

Area Of Online Internship	Machine learning
Intern Name	Ravish Kumar
Name Of Institution	INDIAN INSTITUTE OF TECHNOLOGY, INDORE
Faculty Mentor Name	Prof. Vimal Bhatia
Duration	2 MONTHS (01/01/2022 TO 01/03/2022)
Date Of Submission	27/02/2022

Table Of Contents

- **Random forest algorithm**
- **Linear Regression**
- **KNN (k- Nearest Neighbours)**
- **LSTM (Long short-term memory)**

Random forest algorithm

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

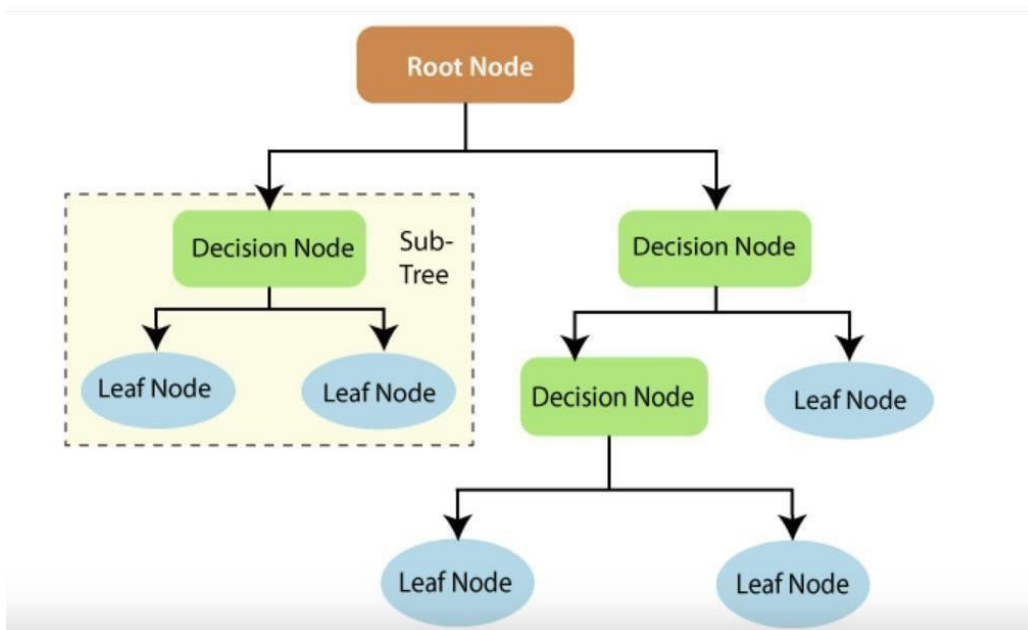


Figure 1. Random forest algorithm

Features of a Random Forest Algorithm

- It's more accurate than the decision tree algorithm.
- It provides an effective way of handling missing data.
- It can produce a reasonable prediction without hyper-parameter tuning.
- It solves the issue of overfitting in decision trees.

- In every random forest tree, a subset of features is selected randomly at the node's splitting point.

Linear Regression

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

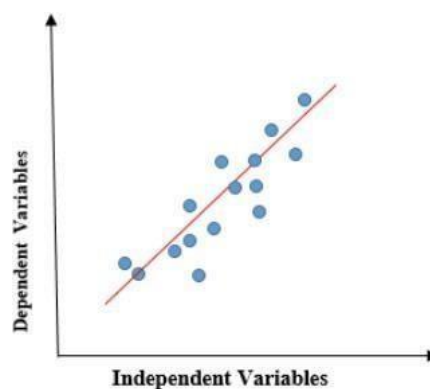


Figure 2. Linear Regression

KNN(k- Nearest Neighbours)

K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint.

The algorithm's learning is:

1. Instance-based learning: Here we do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data.
2. Lazy Learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.
3. Non -Parametric: In KNN, there is no predefined form of the mapping function.

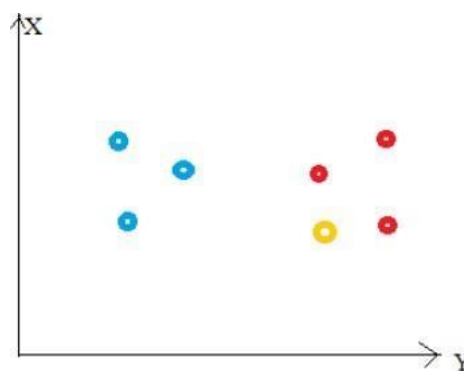


Figure 3. KNN(k- Nearest Neighbours)

LSTM (Long short-term memory)

Long Short-Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. It tackled the problem of longterm dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period. It is used for processing, predicting, and classifying based on time-series data.

