D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

Stock Prediction Using Machine Learning

D. Y. PATIL INTERNATIONAL UNIVERSITY
SUPERVISOR: DR.NIHARIKA SINGH
INTERNSHIP THESIS
BY: KRUSHAL SHAH
20190802058@dypiu.ac.in
2nd August 2021

Abstract

In today's time of covid when most of the world was under lockdown and a lot of businesses had slowed down during the period, stock market was still active and Forecasting of stock price is more important than ever. In this thesis, we have retrieved data from Quandel and Tiingo.

The data received is then processed so that it is easy to work and easy to fit the data in models to predict the stock price in future.

Two different models are used to compare which model would give better results. The two models are LSTM (Long-Short term Memory) and ARIMA (Auto-Regressive Integrated Moving Average).

All the code was developed on google colab as it was fast and easy to use with the added benefit of sharing the code with the supervisor.

The tools used are pandas, numpy, matplotlib, sklearn, statsmodels, keras and tensorflow.

Keywords
Forecasting, ARIMS, LSTM

Supervisor
Dr. Niharika Singh

Foreword

This is the first thesis that I have worked on and a lot of time and effort is gone into making the forecasting algorithms. There were a lot of mistakes and a lot of learning in the 2 months of the project.

I am thankful to the HOD of DYPIU Dr. Bahubali Shiragapur and the DYPIU team because of which I have received the opportunity to work on such a great project.

I would like to thank my supervisor, Dr. Niharika Singh, for mentoring me through the project.

And special thanks to the team members of the project who were there at every step of this project.

Krushal Shah
August 2, 2021

Contents

1 Introduction

Before proceeding with the project first we would have to know what the stock market and shares are.

In simple words, a share indicates a unit of ownership of the particular company. If you are a shareholder of a company, it implies that you, as an investor, hold a percentage of ownership of the issuing company. As a shareholder you stand to benefit in the event of the company's profits, and also bear the disadvantages of the company's losses.To buy these shares you have to go to the stock market.

There are 2 stock market in India first is NSE (National Stock Exchange) and the second one is BSE (Bombay Stock Exchange)

In this project we have used the data of 5 companies : MSFT Microsoft Corporation, AAPL APPLE, BA Boeing company, IBM and INTC Intel.

2 Objective

The objective of the project was to learn about the stock. How to import stock data and process the data. Further splitting the data and fitting it in a prediction algorithm to forecast the price of the stock in the future. Eventually finding the best algorithm suitable for prediction.

3 Related Work

3.1 Retrieving Data

The data can be retreated by two methods. First, the data could be downloaded from websites like Quandl and the second way is to use tools like Tiingo. Both these methods were used in the project.

3.2 Processing Data

The data received has more data than we need. So we have to clean the data and focus on the closing price of the stock. The date is set as an index so that the data is easier to read and handle.

3.3 Feature Scaling

There are two ways for performing feature scaling.

The first is standardization. This is done by using the preprocessor MinMaxScaler() in the package sklearn. In this, the data is scaled down between 0 and 1. It has a standard deviation of 0 and a mean of 0.

The second is normalization. This is done by using the preprocessor StandardScaler() in the package sklearn. In this, the data is scaled down so that it has a standard deviation of 1 and a mean of 0.

3.4 Fitting the Data

After the data is processed it is split into test and train data. The training data is then fitted in the prediction algorithm and the model is trained. After the data is trained it is tested on the test data.

In this project LSTM and ARIMA, 2 models were used.

