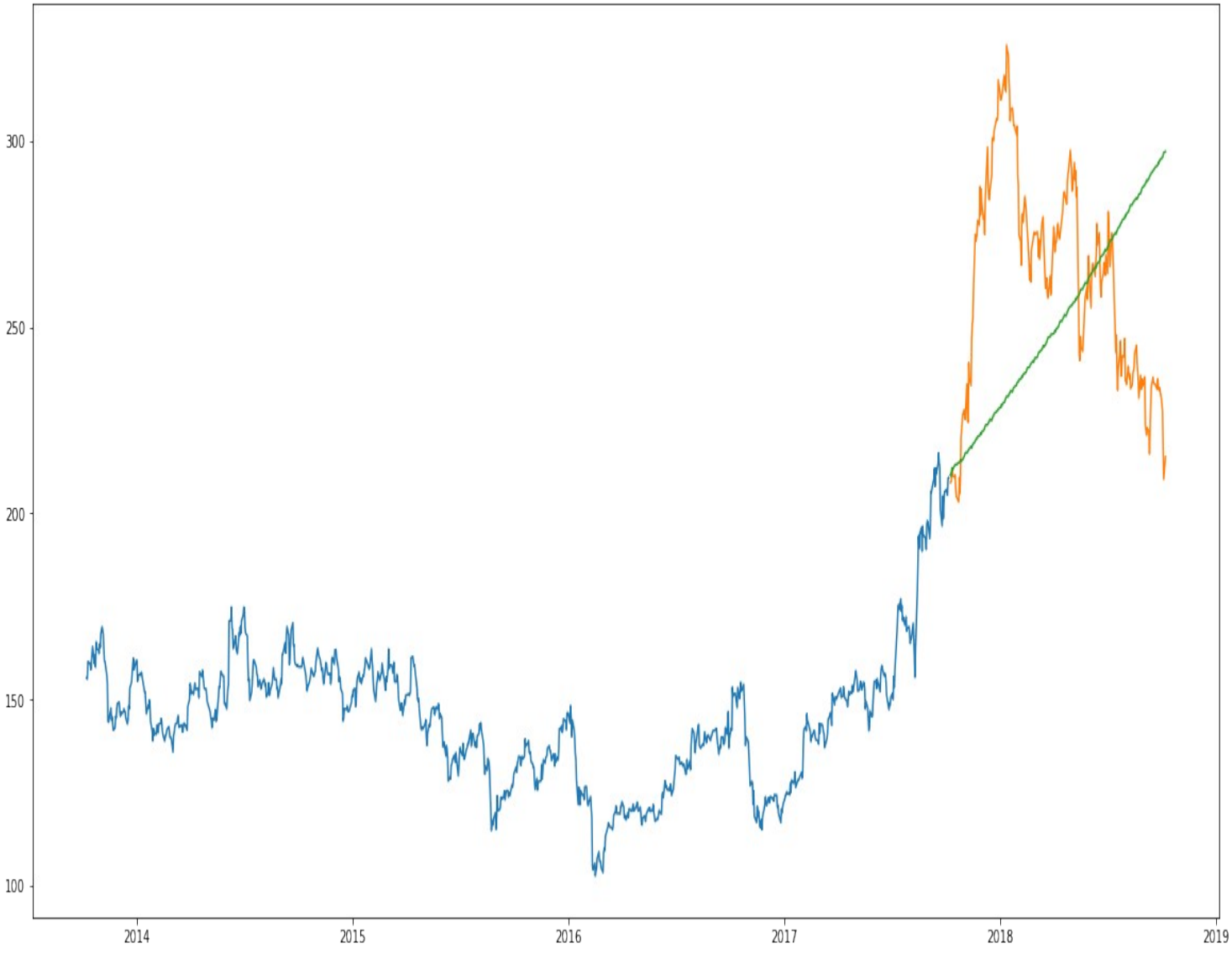
```
from pyramid.arima import auto_arima



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'])
```