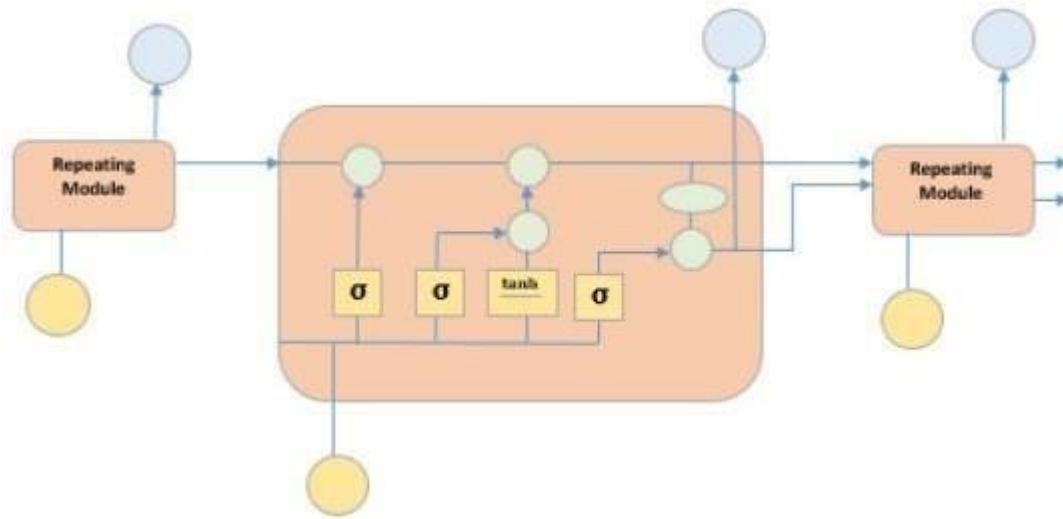


Figure 4. LSTM (Long short-term memory)

Problem

Predict the stock market price of the next few days using previous stock market data (equity or indices) using **Machine learning** or **deep learning**.

Solution

1. News headlines as Data for prediction:

Natural language processing (NLP)

NLP is a field in machine learning with the ability of a computer to understand, analyze, manipulate, and potentially generate human language.

There are various kinds of news articles and based on that the stock price fluctuates. We will analyze the news heading using sentiment analysis using NLP and then we will predict the stock will increase or decrease. It is all about stock sentiment analysis.

Implementation

Let's first import required libraries,

jupyter stock prediction using news Last Checkpoint: 3 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

```
In [ ]: import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
In [ ]: df = pd.read_csv('F:Stock-Sentiment-Analysis-master/Stock News Dataset.csv', encoding = "ISO-8859-1")
```

In [10]: df.head(5)

Out[10]:

	Date	Label	Top1	Top2	Top3	Top4	Top5	Top6	Top7	Top8	...	Top16	Top17	Top18	Top19
0	2000-01-03	0	A 'hindrance to operations': extracts from the...	Scorecard	Hughes' instant hit buoys Blues	Jack gets his skates on at ice-cold Alex	Chaos as Maracana builds up for United	Depleted Leicester prevail as Elliott spoils E...	Hungry Spurs sense rich pickings	Gunners so wide of an easy target	...	Flintoff injury piles on woe for England	Hunters threaten Jospin with new battle of the...	Kohl's successor drawn into scandal	Ti differen betwei men al wom
1	2000-01-04	0	Scorecard	The best lake scene	Leader: German sleaze inquiry	Cheerio, boyo	The main recommendations	Has Cubie killed fees?	Has Cubie killed fees?	Has Cubie killed fees?	...	On the critical list	The timing of their lives	Dear doctor	Irish co halts IF mar extraditi to Noi
2	2000-01-05	0	Coventry caught on counter by Flo	United's rivals on the road to Rio	Thatcher issues defence before trial by video	Police help Smith lay down the law at Everton	Tale of Trautmann bears two more retellings	England on the rack	Pakistan retaliate with call for video of Walsh	Cullinan continues his Cape monopoly	...	South Melbourne (Australia)	Necaxa (Mexico)	Real Madrid (Spain)	Ra Casablan (Morocc
3	2000-01-06	1	Pilgrim knows how to progress	Thatcher facing ban	McIlroy calls for Irish fighting spirit	Leicester bin stadium blueprint	United braced for Mexican wave	Auntie back in fashion, even if the dress look...	Shoalb appeal goes to the top	Hussain hurt by 'shambles' but lays blame on e...	...	Putin admits Yeltsin quit to give him a head s...	BBC worst hit as digital TV begins to bite	How much can you pay for...	Christm: glitch
4	2000-01-07	1	Hitches and Horlocks	Beckham off but United survive	Breast cancer screening	Alan Parker	Guardian readers: are you all whingers?	Hollywood Beyond	Ashes and diamonds	Whingers - a formidable minority	...	Most everywhere: UDs	Most wanted: Chloe lunettes	Return of the cane 'completely off the agenda'	Fro Slee Hollow Greenel

5 rows x 27 columns

Let's divide our dataset into the training and testing parts.

```
In [ ]: train = df[df['Date'] < '20150101']
test = df[df['Date'] > '20141231']
```

do some feature engineering on our dataset.


```
In [ ]: # Removing special characters
data=train.iloc[:,2:27]
data.replace("[^a-zA-Z]", " ", regex=True, inplace=True)
# Renaming column names for better understanding and ease of access
list1= [i for i in range(25)]
new_Index=[str(i) for i in list1]
data.columns= new_Index
data.head(5)
```