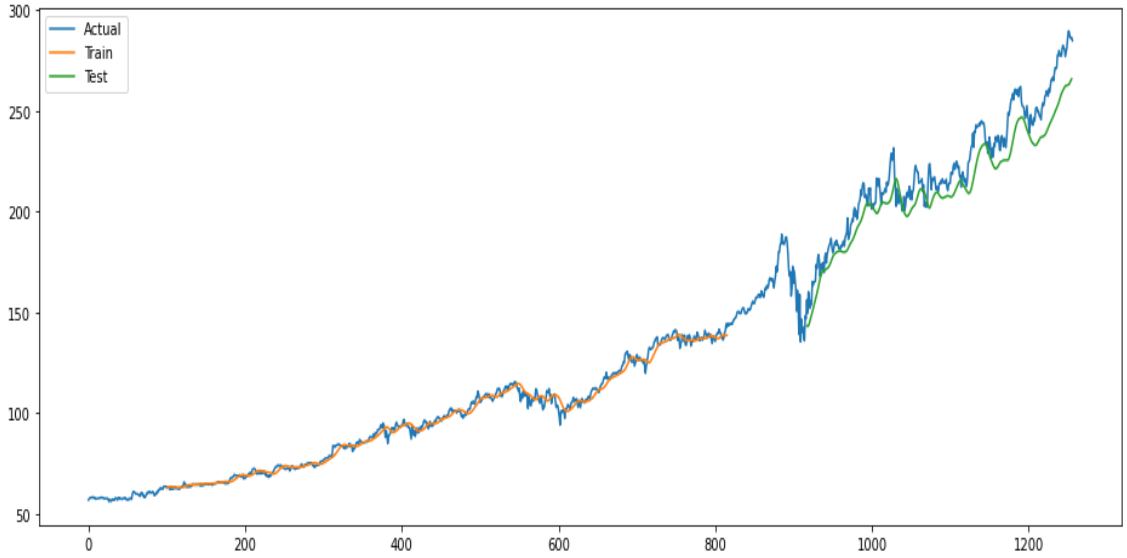
3.5 Research methodology

The project was created on google colab. Basic knowledge of shares and stock market was received by group discussion with the team and mentor. The data used in the project is collected by website Quandl and tool like Tiingo. The prediction algorithm was made with the help of youtube videos and official package documents of sklearn, keras, and tensorflow.

4 Results

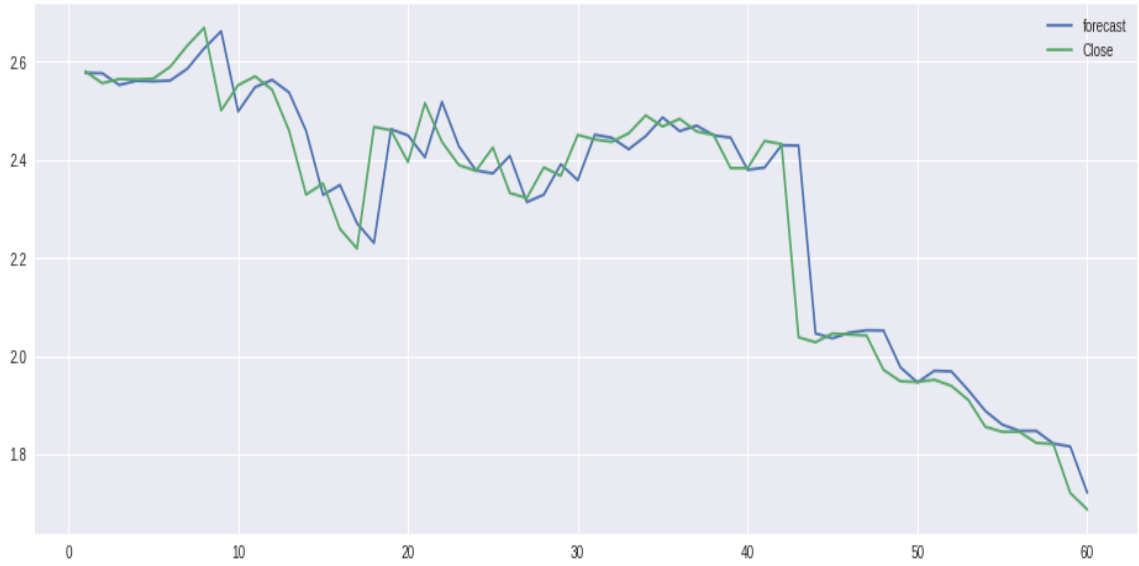
LSTM (RMSE) = 212.28833181413745

The graph below shows the stock price of Microsoft Corporation using LSTM model. In this the Actual data is plotted along side with the Train and Test Data. you can see how accurate the model is.



ARIMA (RMSE) = 100.06376152371435

The graph below shows the stock price of Microsoft Corporation using ARIMA. In this the Actual data is plotted along side with the Train and Test Data. you can see how accurate the model is.



5 Conclusion

In this document, we have seen various types of tools and different approaches that I have learned to use during the project

Analyzing all the data, the conclusion is that the LSTM model is an effective time series algorithm. The reason because this algorithm works so well is that it stores data in memory and learns from the previous data.

ARIMA model is also a good model however it is not very accurate for long time prediction. The margin of error keeps increasing as time keeps increasing.

