Machine Learning Engineer Nanodegree

Capstone Project

Prakhar Tripathi

April 7, 2019

I. Definition

Project Overview

- The project deals with the prediction of stock prediction of AMD which is under NASDAQ where lots of investment is being provided to investors.
- This project is also helpful for person not engaged in stock and marketing professions using the graphs for comparison.
- 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. physhological, rational and irrational behaviour, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.
- In this article, 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 move on to advanced techniques like LSTM.
- Technology used is python and its various libraries of scientific and mathematical problem.

Problem Statement

The project deals with the historical data set of AMD where several features are mentioned(open,close,volume etc.) in data set.These features are the essence to solve any of the stocks related problem.

- The problem is clearly mentioned in layman word and easy .The proper display of data sets is provided in each step.
- I have discussed the topic with some alumni of the Udacity in past and some of my researcher friends.
- After analyzing the proper clearly from my subordinates I tried to make the presentation of data sets more precise and easy to understand.
- The problem is divided in some steps as-

A)Fundamental Analysis involves analyzing the company’s future profitability on the basis of its current business environment and financial performance.

B)Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

Metrics

Metrics are an important method to improve functionality of a model .It reduces error and helps in better prediction .All the metrics are provided easily by scikit library of python.

- The metrics I discussed is clearly stated and is root mean squared error method in scikit library.It is a better method in solving and it clearly solves the mentioned problem.The LSTM is also checked using back propagation technique.
- These methodologies are well discussed and are used in many of the current problem solving situations.
- The main metrics used is Mean squared error.
- RMSE(root mean squared evaluation)-It is very good model for large continuous data .Unlike the absolute mean error it severely punishes large Error.

It is-

$$((Y_{pred}-Y_{obs})^2/n))^{1/2}=\text{Error}$$

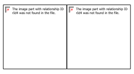
Y_{pred}-linear regression output

Y_{obs-}-Neural network output

n - total input

d/d(E)=gradient descent

-



II. Analysis

Data Exploration

In this project the the work is performed on Jupyter notebook provided by Anaconda .The project is been saved in '.ipynb' format.The data used is completely in date time format using date time parsing method for futuristic predictions.   Initially all the features have been visualized properly but later many of features have been reduced for better performance.

- The data set can easily be extracted from yahoo finance and is the most reliable data set present.To maintain dynamics the data extraction is connected to API.
- The data set dynamics helps to maintain the better management of project .This thing is clearly visible in the the code line where value of t is asked.

- In the data set it is clearly mentioned that the data timing is between (2009,5,22) to   (2018,8,29) .

- The various statistical features like mean ,standard deviation ,max and min are defined very properly too.
- The top and the bottom values are mentioned too.
- But if the data set is not present it would be hard to predict anything about.In that case one shall be creating data set using

hadoop by studying the data base of a firm.The rest features will be included in the project after a deep analysis too.

```
datl= pd.read_csv(r'A.csv')
print (datl.head())
print ('\n Data Types:')
print (datl.dtypes)
```

```
         Date       High        Low       Open      Close     Volume  \
0  2009-05-21  13.154507  12.510730  13.032905  12.646638  4439900.0
1  2009-05-22  12.804006  12.482118  12.703862  12.653791  3602900.0
2  2009-05-26  12.939914  12.446352  12.632332  12.911302  3461500.0
3  2009-05-27  13.090129  12.753934  12.939914  12.796853  3757800.0
4  2009-05-28  13.018598  12.517882  12.947067  12.861230  3126600.0

   Adj Close
0  11.648037
1  11.654627
2  11.891805
3  11.786391
4  11.845683

 Data Types:
Date           object
High          float64
Low           float64
Open          float64
Close         float64
Volume        float64
Adj Close     float64
dtype: object
```

- Removal of features of less significance is done using minmaxscaler.
- The data set is clearly perfect in itself as it includes almost all the features necessary for good prediction like the min and max features.
- Since the data prediction is made on volume analysis the graph and tabular information is clearly provided in here.

```
import random
ts=d['Adj Close']
t=random.choice(ts)
```

```
t
```

```
26.27754020690918
```

```
d.shape
```

```
(2336, 6)
```

- The start and end timing of data are as follows.

```
print (d.head(10))
print (d.tail(10))
```