	0	1	2	3	4	5	6	7	8	9	...	15	16	17	
0	A hindrance to operations extracts from the...	Scorecard	Hughes instant hit buoys blues	Jack gets his skates on at ice cold alex	Chaos as Maracana builds up for United	Depleted Leicester prevail as Elliott spoils E...	Hungry Spurs sense rich pickings	Gunners so wide of an easy target	Derby raise a glass to Strupar s debut double	Southgate strikes Leeds pay the penalty	...	Flintoff injury piles on woe for England	Hunters threaten Jospin with new battle of the...	Kohl s successor drawn into scandal	
1	Scorecard	The best lake scene	Leader German sleaze inquiry	Cheerio boyo	The main recommendations	Has Cubie killed fees	Has Cubie killed fees	Has Cubie killed fees	Hopkins furious at Foster s lack of Hannibal...	Has Cubie killed fees	...	On the critical list	The timing of their lives	Dear doctor	
2	Coventry caught on counter by Flo	United s rivals on the road to Rio	Thatcher issues defence before trial by video	Police help Smith lay down the law at Everton	Tale of Trautmann bears two more retellings	England on the rack	Pakistan retaliate with call for video of Walsh	Cullinan continues his Cape monopoly	McGrath puts India out of their misery	Blair Witch bandwagon rolls on	...	South Melbourne Australia	Necaxa Mexico	Real Madrid Spain	C
3	Pilgrim knows how to progress	Thatcher facing ban	McIlroy calls for Irish fighting spirit	Leicester bin stadium blueprint	United braced for Mexican wave	Auntie back in fashion even if the dress look...	Shoaib appeal goes to the top	Hussain hurt by shambles but lays blame on e...	England s decade of disasters	Revenge is sweet for jubilant Cronje	...	Putin admits Yeltsin quit to give him a head s...	BBC worst hit as digital TV begins to bite	How much can you pay for	
4	Hiltches and Horlocks	Beckham off but United survive	Breast cancer screening	Alan Parker	Guardian readers are you all whingers	Hollywood Beyond	Ashes and diamonds	Whingers a formidable minority	Alan Parker part two	Thuggery Toxins and Ties	...	Most everywhere UDis	Most wanted Chloe lunettes	Return of the cane completely off the agenda	G

convert all these characters into smaller characters,

```
In [ ]: # Converting headlines to Lower case
for index in new_Index:
    data[index] = data[index].str.lower()
data.head(1)
```

combine all news headlines based on the index,

```
In [ ]: headlines = []
for row in range(0,len(data.index)):
    headlines.append(' '.join(str(x) for x in data.iloc[row,0:25]))
```

```
In [15]: headlines[0]
```

```
Out[15]: 'a hindrance to operations extracts from the leaked reports scorecard hughes instant hit buoys blues jack gets his skates on at ice cold alex chaos as maracana builds up for united depleted leicester prevail as elliot spoils everton s party hungry s purs sense rich pickings gunners so wide of an easy target derby raise a glass to strupar s debut double southgate strikes lee ds pay the penalty hammers hand robson a youthful lesson saints party like it s wear wolves have turned into lambs stump m ke catches testy gough s taunt langer escapes to hit flintoff injury piles on woe for england hunters threaten jospin with new battle of the somme kohl s successor drawn into scandal the difference between men and women sara denver nurse turned solidior diana s landmine crusade put tories in a panic yeltsin s resignation caught opposition flat footed russian roulette sold out recovering a title'
```

apply Count Vectorizer and Random Forest Classifier;

```
In [ ]: > ## implement BAG OF WORDS
countvector=CountVectorizer(ngram_range=(2,2))
traindataset=countvector.fit_transform(headlines)
## implement RandomForest Classifier
randomclassifier=RandomForestClassifier(n_estimators=200,criterion='entropy')
randomclassifier.fit(traindataset,train['Label'])
```

predict for the testing dataset;

```
In [ ]: > ## Predict for the Test Dataset
test_transform= []
for row in range(0,len(test.index)):
    test_transform.append(' '.join(str(x) for x in test.iloc[row,2:27]))
test_dataset = countvector.transform(test_transform)
predictions = randomclassifier.predict(test_dataset)
```

Finally, let's check the accuracy;

```
In [22]: matrix=confusion_matrix(test['Label'],predictions)
print(matrix)
score=accuracy_score(test['Label'],predictions)
print(score)
report=classification_report(test['Label'],predictions)
print(report)
```

```
[[139  47]
 [  9 183]]
0.8518518518518519
              precision    recall  f1-score   support

         0       0.94      0.75      0.83        186
         1       0.80      0.95      0.87        192

 accuracy          0.85          378
  macro avg       0.87      0.85      0.85          378
 weighted avg     0.87      0.85      0.85          378
```

Now suppose we have to predict it for tomorrow, take the top 25 news headlines apply all the transformation methods, and finally give it to our model, our model will

say whether your 0 or 1 means stock price will increase or not. This is how we can do stock sentiment analysis using news headlines. **2. Use Equity data for prediction:**

Introduction

Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. psychological, rational and irrational behavior, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.

