Anthony Habib 100662176
November 13, 2019

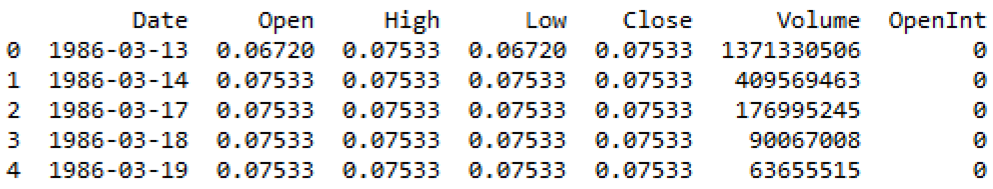
Machine Learning Final Project

1. Frame the problem:

- Predicting stocks with machine learning algorithms are flawed because stocks are can be unpredictable. The reason stocks are unpredictable are due to the price of a stock being influenced by somethings like the companies financials (how well or how bad they are doing at the end of a quarter), the opinion of people (does the company have good reputation or bad reputation), news or social media (Apple releasing a new iPhone will attract new buyers). In my final project, I use Linear Regression, ARIMA (Auto Regressive Integrated Moving Average), and LSTM (Long Short Term Memory) to predict future values based on previous values and how well our machine learning algorithm can study these past values to get an accurate prediction. The dataset I chose to use was the Microsoft Stock Market and in my dataset, it consists of 4 main columns: Open, High, Low, and closing which shows the stock prices opening price, the high price, the low price, and the closing price. I chose to predict the closing data for my dataset because the closing data is the most important for learning the stock due to the tracking of the stock which is why investors and traders can determine the performance of the stock. The Microsoft dataset values were being tracked since 1986-03-13 (March 13th) and end on 2017-11-10 (November 10th). In conclusion, I use two machine learning algorithms (Linear Regression and ARIMA), and one neural network (LSTM) to learn 26 years of data (2013 – 1987) and predict 4 years of data (2013 – 2017) accurately.

2. Explore the Data and gain insights:

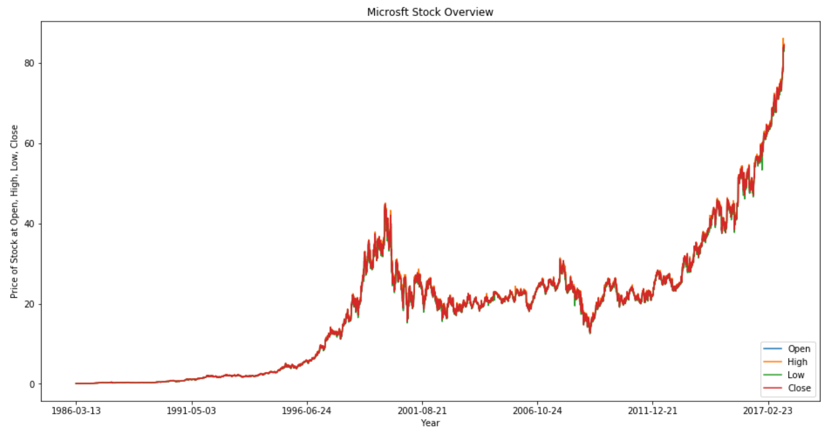
- My data was organized perfectly and has no missing or na values, so, fortunately, I did not have to clean the dataset.



```
         Date     Open     High      Low    Close      Volume  OpenInt
0  1986-03-13  0.06720  0.07533  0.06720  0.07533  1371330506        0
1  1986-03-14  0.07533  0.07533  0.07533  0.07533   409569463        0
2  1986-03-17  0.07533  0.07533  0.07533  0.07533   176995245        0
3  1986-03-18  0.07533  0.07533  0.07533  0.07533    90067008        0
4  1986-03-19  0.07533  0.07533  0.07533  0.07533    63655515        0
```

Anthony Habib 100662176
November 13, 2019

- Here, I plot my columns, open, high, low, close to display what my data looks like plotted and get an idea on how my training and testing data will look like.