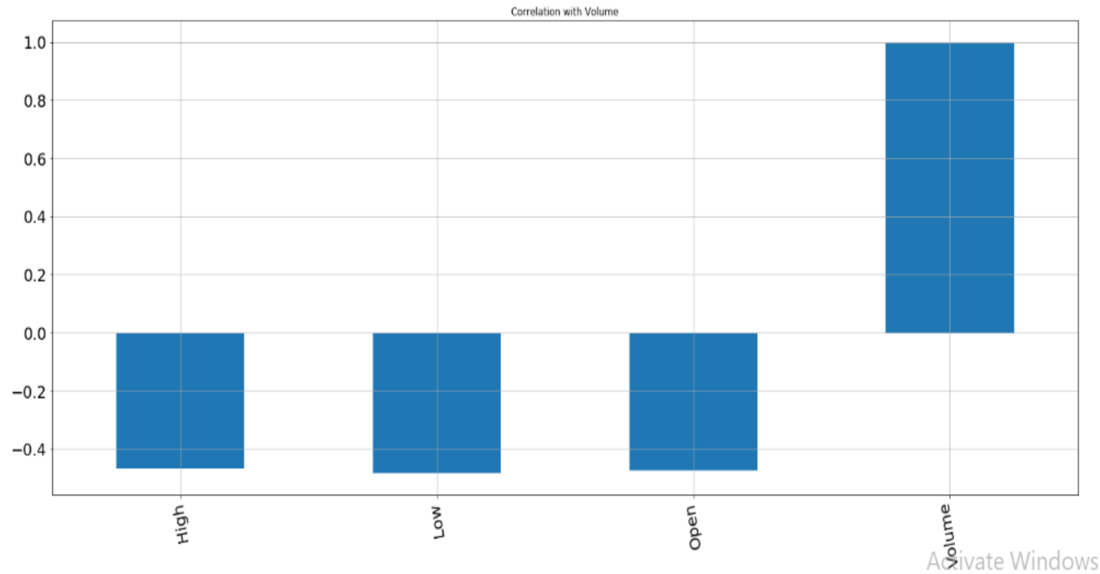
```
                 High        Low       Open      Close     Volume  Adj Close
Date
2009-05-21  13.154507  12.510730  13.032905  12.646638  4439900.0  11.648037
2009-05-22  12.804006  12.482118  12.703862  12.653791  3602900.0  11.654627
2009-05-26  12.939914  12.446352  12.632332  12.911302  3461500.0  11.891805
2009-05-27  13.090129  12.753934  12.939914  12.796853  3757800.0  11.786391
2009-05-28  13.018598  12.517882  12.947067  12.861230  3126600.0  11.845683
2009-05-29  13.040057  12.711016  12.911302  13.040057  2769200.0  12.010394
2009-06-01  13.741058  13.190271  13.190271  13.505007  5764500.0  12.438632
2009-06-02  14.163090  13.469242  13.497854  13.984263  5233600.0  12.880044
2009-06-03  14.020029  13.705294  13.941345  13.826896  5618100.0  12.735099
2009-06-04  14.463519  13.869814  13.869814  14.334764  4408100.0  13.202869
                 High        Low       Open      Close     Volume  Adj Close
Date
2018-08-16  65.669998  64.809998  65.040001  65.500000  3149500.0  65.067375
2018-08-17  65.480003  64.279999  65.480003  64.660004  2449200.0  64.232925
2018-08-20  65.110001  64.459999  64.570000  64.470001  1619900.0  64.044182
2018-08-21  65.400002  64.620003  64.739998  64.930000  2982800.0  64.501137
2018-08-22  65.870003  64.550003  64.739998  65.650002  2565600.0  65.216385
2018-08-23  65.989998  65.529999  65.690002  65.690002  2287700.0  65.256126
2018-08-24  66.260002  65.589996  65.739998  65.980003  1904200.0  65.544205
2018-08-27  66.349998  65.860001  66.330002  66.080002  1158800.0  65.643539
2018-08-28  67.300003  66.349998  66.410004  66.690002  2284300.0  66.249519
2018-08-29  67.279999  66.400002  66.690002  67.010002  1852100.0  66.567413
```

Exploratory Visualization

The data visualization is done by using matplot library of python.There are various types of graph being used such as histograms ,dotted graph and normal linear curves. In LSTM the graph is plotted in a non linear format .

- In the beginning of the project a comparative histogram chart is plotted for various features of data.
- The next graph depicts the volume of stock sold in various time intervals.
- There is a graph depicting the high and low column.
- After tensor flow module a graph is being plotted for the distinction between actual and predicted price.