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.

We will work with historical data about the stock prices of a publicly listed company. We will implement a mix of machine learning algorithms to predict the future stock price of this company, starting with simple algorithms like averaging and linear regression, and then moving on to advanced techniques like Auto ARIMA and LSTM.

Stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

- Fundamental Analysis involves analyzing the company's future profitability based on 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.

Implementation

```
In [ ]: import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
In [ ]: df = pd.read_csv('NSE-TATAGLOBAL(1).csv')
```

```
In [ ]: 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

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

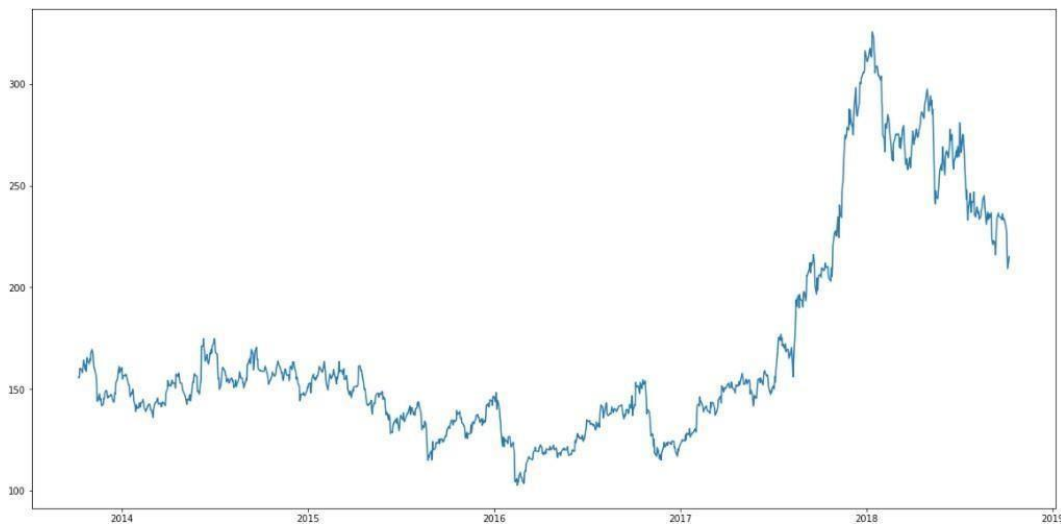


Figure 5. True Adjusted Close Value

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 regularly. 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.

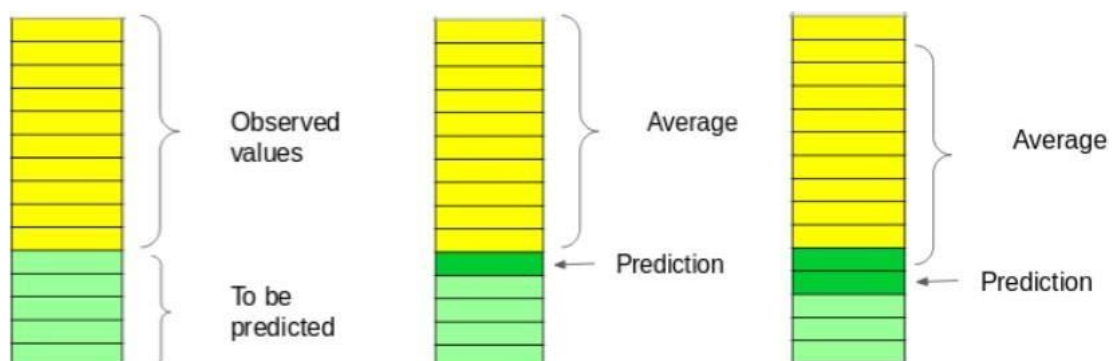
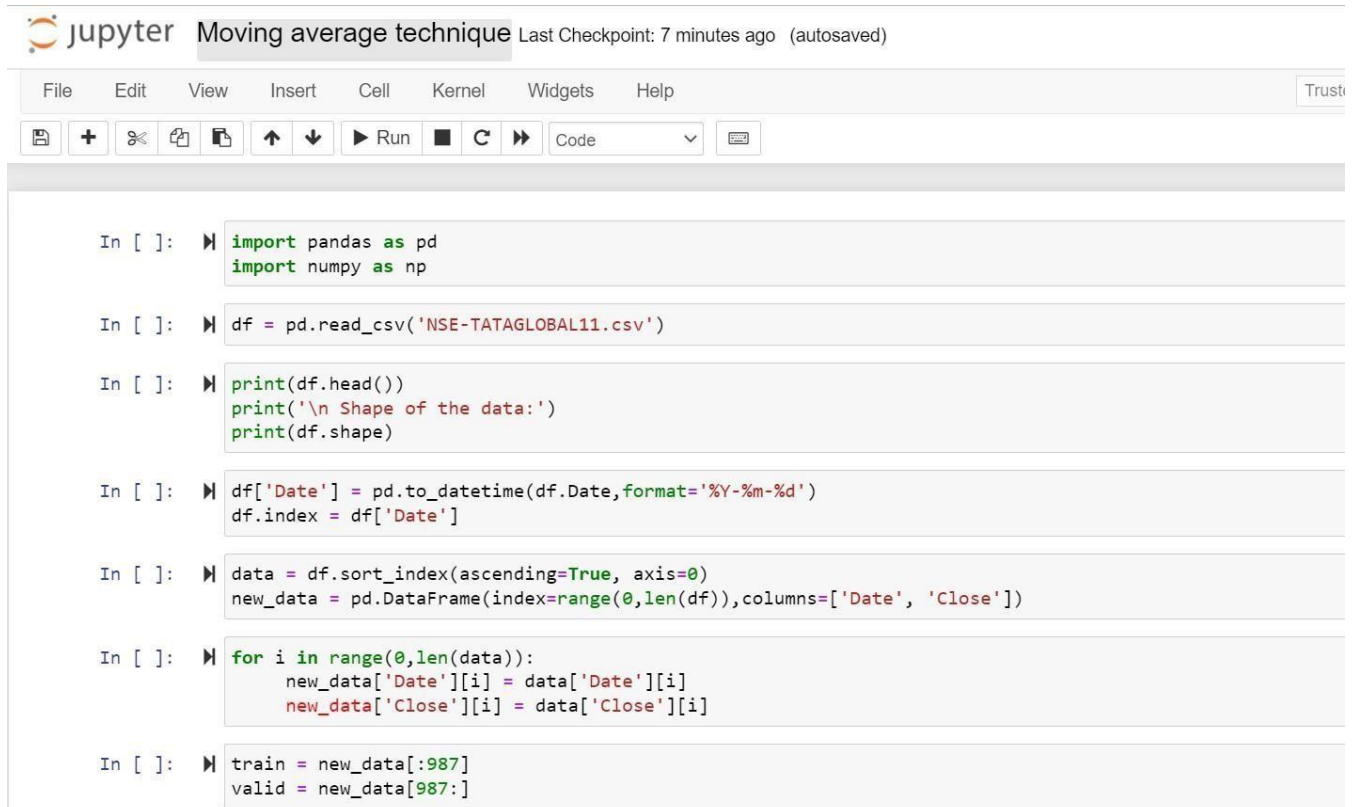


Figure 6. Moving Average

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



The image shows a Jupyter Notebook interface with the title "Moving average technique" and a status bar indicating "Last Checkpoint: 7 minutes ago (autosaved)". The notebook has a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Below the menu bar is a toolbar with icons for saving, adding, deleting, and running code cells. The main area contains six code cells, each starting with "In []:" and followed by a prompt character "▶".