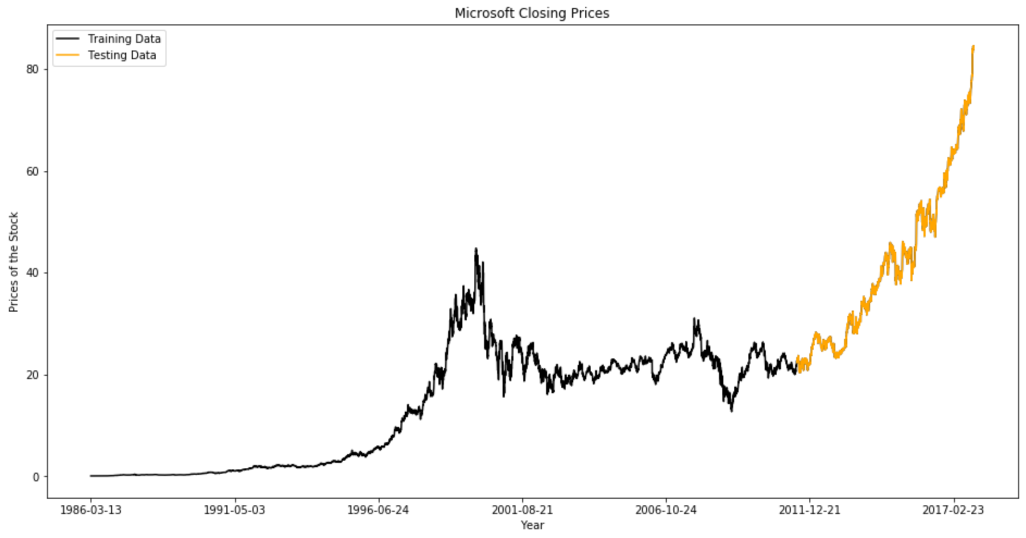


- Here, I can visualize the training and testing values

```
train_data_c = training['Close'].values
test_data_c = testing['Close'].values
```

```
training [ 0.07533  0.07533  0.07533 ... 22.598   22.724   22.478  ]
testing [22.4    22.478 22.342 ... 84.56  84.09  83.87 ]
```

- In this graph, I show that I split my data into 80% training and 20% testing.



3. Prepare the Data

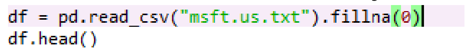
- Preparing Linear Regression
    - Make sure that if we have any missing values we fill it with 0

```
df = pd.read_csv("msft.us.txt").fillna(0)
df.head()
```

    - Dates need to be converted and then later decoded

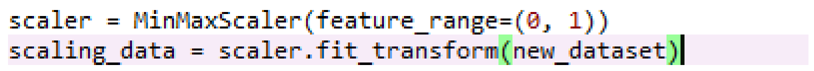
```
# dates need to be encoded
le = preprocessing.LabelEncoder()
df['Date'] = le.fit_transform(df['Date'])

# dates need to be decoded to be displayed
df['Date'] = le.inverse_transform(df['Date'])
```

- Preparing ARIMA
  - Make sure that if we have any missing values we fill it with 0



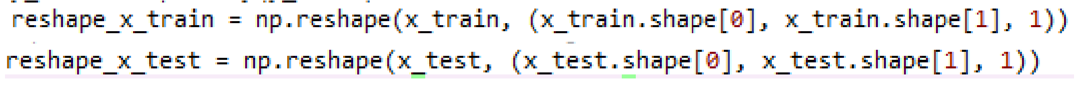
```
df = pd.read_csv("msft.us.txt").fillna(0)
df.head()
```

- Preparing LSTM
  - Scaling our data



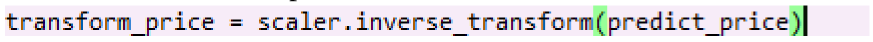
```
scaler = MinMaxScaler(feature_range=(0, 1))
scaling_data = scaler.fit_transform(new_dataset)
```

  - Reshapes our array



```
reshape_x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
reshape_x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

  - Inverse transform our predicted data