<matplotlib.axes._subplots.AxesSubplot at 0x280d1224710>



Algorithms and Techniques

Various algorithms of machine learning are being used and especially of better performance .

- One of the algorithm used is linear regression and decision tree classifier which are important part of supervised learning methodology.
- The linear regression uses continuous data form for classification.
- The decision algorithm is available for both classification and regression.
- The concept of normalization is used thoroughly to remove features of lesser significant features.
- In metrics gradient descent algorithm reduces error every time.
- In neural network of LSTM uses feed-forward and back propagation for executing of features and reducing of errors.
- LSTM is a special neural network algorithm where data of correction is stored in memory buffer.
- Each of the the feature can extracted using scikit learning library.
- In LSTM keras library has been implemented.
- To use the linear regression and decision trees the data set is first being separated into training and testing parts.
- The metrics is done on the testing and training parts too.
- In neural network matrix type data is broken in vectors and used as perceptron.

Linear Regression-**Linear regression** is a **linear** approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple **linear regression**.

$$a = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

LSTM-**Long short-term memory** (**LSTM**) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections that make it a "general purpose computer"

$$\begin{aligned}
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i\right) \\
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f\right) \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o\right) \\
\tilde{C}_t &= \tanh\left(x_t U^g + h_{t-1} W^g\right) \\
C_t &= \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right) \\
h_t &= \tanh(C_t) * o_t
\end{aligned}$$

Gradient Descent-Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.

$$f'(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N} \sum -2(y_i - (mx_i + b)) \end{bmatrix}$$

Benchmark

The benchmark model has been implemented as Linear regression as presented in project .This model has shown how the futuristic models shold work.

- The result clearly depicts the RMSE score and R2 score which are outcome as of implementation of Linear regression.
- In statistical analysis of model the benchmark model provides actual observation while other provides hypothetical or predicted model.
- Linear regression model score has also been predicted.
- Also an additional benchmark of decision tree regressor is used which is having better score.

III. Methodology

Data Preprocessing

Data preprocessing is an important methodology to identify and learn data.It also helps in identification of important features and removal of many unnecessary features.It involves transforming raw **data** into an understandable format. Real-world **data** is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. **Data preprocessing** is a proven method of resolving such issues.

An alternative approach to Z-score normalization (or standardization) is the so-called **Min-Max scaling**(often also simply called "normalization" - a common cause for ambiguities).
In this approach, the data is scaled to a fixed range - usually 0 to 1.
The cost of having this bounded range - in contrast to standardization - is that we will end up with smaller standard deviations, which can suppress the effect of outliers.

A Min-Max scaling is typically done via the following equation:

$$X_{norm}=X-X_{min}/X_{max}-X_{min}$$

```
#feature reduction
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
feature_minmax_transform_data = scaler.fit_transform(test[feature_columns])
feature_minmax_transform = pd.DataFrame(columns=feature_columns, data=feature_minmax_transform_data, index=test.index)
feature_minmax_transform.head()
```

	Open	High	Low	Adj Close	Volume
Date					
2009-05-21	0.006429	0.005635	0.001041	0.000000	0.157321
2009-05-22	0.001148	0.000000	0.000578	0.000106	0.123619
2009-05-26	0.000000	0.002185	0.000000	0.003911	0.117926
2009-05-27	0.004937	0.004600	0.004973	0.002220	0.129856
2009-05-28	0.005051	0.003450	0.001156	0.003171	0.104442

Implementation

Step1:

.Data picking-a)Pandas- python library

import pandas as pd

import datetime

import pandas_datareader as web

from pandas_datareader import data

#dynamic dataset

tickers = ['AMD']

d = web.DataReader("A",'yahoo',start,end)

d.to_csv('A.csv')

Step2:

Data Preprocessing

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

training_set_scaled = sc.fit_transform(training_set)

Step3:

Data comparison

import matplotlib.pyplot as plt

X=d.drop(['Adj Close'],axis=1)

X=X.drop(['Close'],axis=1)

Step4.