```
In [ ]: ▶ import pandas as pd
import numpy as np

In [ ]: ▶ df = pd.read_csv('NSE-TATAGLOBAL11.csv')

In [ ]: ▶ print(df.head())
print('\n Shape of the data:')
print(df.shape)

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

In [ ]: ▶ data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

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

In [ ]: ▶ train = new_data[:987]
valid = new_data[987:]
```

```
In [ ]: train = new_data[:987]
        valid = new_data[987:]

In [ ]: print('\n Shape of training set:')
        print(train.shape)

In [ ]: print('\n Shape of validation set:')
        print(valid.shape)

In [ ]: preds = []
        for i in range(0, valid.shape[0]):
            a = train['Close'][len(train)-248+i:].sum() + sum(preds)
            b = a/248
            preds.append(b)

In [ ]: rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-preds),2)))
        print('\n RMSE value on validation set:')
        print(rms)

In [ ]: valid['Predictions'] = 0
        valid['Predictions'] = preds
        plt.plot(train['Close'])
        plt.plot(valid[['Close', 'Predictions']])
```

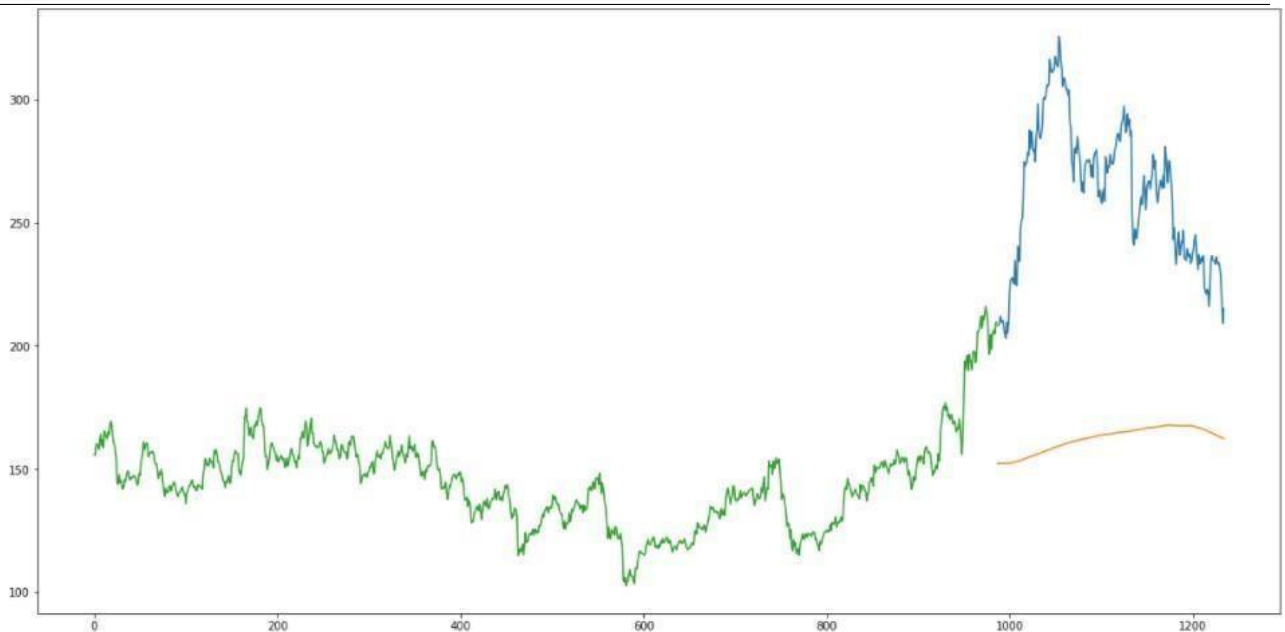


Figure 7. Prediction by Moving Average (Graph between True and average value)

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).

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:

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

Here, X_1, X_2, \dots, X_n represent the independent variables while the coefficients $\theta_1, \theta_2, \dots, \theta_n$ represent the weights.

Implementation

 **jupyter** Linear Regression technique Last Checkpoint: 5 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

        Run    Code 

```
In [ ]: import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

In [ ]: df = pd.read_csv('NSE-TATAGLOBAL(1).csv')
df.head()

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

In [ ]: data = df.sort_index(ascending=True, axis=0)

In [ ]: 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]
```


File Edit View Insert Cell Kernel Widgets Help



```
In [ ]: from fastai.structured import add_datepart
add_datepart(new_data, 'Date')
new_data.drop('Elapsed', axis=1, inplace=True)
```

```
In [ ]: 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
```

```
In [ ]: 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']
```

```
In [ ]: from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)
```

```
In [ ]: preds = model.predict(x_valid)
rms = np.sqrt(np.mean(np.power((np.array(y_valid) - np.array(preds)), 2)))
rms
```

File Edit View Insert Cell Kernel Widgets Help



```
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']
```

```
In [ ]: from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)
```

```
In [ ]: preds = model.predict(x_valid)
rms = np.sqrt(np.mean(np.power((np.array(y_valid) - np.array(preds)), 2)))
rms
```

```
In [ ]: 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']])
```

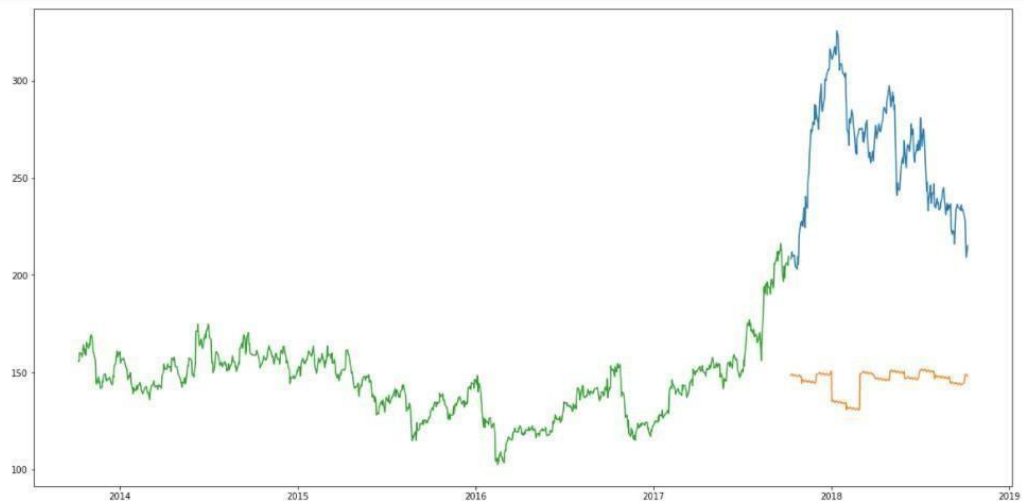


Figure 8. Prediction by Linear Regression (Graph between True and regression value)

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 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 where the independent features are useful for determining the target value.

k-Nearest Neighbours

Introduction

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

Implementation

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [ ]: import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
In [ ]: df = pd.read_csv('NSE-TATAGLOBAL(1).csv')
df.head()
```

```
In [ ]: from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
In [ ]: 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)
```

