Machine Learning Engineer Nanodegree

Capstone Project

Syeda Zainab Akhtar

May 7, 2020

I. Definition

Project Overview

The project deals with the analysis of NASDAQ market AMD where a lot of capital is being made accessible to buyers.This project is often useful for individuals who are not interested in the stock market who use graphs for comparative purposes. Forecasting the success of the stock market is one of the most challenging tasks to do. There are too many considerations involved in predicting – physical conditions vs. physological, moral and unreasonable behavior, etc.Many of these factors contribute to render share prices unpredictable and extremely challenging to forecast with a large degree of precision.In this article, we must deal with historical statistics on the stock prices of a publicly listed firm. We will use a variety of machine learning algorithms to forecast the future stock price of this product, beginning with simple algorithms such as averaging and linear regression, and then using advanced techniques such as LSTM.The hardware utilized is the science and technical issue of Python and its numerous libraries.

Problem Statement

The project includes the historical AMD data set where a variety of features (open, close, length, etc.) are included in the data collection process. Such features are the essence of any inventory-dependent problem.

The question is clearly defined in the layman word and simple. The proper view of the data sets is given in every stage.

In the past, I have addressed the topic with several Udacity alumni and several of my researcher colleagues.

After reviewing the information directly from my superiors, I tried to make the description of the data sets more precise and easy to grasp.

The question is split in several phases as -Fundamental research includes evaluating the potential viability of the firm on the grounds of the present operating climate and financial results.

Economic research, on the other hand, means interpreting the charts and utilizing mathematical evidence to detect patterns in the financial market.

Metrics

Metrics are an important way to enhance the efficiency of the product. It reduces ambiguity and makes for improved forecasting. All measurements are easily provided by the Python Scikit Library.

The parameters I have mentioned are explicitly specified and is the root mean squared error approach in the scikit collection. It is a simpler way of addressing the problem which specifically explains it. The LSTM is often tested using the backspreading methodology.

These methodologies are widely known and are seen in much of the latest problem-solving scenarios.

The key measures used are 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-

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

Y_{pred}-linear regression output

Y_{obs-}-Neural network output

n - total input

d/d(E)=gradient descent

II. Analysis

Data Exploration

In this report, the work is conducted on the Anaconda-supported Jupyter notebook. The analysis is housed in the.ipynb file. The data used were entirely in the date time format using the date time encoding method for possible predictions. Initially all features were correctly visualized, but later some features were raising for better performance.

The data set can be quickly retrieved from yahoo finance and is the most accurate data set available. To ensure dynamic data extraction, the data set is linked to the API.
Data set complexity allows to sustain effective project management. This is easily evident in the code line where the meaning of t is demanded.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.

- Even if the data set is not available, it will be impossible to determine everything about it. In this situation, it is important to build a data set using hadoop by analyzing the firm's database. The remaining functionality would then be used in the project following an in-depth review.

```
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
```