Date-Time Analysis

m = pd.read_csv(r'A.csv', parse_dates=['Date'],
na_values=['990.99'],index_col = ['Date'])

cal = m[start :end]

cal.head()

#plot

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

plt.plot(d['Adj Close'], label='Close Price history')

Step 5:

LSTM analysis

Importing the Keras libraries and packages

from keras import *

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Dropout

Step6:

Linear Regression

from sklearn.linear_model import LinearRegression

lin=LinearRegression()

lin.fit(X_train, y_train)

lin.score(X_train, y_train)

```
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

regressor.fit(X_train, y_train, epochs = 50, batch_size = 16)
```

```
Epoch 31/50
2275/2275 [==============================] - 27s 12ms/step - loss: 0.0010
Epoch 32/50
2275/2275 [==============================] - 28s 12ms/step - loss: 8.8617e-04
Epoch 33/50
2275/2275 [==============================] - 31s 13ms/step - loss: 9.2178e-04
Epoch 34/50
2275/2275 [==============================] - 33s 15ms/step - loss: 8.1436e-04
Epoch 35/50
2275/2275 [==============================] - 28s 12ms/step - loss: 8.5403e-04
Epoch 36/50
2275/2275 [==============================] - 27s 12ms/step - loss: 8.9073e-04
Epoch 37/50
2275/2275 [==============================] - 27s 12ms/step - loss: 7.8265e-04
Epoch 38/50
2275/2275 [==============================] - 28s 12ms/step - loss: 7.7393e-04
Epoch 39/50
2275/2275 [==============================] - 27s 12ms/step - loss: 7.0199e-04
Epoch 40/50
2275/2275 [==============================] - 29s 13ms/step - loss: 8.0856e-04
```


epochs

Refinement

The preprocessing step involves the betterment of data sets which accordingly is very necessary.

Initially

2009-05-21 13.154507 12.510730 13.032905 12.646638 4439900.0 11.648037

Finally

2009-05-21 0.006429 0.005635 0.001041 0.000000 0.157321

One thing every new developer working further on this project must understand that some times this code may not work as same in project. Many times there may be problem sometime with your system such as any upgraded library or any hardware problem that may be upgraded accordingly. For eg. the tensorflow module version may lead to no result or error.

```
y_pred_test_lstm = model_lstm.predict(X_tst_t)
y_train_pred_lstm = model_lstm.predict(X_tr_t)
print("The R2 score on the Train set is:\t{:0.3f}".format(r2_score(y_train, y_train_pred_lstm)))
r2_train = r2_score(y_train, y_train_pred_lstm)

print("The R2 score on the Test set is:\t{:0.3f}".format(r2_score(y_test, y_pred_test_lstm)))
r2_test = r2_score(y_test, y_pred_test_lstm)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-56-60dcb1a76f00> in <module>
      1
      2
```