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x_valid)
x_valid = pd.DataFrame(x_valid_scaled)
```

```
In [ ]: params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)
```

```
In [ ]: model.fit(x_train,y_train)
preds = model.predict(x_valid)
```

```
In [ ]: rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

```
In [ ]: valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(valid[['Close', 'Predictions']])
plt.plot(train['Close'])
```

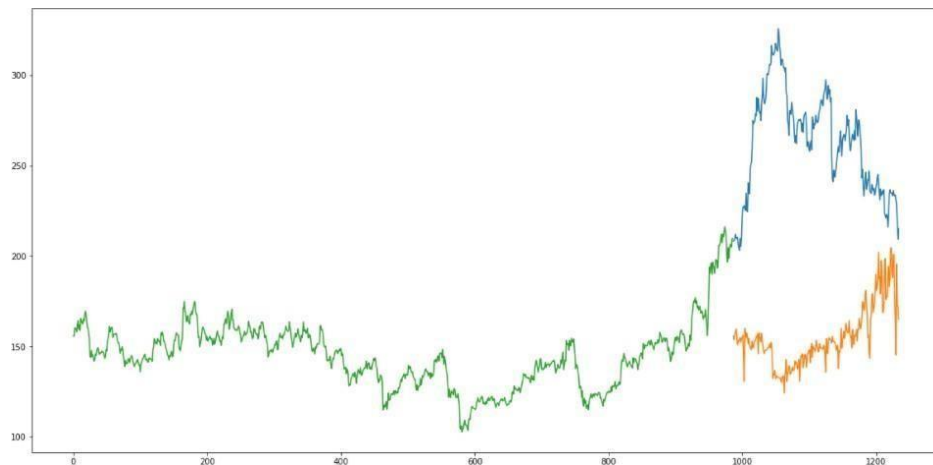


Figure 9. Prediction by KNN (Graph between True and KNN value)

The RMSE (115.17086550026721) 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.