6 References

Youtube Videos:
https://www.youtube.com/watch?v=gqryqIlvEoM
https://www.youtube.com/watch?v=JMT1eGJ9c2M
https://www.youtube.com/watch?v=H6du$_p fuznEt = 188s$

Documents:
https://api.tiingo.com/documentation/general/overview
https://faroit.com/keras-docs/2.0.2/
https://www.tensorflow.org/api$_d ocs/python/tf$

Appendix A LSTM

Import Packeges
import pandas$_{d}atareaderaspdr$
$importpandasaspd$
$importmatplotlib.pyplotasplt$
$importnumpyasnp$
$fromnumpyimportarray$
$fromsklearn.preprocessingimportMinMaxScaler$
$LSTMmodel$
$fromtensorflow.keras.modelsimportSequential$
$fromtensorflow.keras.layersimportDense$
$fromtensorflow.keras.layersimportLSTM$
$importtensorflowastf$
$importmath$
$fromsklearn.metricsimportmean_squared_error$
$importmatplotlib.pyplotasplt$
$frommatplotlibimportrcParams$
$rcParams['figure.figsize'] = 16, 6$

Data Collection
key=””

df = pdr.get$_{d}ata_{t}iingo('MSFT', api_{k}ey = key)$

df.to$_{c}sv('MSFT.csv')$

df=pd.read$_{c}sv('MSFT.csv')$

df.head()

df1=df.reset$_{i}ndex()['close']$

plt.plot(df1);

LSTM are sensitive to the scale of the data. so we apply
MinMax scaler

scaler=MinMaxScaler(feature$_{r}ange = (0, 1))$
$df1 = scaler.fit_{t}ransform(np.array(df1).reshape(-1, 1))$

splitting dataset into train and test split
training$_{s}ize = int(len(df1) * 0.65)$
$test_{s}ize = len(df1) - training_{s}izetrain_{d}ata, test_{d}ata = df1[0 : training_{s}ize, :], df1[training_{s}ize : len(df1), : 1]$

training$_s ize, test_s ize$

convert an array of values into a dataset matrix
def create$_d ataset(dataset, time_s tep = 1)$:
$dataX, dataY = [], []$
$for i in range(len(dataset) - time_s tep - 1)$:
$a = dataset[i : (i + time_s tep), 0] i = 0, 0, 1, 2, 3 - - - - - 99100$
$dataX.append(a)$
$dataY.append(dataset[i + time_s tep, 0])$
$return numpy.array(dataX), numpy.array(dataY)$

reshape into X=t,t+1,t+2,t+3 and Y=t+4
import numpy
time$_s tep = 100$
$X_t rain, y_t rain = create_d ataset(train_d ata, time_s tep)$
$X_t est, ytest = create_d ataset(test_d ata, time_s tep)$

print(X$_t rain.shape), print(y_t rain.shape)$

print(X$_t est.shape), print(ytest.shape)$

reshape input to be [samples, time steps, features] which is required for LSTM
X$_t rain = X_t rain.reshape(X_t rain.shape[0], X_t rain.shape[1], 1)$
$X_t est = X_t est.reshape(X_t est.shape[0], X_t est.shape[1], 1)$

model=Sequential()
model.add(LSTM(50,return$_s equences = True, input_s hape = (100, 1)))$
$model.add(LSTM(50, return_s equences = True))$
$model.add(LSTM(50))$
$model.add(Dense(1))$
$model.compile(loss =' mean_s quared_e rror', optimizer =' adam')$

model.summary()

model.summary()

model.fit(X$_t rain, y_t rain, validation_d ata = (X_t est, ytest), epochs = 100, batch_s ize = 64, verbose = 1);$

Lets Do the prediction and check performance metrics
train$_p redict = model.predict(X_t rain)$
$test_p redict = model.predict(X_t est)$

Transformback to original form
train$_p redict = scaler.inverse_t ransform(train_p redict)$

$test_predict = scaler.inverse_transform(test_predict)$

Calculate RMSE performance metrics
math.sqrt(mean_$squared_error(y_train, train_predict)$)

Test Data RMSE
math.sqrt(mean_$squared_error(ytest, test_predict)$)