Results

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

rms

44.954584993246954
```

```
#plot

plt.plot(train['Close'])

plt.plot(valid['Close'])

plt.plot(forecast['Prediction'])
```



Inference

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. In the next section, we will implement a time series model that takes both trend and seasonality of a series into account.

Prophet

Introduction

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 dataframe 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. Here is an interesting article that explains Prophet in a simple and intuitive manner:

- Generate Quick and Accurate Time Series Forecasts using Facebook's Prophet.

Implementation

```
#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]
```

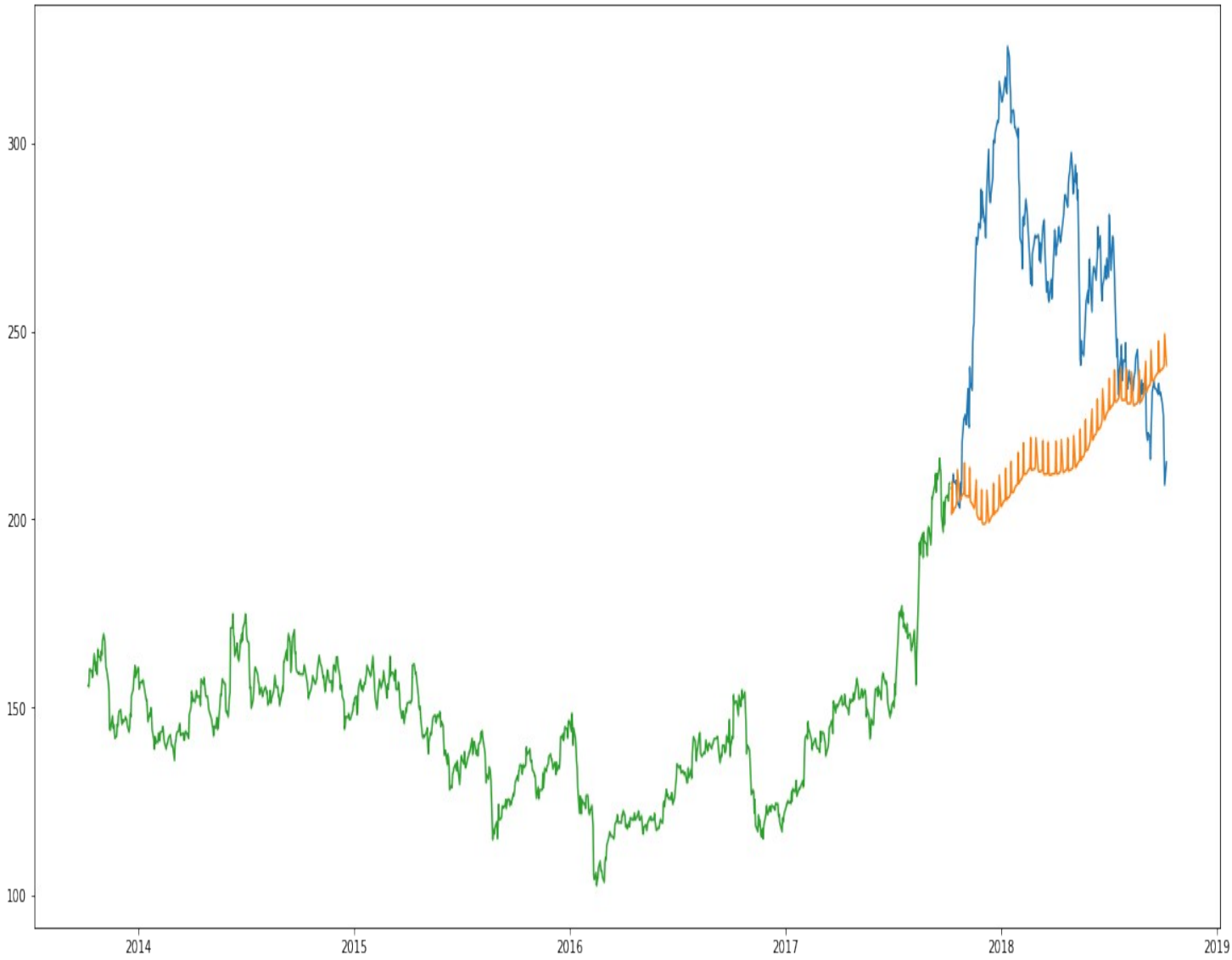
```
    new_data['Close'][i] = data['Close'][i]