```
transform_price = scaler.inverse_transform(predict_price)
```

4. Machine Learning Algorithms:

- Linear Regression – regression was great to implement on this dataset. Plotting my predicted linear regression was done by dropping my predicting value (close) so that I can use my open, high, low, and volume indices to predict my close value. The performance of Linear Regression is fast and accurate, but isn't our best model used for our dataset.

Linear Regression Performance Table

Performance Measure	Value
MSE (average square between raw vs predicted)	0.5132385168216383
MAE (difference between raw vs predicted)	0.15450631824646446
RSME (rooted square between raw vs predicted)	0.5132385168216383
Elapsed Time (time took to run)	0.6981396675109863 seconds

- ARIMA – is the process of taking in past values to predict future values. 3 main parameters need to be filled when predicting our future values.
  - P = past values used to predict next values
  - Q = past errors to predict future
  - D = order of differencing
- In the dataset we use our close to predict close. The AR in ARIMA refers to AutoRegression which regresses a variable on past values of itself.
- The performance of ARIMA compares to be my best model but ultimately took the most time. I could substitute different testing and training values to reduce the time but this might affect the performance of the algorithm.

ARIMA Performance Table

Performance Measure	Value
MSE (average square between raw vs predicted)	0.3373808802323729
MAE (difference between raw vs predicted)	0.3876097590981285
RSME (rooted square between raw vs predicted)	0.3373808802323729

Elapsed Time (time took to run)	294.4608640670776 seconds (5 minutes)

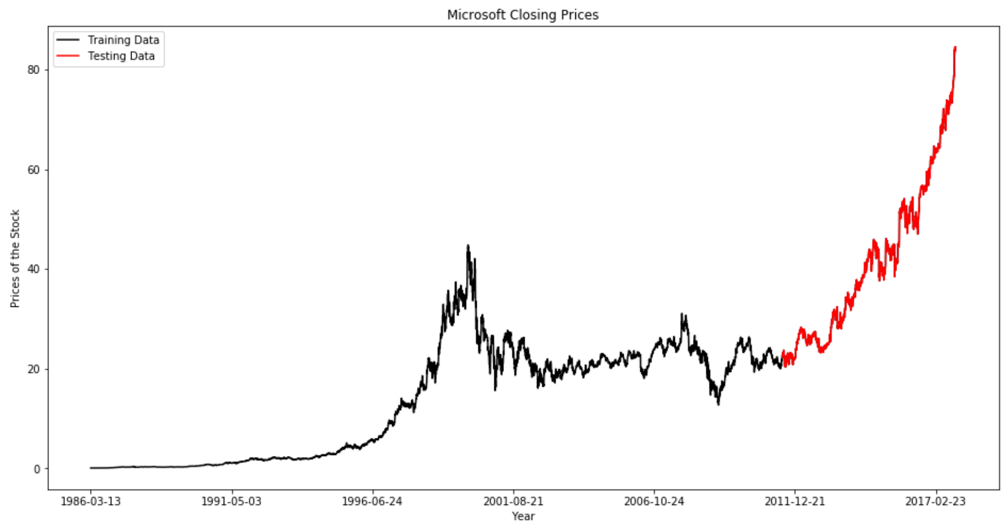
- LSTM – Long short term Memory (LSTM) is widely used to predict data. LSTM is effective because it stores older data that is important and forgets data that is not important. LSTM uses 3 important parameters like ARIMA.