- The elimination of less important functionality is achieved using minmaxscaler.
- The data set is absolutely ideal in itself, because it contains nearly all the functionality required for successful estimation, such as the min and max parameters.
- Because the data forecast is based on the volume study, the graph and the tabular details are clearly given 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))
```

Date	High	Low	Open	Close	Volume	Adj Close
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

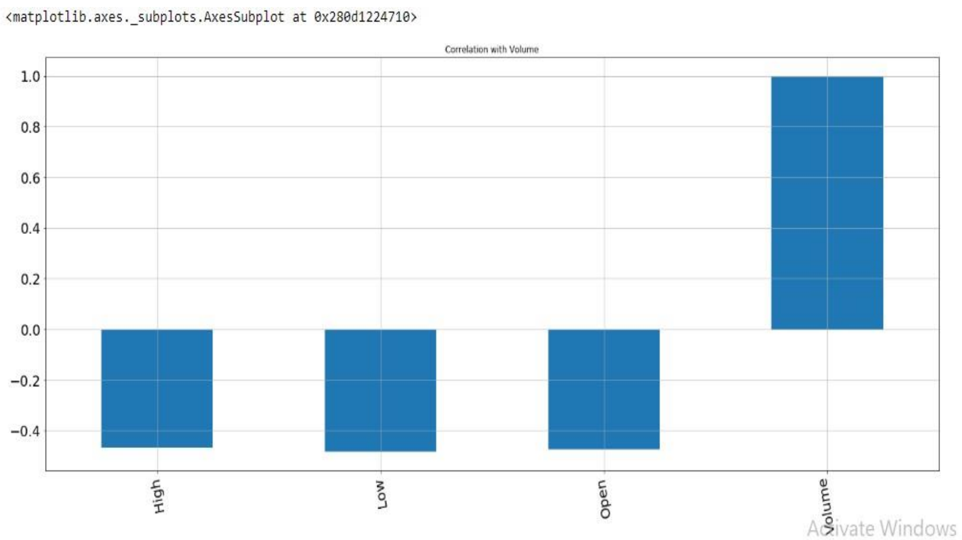
Date	High	Low	Open	Close	Volume	Adj Close
2018-08-16	65.669998	64.809998	65.040001	65.500000	3149500.0	65.067375
2018-08-17	65.480003	64.279999	65.480003	64.660004	2449200.0	64.232925
2018-08-20	65.110001	64.459999	64.570000	64.470001	1619900.0	64.044182
2018-08-21	65.400002	64.620003	64.739998	64.930000	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

Graph visualization is done using the python matplot module. Various types of graphs are used, such as histograms, dotted graphs and normal linear curves.

In the LSTM, the graph is seen in a non-linear format.

The comparison of histogram maps of various data types is seen at the beginning of the method. The following table indicates the volume of the drug sold at various times.



Algorithms and Techniques

Various machine learning algorithms are being used and, in particular, enhanced efficiency.

- One of the algorithms included is a linear regression and decision tree classifier that is an essential aspect of the supervised learning approach.
- Linear regression uses a continuous method of data for classification.
- The judgment method is valid for both classification and regression purposes.
- The definition of standardization is used extensively to delete elements with less important devices.
- The gradient descent algorithm reduces error every time.
- In the LSTM neural network, feed-forward and backspread are used to perform features and reduce errors.
- LSTM is a special neural network algorithm where the correction data is processed in a memory buffer.• Each of the functionality can be accessed from a scikit app repository.
- Keras has been included in the set of the LSTM.
- For the usage of linear regression and decision trees, the data set is first separated into training and testing parts.

- Assessment and monitoring of pa interventions shall be carried out.
- The neural network matrix data structure is separated into vectors and used as a perceptron.

Linear Regression — Linear regression is a statistical tool for estimating the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The state of an explanatory variable is referred to as simple linear regression.

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

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

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 test model was applied as a linear regression as discussed in the study. This experiment demonstrated how revolutionary methods are operating.

• The analysis explicitly indicates the RMSE value and the R2 value arising from the application of the linear regression.

• In the mathematical study of the experiment, the benchmark experiment offers an real observation while the other model offers a potential or projected style.

•Also an additional benchmark of decision tree regressor is used which is having better score.

III. Methodology

Data Preprocessing

Preprocessing data is a crucial strategy for identifying and interpreting data. It also allows the identification of important features and the removal of certain obsolete apps. It involves the translation of raw data into an open format. Real-world proof is frequently incomplete, inconsistent and/or lacking in certain behaviors or trends and is required to contain a number of mistakes. Pre-processing data is a proven method of resolving such issues.

The alternate solution to Z-score normalization (or standardization) is the so-called Min-Max scaling (often also simply named "normalization "- a frequent source of ambiguity).

In this method, the data is scaled to a set range-usually from 0 to 1.

The downside of getting this restricted range-in comparison to standardization-is that we may wind up with smaller standard deviations, which could be ignored.

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:

Long short term memory 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 aspect any new developer working more on this project will realize is that often this application does not function the same in project. Often there might be bugs with the program, such as any updated library or device issues that must be modified appropriately. For example, the version of the tensorflow module does not result in any outcome or mistake.