Plotting
shift train predictions for plotting
look_$back = 100$
$trainPredictPlot = numpy.empty_like(df1)$
$trainPredictPlot[:, :] = np.nan$
$trainPredictPlot[look_back : len(train_predict) + look_back, :] = train_predict$
$shifttestpredictionsforplotting$
$testPredictPlot = numpy.empty_like(df1)$
$testPredictPlot[:, :] = numpy.nan$
$testPredictPlot[len(train_predict) + (look_back * 2) + 1 : len(df1) - 1, :] = test_predict$
$plotbaselineandpredictions$
$plt.plot(scaler.inverse_transform(df1), label =' Actual')$
$plt.plot(trainPredictPlot, label =' Train')$
$plt.plot(testPredictPlot, label =' Test')$
$plt.legend();$
$plt.show();$

len(test_$data)$

x_$input = test_data[341 :].reshape(1, -1)$
$x_input.shape$

temp_$input = list(x_input)$
$temp_input = temp_input[0].tolist()$

demonstrate prediction for next 10 days
lst_$output = []$
$n_steps = 99$
$i = 0$
$while(i < 30) :$

if(len(temp_$input) > 99) :$
$print(temp_input)$
$x_input = np.array(temp_input[1 :])$
$print("dayinput".format(i, x_input))$

```
x_input = x_input.reshape(1, -1)
x_input = x_input.reshape((1, n_steps, 1))
print(x_input)
yhat = model.predict(x_input, verbose = 0)
print("dayoutput".format(i, yhat))
temp_input.extend(yhat[0].tolist())
temp_input = temp_input[1 :]
print(temp_input)
lst_output.extend(yhat.tolist())
i = i + 1
else :
x_input = x_input.reshape((1, n_steps, 1))
yhat = model.predict(x_input, verbose = 0)
print(yhat[0])
temp_input.extend(yhat[0].tolist())
print(len(temp_input))
lst_output.extend(yhat.tolist())
i = i + 1


day_new = np.arange(1, 101)
day_pred = np.arange(101, 131)


plt.plot(day_new, scaler.inverse_transform(df1[1157 :]));
plt.plot(day_pred, scaler.inverse_transform(lst_output));


df3=df1.tolist()
df3.extend(lst_output)
plt.plot(df3[1200 :]);


df3=scaler.inverse_transform(df3).tolist()


plt.plot(df3);
```

Appendix B ARIMA

```
from google.colab import drive
drive.mount('/content/drive')


For ARIMA
from IPython.core.debugger import set_trace


import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['figure.figsize'] = 16,6
import time
import math


plt.style.use(style ="seaborn")


df1 = pd.read_csv("/content/drive/MyDrive/DYPIUSEM4/INTERNSHIP/EOD - MSFTMicrosoftCorporationMAX.csv")
df1.head()


plt.xlabel("Date")
plt.ylabel("Stock price")
plt.title("Microsoft Corporation")
plt.plot(df1.Date,df1.Open, label = "Open");
plt.plot(df1.Date,df1.High, label = "Hign");
plt.plot(df1.Date,df1.Low, label = "Low");
plt.plot(df1.Date,df1.Close, label = "Close");
plt.legend();
rcParams['figure.figsize'] = 16,6


**Using Date As Index**

Set Date as Index and Make A Graph
df1['Date'] = pd.to_datetime(df1.Date)
df1['Date'] = pd.to_datetime(df1.Date, format =' df1.index = df1['Date']
plt.xlabel("Date")
plt.ylabel("StockCloseprice")
plt.title("MicrosoftClosedata(usingdateasindex)")
plt.plot(df1.Close);
```

Scaling The data Using MinMaxScaler

import sklearn.preprocessing
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature$_range = (0, 1))$
$df1_MinMaxScaler = scaler.fit_transform(df1[['Close','Volume']])$
$print(df1_scaler)$

Plotting Graph of Scaled Data Using MinMaxScaler(Close price)

plt.xlabel("Date")
plt.ylabel("Stock Close price(Scaled)")
plt.title("Microsoft Close data (MinMaxScaler)")
plt.plot(df1.Date , df1$_MinMaxScaler[:, 0], label = ' Microsoftclosepricescaler');$

plt.xlabel("Date")
plt.ylabel("Volume(Scaled)")
plt.title("Microsoft Volume (MinMaxScaler)")
plt.plot(df1.Date , df1$_MinMaxScaler[:, 1], label = ' Microsoftclosepricescaler');$