    - Input gate = adds information
    - Forget gate = removes information that is not needed
    - Output gate = selects the important information to be displayed
- The performance of LSTM was great although it was my worst performing algorithm. I think this was due to the amount of epochs and time I let it run. I have tested higher epochs and the results were just as good as linear regression with MSE of 0. 762.
- Epoch of 5 and batch size of 32, my performance table shows higher values
- When I tested it epoch of 20 and batch size of 32, I was getting low scores of <0.8

LSTM Performance Table

Performance Measure	Value
MSE (average square between raw vs predicted)	4.509396263350358
MAE (difference between raw vs predicted)	1.4805798794741025
RSME (rooted square between raw vs predicted)	2.123533909159531
Elapsed Time (time took to run)	42.901623010635376 seconds

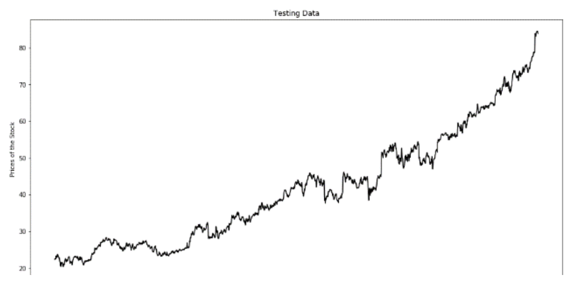
Judging by the MSE, MAE, and RSME, the values that are closest to 0 are seen as the best model. Therefore, the model that is chosen to be the best is **ARIMA**.

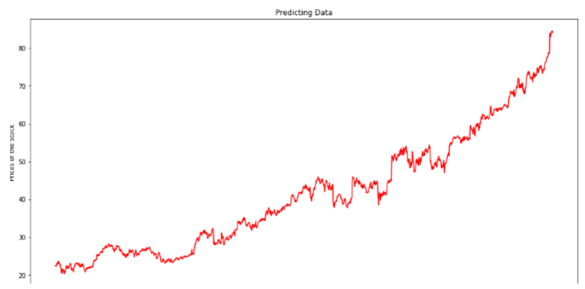
5. **Best Performing Algorithm:**
    - ARIMA Raw Overview of Data – I display and train ARIMA with 80% (6386 values) and 20% (1596 values)

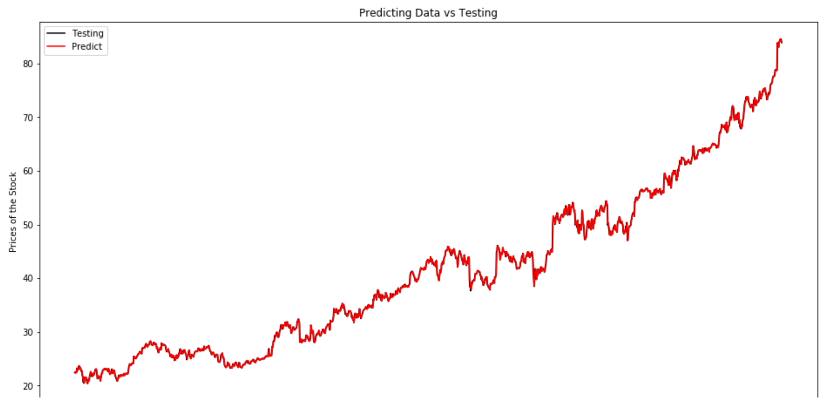


Anthony Habib 100662176
November 13, 2019

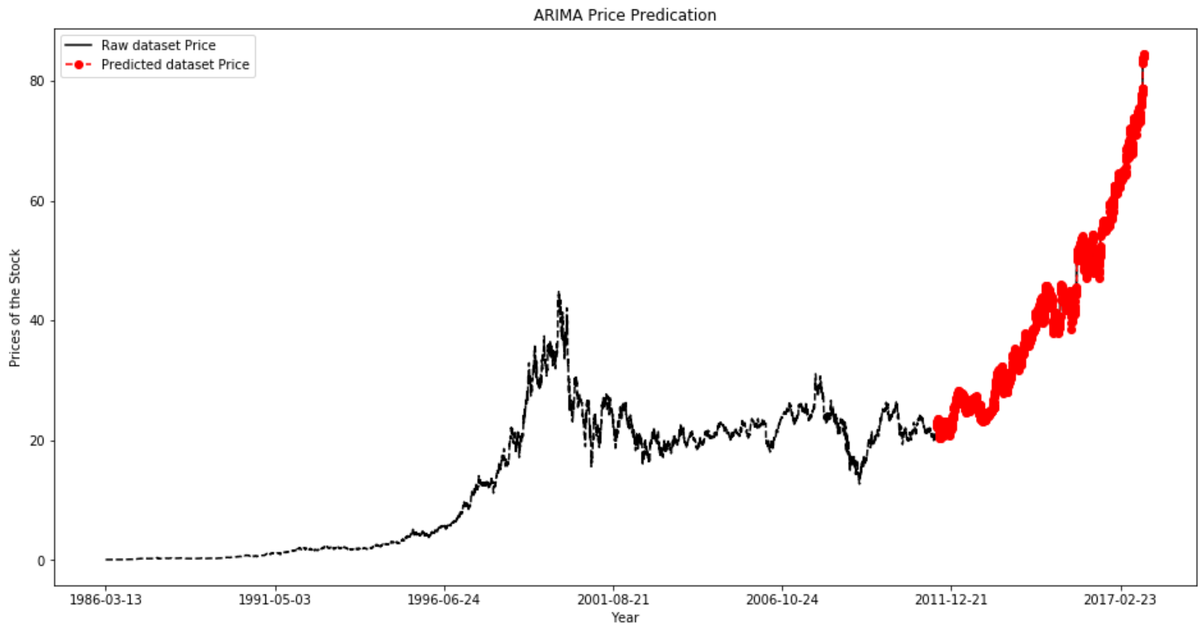
- ARIMA Raw Testing Data vs ARIMA Predicting Testing Data – I displayed a closer up image to analyze what the testing and prediction data look like. Since I thought they were the same, I had plotted the testing data against the predicting data to see that ARIMA prediction were very accurate in my case.







- ARIMA Predicting Overview of Data – An overview of my dataset is displayed and the predicted dataset is labelled with a thick marker indicating what it has predicted as my future values.


ARIMA Price Predication
Raw dataset Price
Predicted dataset Price
Prices of the Stock
Year