Implementation

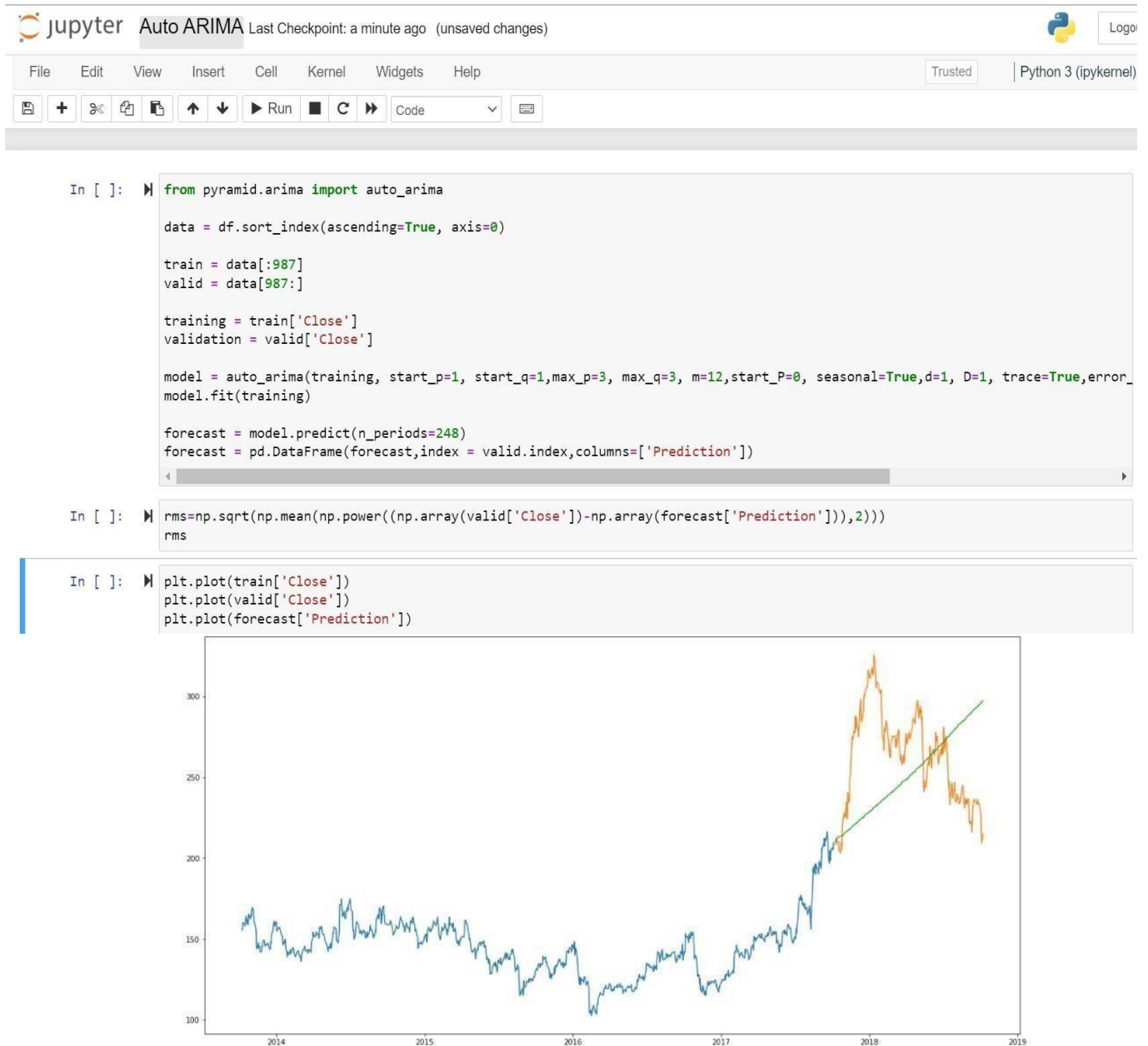


Figure 10. Prediction by Auto ARIMA (Graph between True and ARIMA value)

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.

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 that LSTM can store past important information 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

Implementation

 jupyter LSTM technique Last Checkpoint: 4 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

       Run    Code 

```
In [ ]: ▶ from sklearn.preprocessing import MinMaxScaler
        from keras.models import Sequential
        from keras.layers import Dense, Dropout, LSTM
```

```
In [ ]: ▶ 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]
```

```
In [ ]: ▶ new_data.index = new_data.Date
        new_data.drop('Date', axis=1, inplace=True)
```

```
In [ ]: ▶ dataset = new_data.values

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


File Edit View Insert Cell Kernel Widgets Help



```
In [ ]: 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))

In [ ]: 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))

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