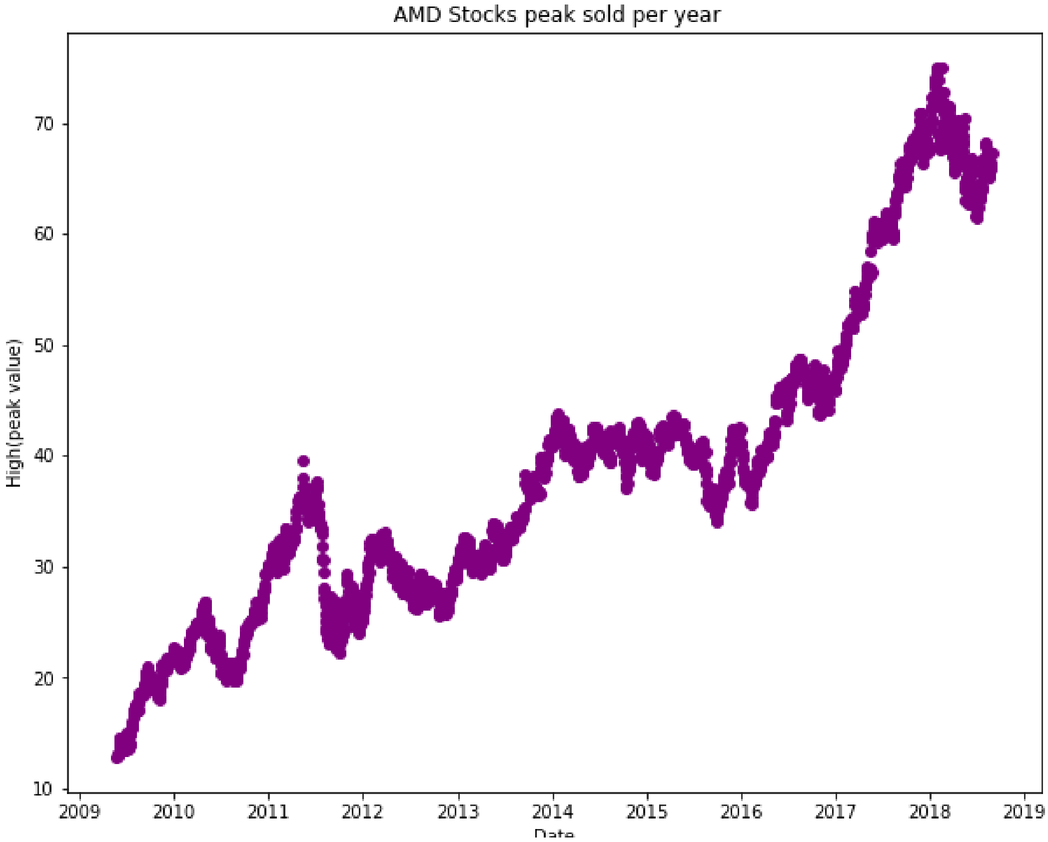
Model Evaluation and Validation

Although this data set is good to predict using normal methods of supervised learning.But using LSTM gives it an upper edge and makes better predictions.

This model is highly robust for any type of prediction because of its extraction of data set from API.Although it`s limitation is that this model could only be performed online.

Initial graphing -a)This depicts the growth in stock performance each year.

Out[34]: [<matplotlib.lines.Line2D at 0x280d90f0ac8>]



B)This depicts the linear regression (benchmark result) with 0.48 score with grah depicting testing and training set results.

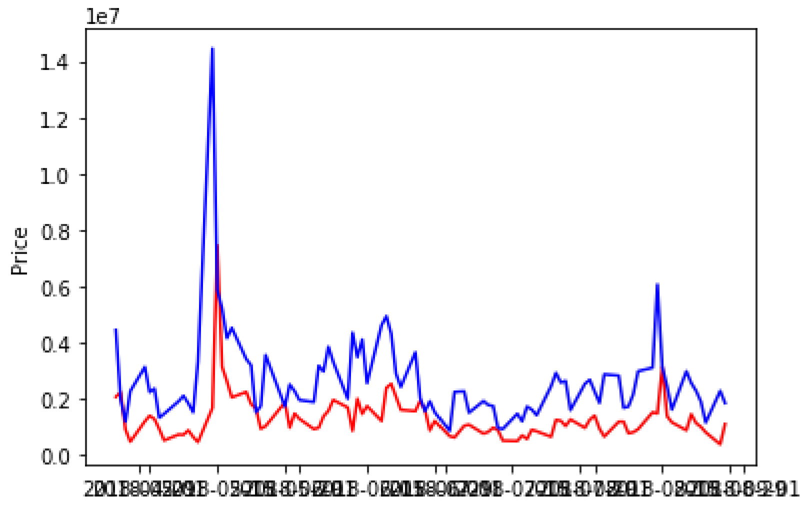
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

dt = LinearRegression()

benchmark_dt=dt.fit(X_train, y_train)

validate_result(benchmark_dt, 'Linear Regression')
```

```
RMSE:   2028054.6556227256
R2 score:   -0.48549560657961277
```



C) This section of result depicts the result on the basis of LSTM performance with test set performing very well w.r.t. test set hence it is over fitting situation.

```
2106/2106 [==============================] - 1s 341us/step - loss: 1.0833
Epoch 20/20
2106/2106 [==============================] - 1s 327us/step - loss: 0.9035
```

```
y_pred_test_lstm = model_lstm.predict(X_tst_t)
y_train_pred_lstm = model_lstm.predict(X_tr_t)
print("The R2 score on the Train set is:\t{:0.3f}".format(r2_score(y_train, y_train_pred_lstm)))
r2_train = r2_score(y_train, y_train_pred_lstm)

print("The R2 score on the Test set is:\t{:0.3f}".format(r2_score(y_test, y_pred_test_lstm)))
r2_test = r2_score(y_test, y_pred_test_lstm)
```