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)
```

Chapter 4

Results

```
#rmse

forecast_valid = forecast['yhat'][987:]

rms=np.sqrt(np.mean(np.power((np.array(valid['y'])-np.array(forecast_valid)),2)))

rms

57.494461930575149

#plot

valid['Predictions'] = 0

valid['Predictions'] = forecast_valid.values



plt.plot(train['y'])

plt.plot(valid[['y', 'Predictions']])
```



4.1 Inference

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 it's 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.

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

4.2 Long Short Term Memory (LSTM)

Introduction

LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. 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

For a more detailed understanding of LSTM and its architecture, you can go through the below article:

- Introduction to Long Short Term Memory

For now, let us implement LSTM as a black box and check it's performance on our particular data.

4.2.1 Implementation

```
#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)):
```

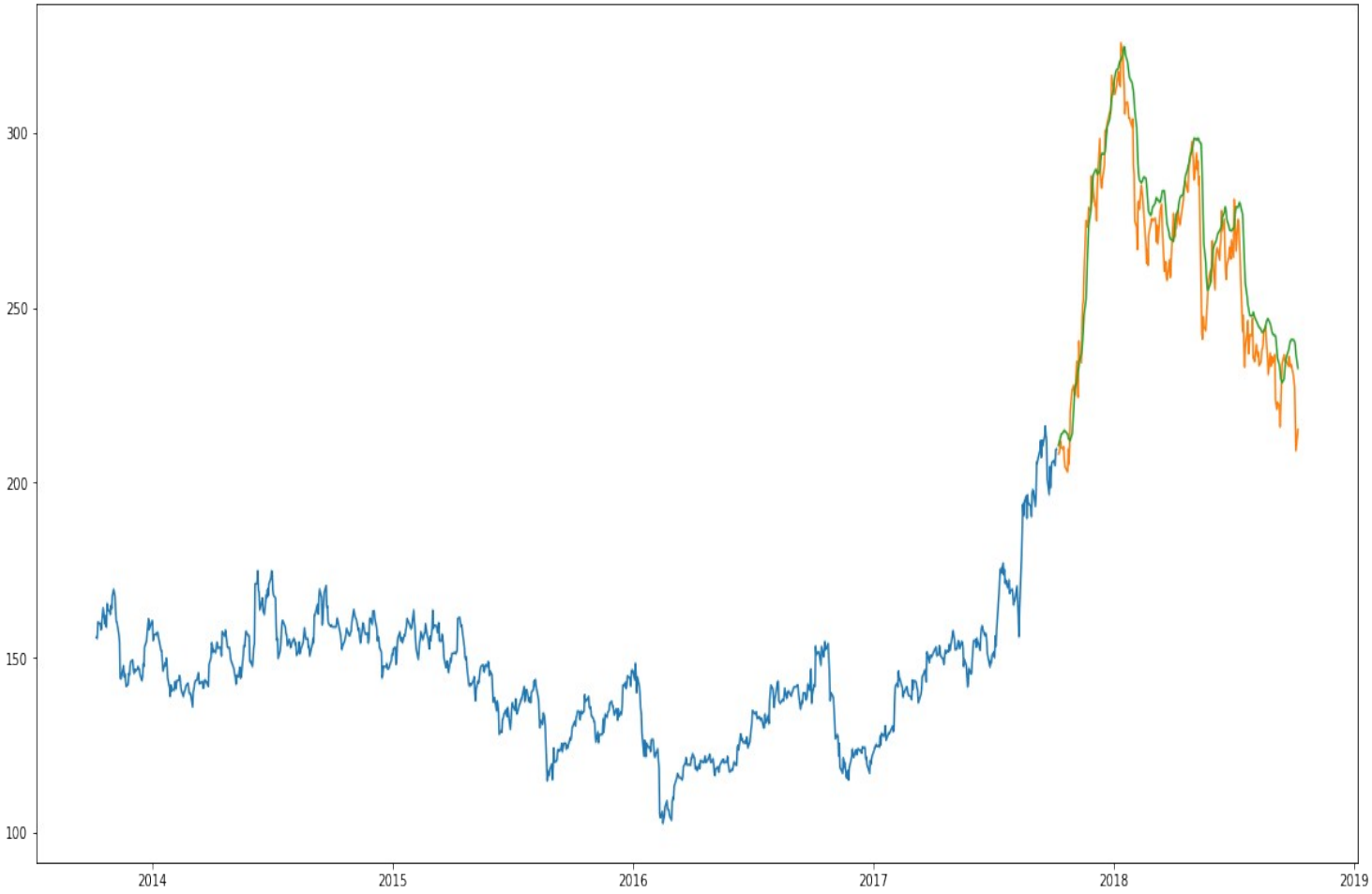
```
    new_data['Date'][i] = data['Date'][i]