```
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
```

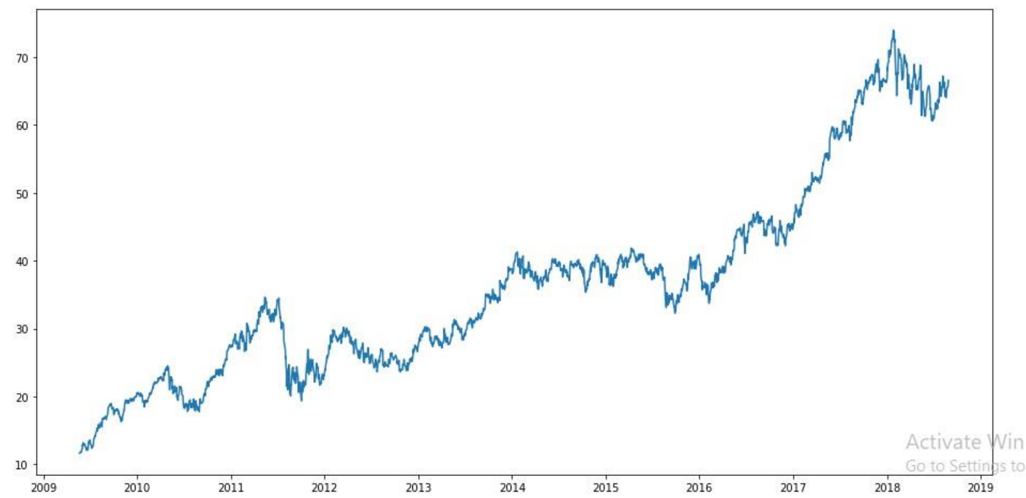
IV. Results

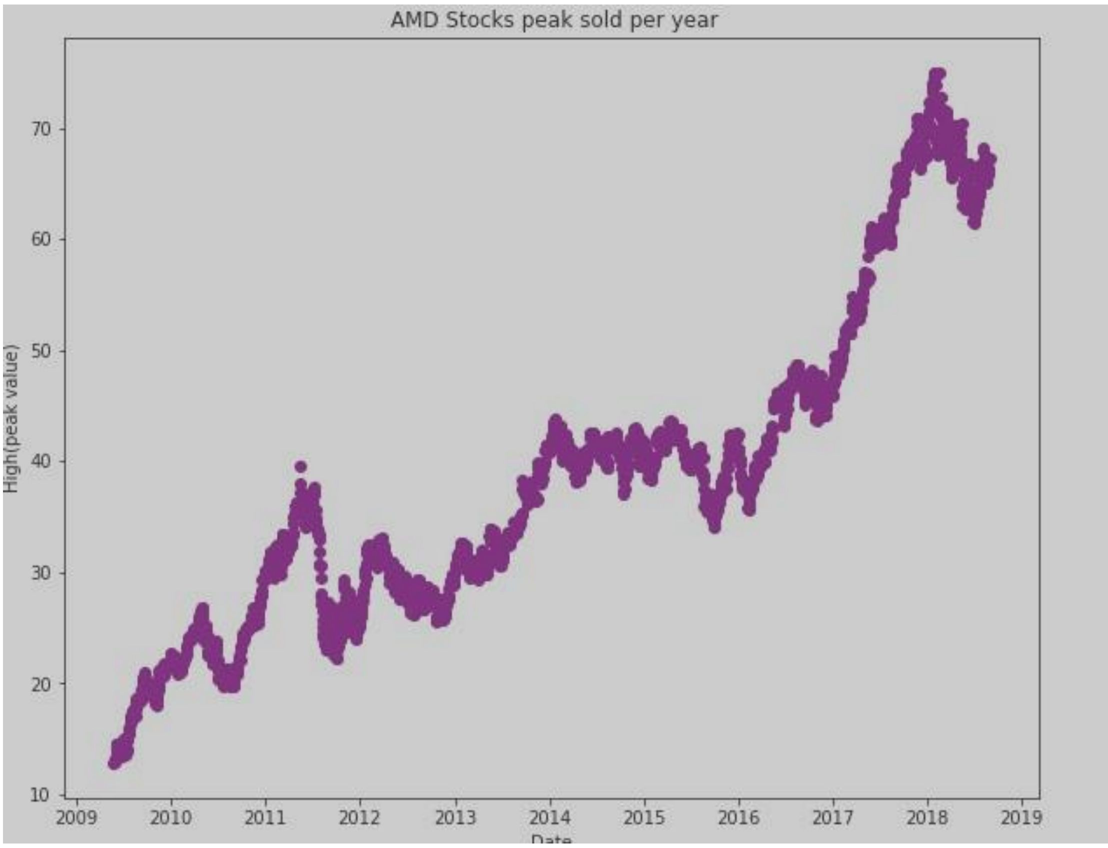
Model Evaluation and Validation

While this data set is useful for forecasting the usage of standard supervised learning techniques. However using LSTM gives it an upper hand and allows stronger predictions.

This model is extremely reliable for any form of estimation owing to its data set abstraction from the API. Nevertheless, the drawback is that this model may only be rendered online.
Original Graphing-a) This indicates the increase in stock output ea

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.

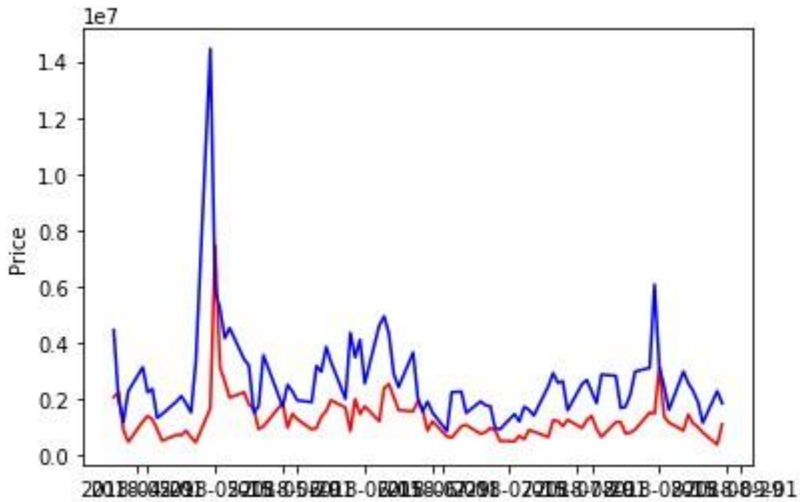
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

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
```