```
The R2 score on the Train set is:       0.994
The R2 score on the Test set is:        0.358
```

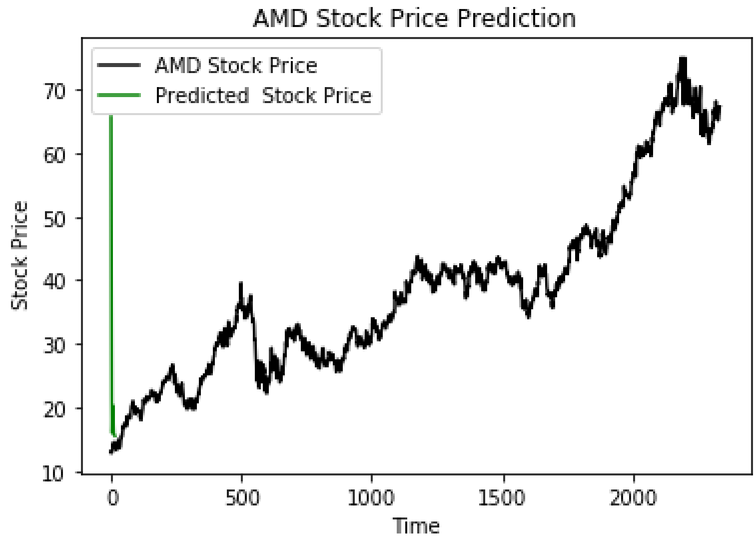
```
lstm= model_lstm.evaluate(X_tst_t, y_test, batch_size=1)
```

```
140/140 [==============================] - 0s 3ms/step
```

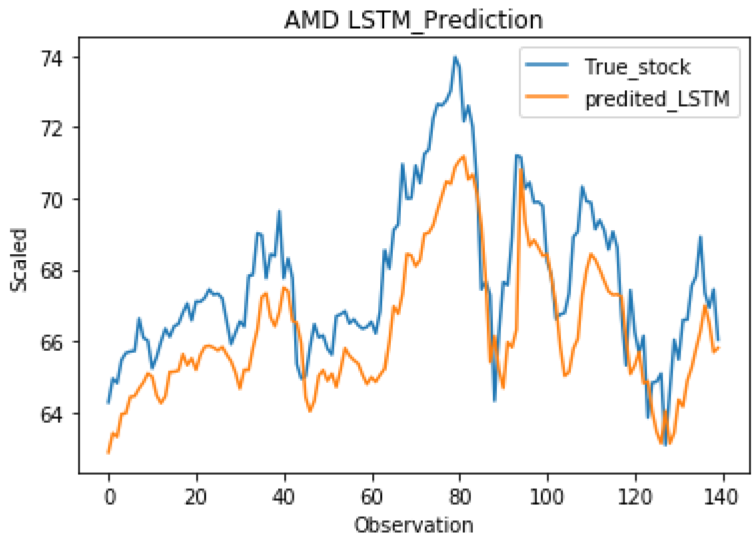
```
print('LSTM: %f'%lstm)
```

```
LSTM: 3.153766
```

```
y_pred_test_LSTM = model_lstm.predict(X_tst_t)
```



D)This is the final output where there is comparison between true and predicted result.



Hyper parameters are also used to increase optimization-

A)Ingraph-

```
X.corrwith(d['Volume']).plot.bar(
        figsize = (25, 10), title = "Correlation with Volume", fontsize = 20,
        rot = 100, grid = True)
```

Parameters like size ,title name etc defined.

B)In algorithms

```
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

regressor.fit(X_train, y_train, epochs = 50, batch_size = 16)
```

Various parameters like type of metrics ,layer size ,epoch and no. Of layers is defined.

Justification

A)In this portion there is a comparison of scores between Linear regression and decision tree where train score is much good for decision tree(1.0)and because of this it overfits the process hence it is not much optimal for learning.

```
from sklearn.linear_model import LinearRegression
lin=LinearRegression()
lin.fit(X_train, y_train)
lin.score(X_train, y_train)
```

```
0.4502059772388196
```

i

```
from sklearn import tree

clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
clf.score(X_train, y_train)
```

```
1.0
```

B)LSTM analysis was found to be good too with great test score.

```
2106/2106 [==============================] - 1s 341us/step - loss: 1.0833
Epoch 20/20
2106/2106 [==============================] - 1s 327us/step - loss: 0.9035
```

```
y_pred_test_lstm = model_lstm.predict(X_tst_t)
y_train_pred_lstm = model_lstm.predict(X_tr_t)
print("The R2 score on the Train set is:\t{:0.3f}".format(r2_score(y_train, y_train_pred_lstm)))
r2_train = r2_score(y_train, y_train_pred_lstm)

print("The R2 score on the Test set is:\t{:0.3f}".format(r2_score(y_test, y_pred_test_lstm)))
r2_test = r2_score(y_test, y_pred_test_lstm)
```

```
The R2 score on the Train set is:	0.994
The R2 score on the Test set is:	0.358
```

```
lstm= model_lstm.evaluate(X_tst_t, y_test, batch_size=1)
```

```
140/140 [==============================] - 0s 3ms/step
```

```
print('LSTM: %f'%lstm)
```

```
LSTM: 3.153766
```

```
y_pred_test_LSTM = model_lstm.predict(X_tst_t)
```

Robustness-The model shows also great robustness.It is changing every time with small change.