    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)
```

Results

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

rms

11.772259608962642

#for plotting

train = new_data[:987]

valid = new_data[987:]

valid['Predictions'] = closing_price

plt.plot(train['Close'])

plt.plot(valid[['Close','Predictions']])
```



Inference

Wow! 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. But are the predictions from LSTM enough to identify whether the stock price will increase or decrease? Certainly not!

As I mentioned at the start of the article, stock price is affected by the news about the company and other factors like demonetization or merger/demerger of the companies. There are certain intangible factors as well which can often be impossible to predict beforehand.

Chapter 5

Conclusion/Future work:

we found that the most suitable algorithm for predicting the market price of a stock based on various data points from the historical data is the random forest algorithm. The algorithm will be a great asset for brokers and investors for investing money in the stock market since it is trained on a huge collection of historical data and has been chosen after being tested on a sample data. The project demonstrates the machine learning model to predict the stock value with more accuracy as compared to previously implemented machine learning models.

FUTURE ENHANCEMENT

Future scope of this project will involve adding more parameters and factors like the financial ratios, multiple instances, etc. The more the parameters are taken into account more will be the accuracy. The algorithms can also be applied for analyzing the contents of public comments and thus determine patterns/relationships between the customer and the corporate employee. The use of traditional algorithms and data mining techniques can also help predict the corporation "s performance .

References

- Ashish Sharma, Dinesh Bhuriya, Upendra Singh. "Survey of Stock Market Prediction Using Machine Learning Approach", ICECA 2017.
- Loke.K.S. “Impact Of Financial Ratios And Technical Analysis On Stock Price Prediction Using Random Forests”, IEEE,2017.
- Xi Zhang1, Siyu Qu1, Jieyun Huang1, Binxing Fang1, Philip Yu2, “Stock Market Prediction via Multi-Source Multiple Instance Learning.” IEEE2018.
- VivekKanade, BhausahebDevikar, SayaliPhadatare, PranaliMunde, ShubhangiSonone. “Stock Market Prediction: Using Historical Data Analysis”, IJARCSSE 2017.
- SachinSampatPatil, Prof. Kailash Patidar, Asst. Prof. Megha Jain, “A Survey on Stock Market Prediction Using SVM”, IJCTET2016.
- https://www.cs.princeton.edu/sites/default/files/uploads/Saahil_magde.pdf
- Hakob GRIGORYAN, “A Stock Market Prediction Method Based on Support Vector Machines (SVM) and Independent Component Analysis (ICA)”, DSJ 2016.
- RautSushrut Deepak, ShindeIshaUday, Dr. D. Malathi, “Machine Learning Approach In StockMarket
- Prediction”, IJPAM2017.

- Pei-Yuan Zhou , Keith C.C. Chan, Member, IEEE, and Carol XiaojuanOu, “Corporate Communication Network and Stock Price Movements: Insights From Data Mining”, IEEE2018.