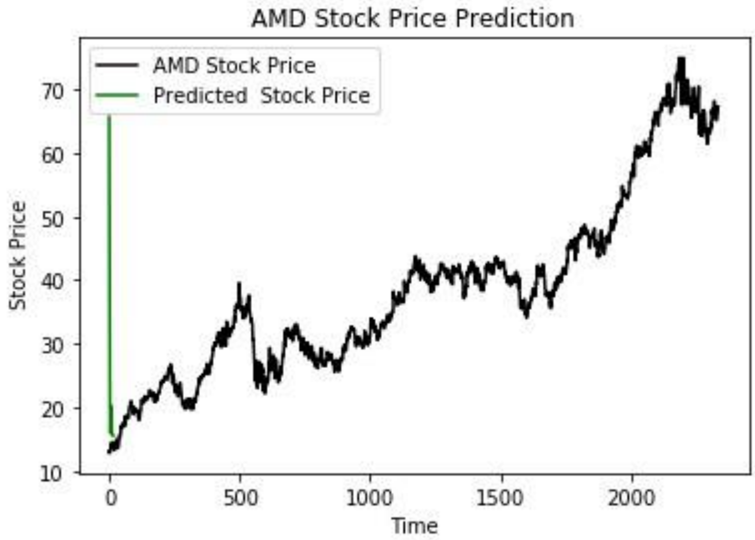
```
lstm= model_lstm.evaluate(X_tst_t, y_test, batch_size=1)
```

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

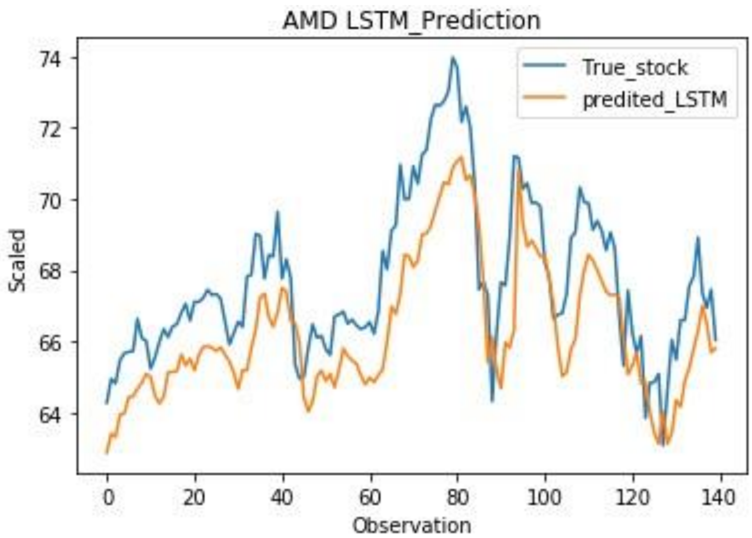
```
print('LSTM: %f'%lstm)
```

```
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.

Explaination

A) In this segment there is a contrast of the results of the linear regression and the decision tree, where the performance of the trainee is really strong for the decision tree(1.0) and, as a result, the model is not suitable 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

B)Long Short Term Memory analysis was found to be really good with good 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 always displays tremendous robustness. This shifts with no adjustment any time.

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