In [ ]: inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1, 1)
inputs = scaler.transform(inputs)
```

File Edit View Insert Cell Kernel Widgets Help



```
In [ ]: 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)

In [ ]: rms = np.sqrt(np.mean(np.power((valid - closing_price), 2)))
rms

In [ ]: train = new_data[:987]
valid = new_data[987:]
valid['Predictions'] = closing_price
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

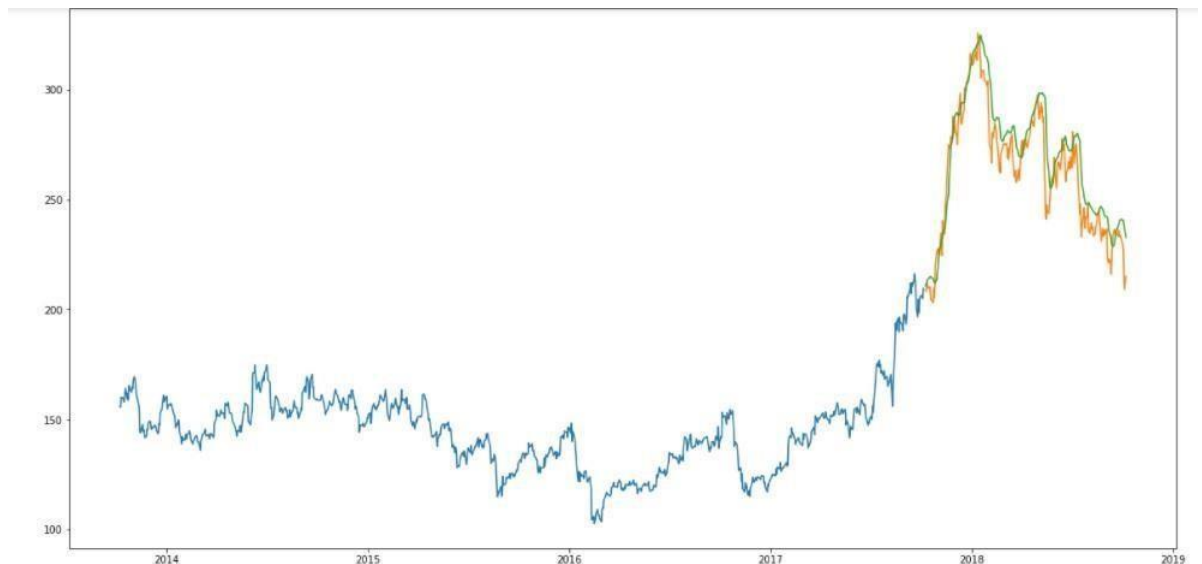


Figure 11. Prediction by LSTM (Graph between True and LSTM value)

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.

Conclusion

We have seen various methods (Linear Regression, KNN, LSTM, etc) to get better predictions in the stock market. Linear regression and the KNN method give us an idea to predict a situation but they did not give us a better understanding. So, we **look at some time series forecasting techniques (LSTM, Auto ARIMA, etc).** LSTM method gives us a better understanding as compared to others and it's widely used for sequence prediction problems and has proven to be extremely effective. So, Various stock market experts use this technique to get better results.