```
import random
ts=d['Adj Close']
t=random.choice(ts)
```

```
t
```

```
63.24946212768555
```

```
import random
ts=d['Adj Close']
t=random.choice(ts)
```

```
t
```

```
26.27754020690918
```

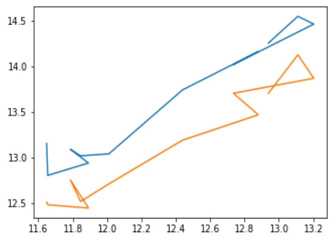
This is clearly visible from above where value of `t` changes every time fetching data from API.eg-

For 12 values

```
from matplotlib.dates import (YEARLY, DateFormatter,
                              rrulewrapper, RRuleLocator, drange)
x=d['Adj Close'].head(12)
y=d['High'].head(12)
z=d['Low'].head(12)
```

```
plt.plot(x,y)
plt.plot(x,z)
```

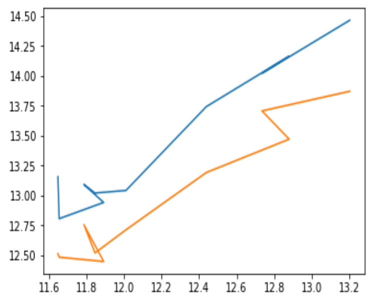
```
[<matplotlib.lines.Line2D at 0x24c1a56a358>]
```



For 10 values

```
plt.plot(x,y)
plt.plot(x,z)
```

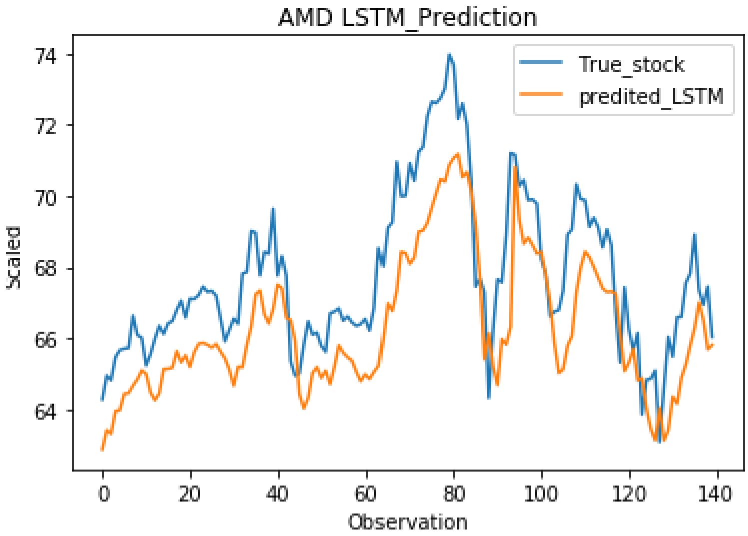
```
[<matplotlib.lines.Line2D at 0x164552c76d8>]
```



V. Conclusion

The project has been completed with best prediction using the particular techniques.

It is almost impossible to find 100% accuracy in modelling of stock prediction .But the line predicted could provided trader with idea on investment .for eg-



Here the red line shows our prediction which has almost same shape as blue one which is actual values but it differs many a time in size and shape.

But still it provides investors a great idea about stocks.

It is interesting to see that our training data works very well but it faces some difficulties in data sets.It is also to be noticed that most of the time the shape is same of both actual and predicted one.

However with linear regression this model works very well but exploration of new techniques can provide newer results.

New methods like random forest or k-map could also provide good /bad results too.

```python
from sklearn.neighbors import KNeighborsRegressor

dt = KNeighborsRegressor(n_neighbors=5)

benchmark_dt=dt.fit(X_train, y_train)

validate_result(benchmark_dt, 'KNN')
```

```
RMSE:  1982319.25717146
R2 score:  -0.4192511723346153
```