```
t
```

```
63.24946212768555
```

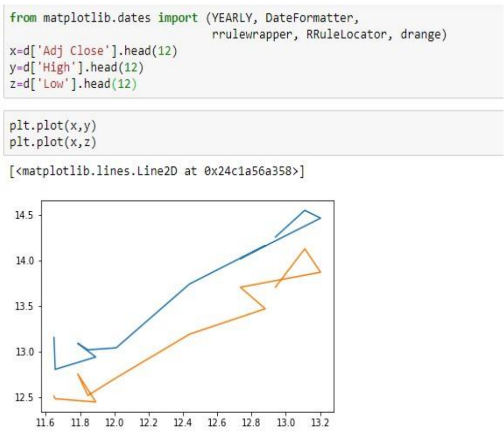
```
import random
ts=d['Adj Close']
t=random.choice(ts)
```

```
t
```

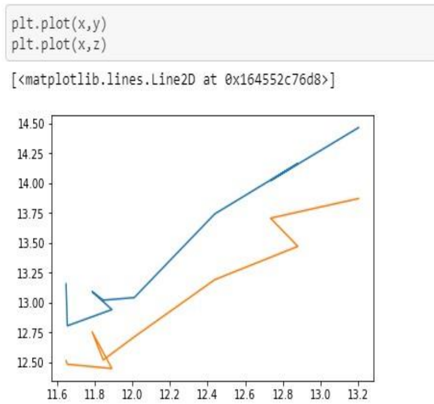
```
26.27754020690918
```

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

For 12 values



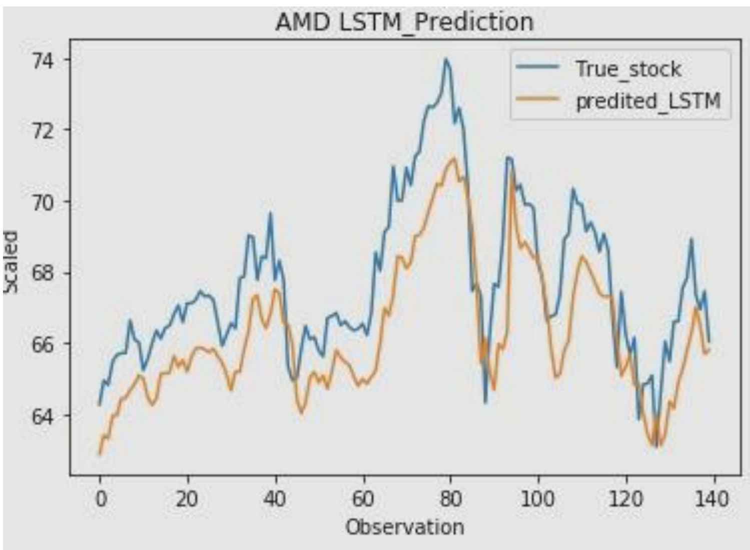
For 10 values



V. Conclusion

The research was concluded with the strongest forecast using different techniques.

This is virtually difficult to achieve 100% precision in the simulation of the market forecast. However the forecasted line may provide the trader with an investment concept.



Here the red line displays our forecast, which is about the same form as the blue one, which is accurate but varies a lot in scale and design. But it also gives investors a better idea.
It's good to see that our training data performs really well, but there are certain problems in data sets. It's also worth mentioning that much of the cases, the form is the same as the real and the expected one.

However, this model fits really well with linear regression, but trying other methods will yield newer tests.

New techniques, such as random forest or k-map, may also produce good / bad outcomes.

```
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
```