Scaling Using Standard Scaler

from sklearn.preprocessing import StandardScaler
SS = StandardScaler(feature$_range(-1, 1))$
$SS = StandardScaler(with_mean = True, with_std = True)$
$df1_SS = SS.fit_transform(df1[['Close','Volume']])$

plt.xlabel("Date")
plt.ylabel("Stock Close price(Scaled)")
plt.title("Microsoft Close data (StandardScaler)")
plt.plot(df1.Date , df1$_SS[:, 0], label = ' Microsoftclosepricescaler');$

plt.xlabel("Date")
plt.ylabel("Volume(Scaled)")
plt.title("Microsoft Volume (StandardScaler)")
plt.plot(df1.Date , df1$_SS[:, 1], label = ' Microsoftclosepricescaler');$

print("std: " , round(df1$_MinMaxScaler.std()))$
$print("mean : ", round(df1_MinMaxScaler.mean()))$

print("std: " , round(df1$_SS.std()))$
$print("mean : ", round(df1_SS.mean()))$

df1$_M inMaxScaler$ = $pd.DataFrame(df1_M inMaxScaler, columns = ['Close','Volume'], dtype = float)$
$df1_M inMaxScaler.head()$

df1$_S S = pd.DataFrame(df1_S S, columns = ['Close','Volume'], dtype = float)$

df1$_O nly_C lose = df1[["Close"]].copy()$

df1$_O nly_C lose.describe()$

ADF Test

Check if price series is stationary
from statsmodels.tsa.stattools import adfuller

result = adfuller(df1$_S S.Close.dropna())$
$print(f"ADF Statistic : result[0]")$
$print(f"p - value : result[1]")$

Autocorrelation Function(ACF) To find D

from statsmodels.graphics.tsaplots import plot$_a cf$

fig , (ax1 , ax2) = plt.subplots(1 , 2 , figsize = (16 , 4))

ax1.plot(df1$_S S.Close)$
$ax1.set_t itle("Original")$

add ; at the end of the plot function so that the plot is not duplicated
plot$_a cf(df1_S S.Close, ax = ax2);$

diff = df1$_S S.Close.diff().dropna()$

fig , (ax1 , ax2) = plt.subplots(1 , 2 , figsize =(16 , 4))

ax1.plot(diff)
ax1.set$_t itle("Differenceonce")$
$plot_a cf(diff, ax = ax2);$

diff = df1$_S S.Close.diff().dropna()$

fig , (ax1 , ax2) = plt.subplots(1 , 2 , figsize =(16 , 4))

ax1.plot(diff)
ax1.set$_t itle("DifferenceTwice")$
$plot_a cf(diff, ax = ax2);$

We can also use pmdarima package to get the number of differencing

!pip install pmdarima

from pmdarima.arima.utils import ndiffs

ndiffs(df1$_S S.Close, test = "adf")$

D = 1

P order of Auto Regressive term (AR)

from statsmodels.graphics.tsaplots import plot$_p acf$

diff = df1$_S S.Close.diff().dropna()$

fig , (ax1 , ax2) = plt.subplots(1 , 2 , figsize =(16 , 4))

ax1.plot(diff)
ax1.set$_t itle("DifferenceTwice")$
$ax2.set_y lim(0, 1)$
$plot_p acf(diff, ax = ax2);$

P = 11 from the graph

Q Moving Average (MA)

diff = df1$_S S.Close.diff().dropna()$

fig , (ax1 , ax2) = plt.subplots(1 , 2 , figsize =(16 , 4))

ax1.plot(diff)
ax1.set$_t itle("DifferenceOnce")$
$ax2.set_y lim(0, 1)$
$plot_a cf(diff, ax = ax2);$

Q = 11

Fitting the ARIMA model
order = (p,d,q)

from statsmodels.tsa.arima$_m odelimportARIMA$

ARIMA Model
model = ARIMA(df1$_S S.Close, order = (11, 1, 1))$
$result = model.fit(disp = 0)$

print (result.summary())
Plot residual errors
residuals = pd.DataFrame(result.resid)
fig , (ax1 , ax2) = plt.subplots(1 , 2 , figsize =(16 , 4))

```
ax1.plot(residuals)
ax2.hist(residuals , density = True);
```

Actual vs Fitted

result.plot$_predict($
$start = 1,$
$end = 60,$
$dynamic = False,$
$);$

Forcasting

Train test Split

```
n = int (len(df1_SS) * 0.8)
train = df1_SS.Close[: n]
test = df1_SS.Close[n :]


print(len(train))
print(len(test))


model = ARIMA(train,order = (11,1,1))
result = model.fit(disp = 0)

step = 30
fc,se,conf = result.forecast(step)


fc = pd.Series(fc , index = test [:step].index)
lower = pd.Series (conf[:,0] , index = test[:step].index)
upper = pd.Series (conf[:,1] , index = test[:step].index)


plt.plot(test[:step],label = ”actual”)
plt.plot(fc , label=”forecast”)
plt.fill_between(lower.index, lower, upper, color = "k", alpha = 0.1)
plt.title("ForecastvsActual")
plt.legend();
```