Udacity Machine Learning Nanodegree Program – Final Capstone Project Report

Stock Price Prediction Using Machine Learning

Shib Shankar Golder

August 9, 2018

Quotes

Warren Buffett ***- “****Risk comes from not knowing what you are doing.”*

Peter Sondergaard ***–*** *“Information is the oil of the 21st century, and analytics is the combustion engine.”*

Peter Lynch ***-*** *"Know what you own and know why you own it."*

Table of Contents

1. Definition

1.1. Project Overview

Someone rightly said that "Investing on just saving account, will not make one millionaire". Well, this statement seems very true but then the risk is higher if one doesn't take calculated risk for a profitable investment. With the advancement of Cloud technology and compute power, one can easily store and process huge data instantly in real time. There are several financial organizations (Bank, Broker, Fund Manager, Investment firms etc.) who usually trade on very weak correlations that are uncovered due to research from a quantitative analyst (also known just as a "quant") who is convinced of the validity of the correlation. While these correlations are weak, the scale at which these quant trading firms operate can make each of these individual strategies worth hundreds of thousands, millions, or even more. Now, the question is, how they are achieving to maximize the profit? Can they predict the Stock? If I ask these same questions, 10-20 years back then the answer would have been very difficult.

There are several organization (Two Sigma Investments, D. E. Shaw (company), Renaissance Technologies (hedge fund), and Hudson River Trading) who have come up with different trading strategies based on several years of experience and have patented these secret trade strategies as IP. Since, their strategies are optimized for speed and reliability, sometime the machine learning techniques they use are usually very simple. Additionally, because of this ruthless competition for profitable trades, quant firms are incredibly secretive and protective of their intellectual property. Now, the success rate has increased many folds because of automating these trade strategies with Data Science and Machine Learning. In recent days, there are several AI powered trading bots used in this business to do trading successfully.

This project depicts a simple way to showcase the power of machine learning mainly Deep Learning RNN (Recurrent Neural Network) in trading business. Here, one can learn few simple ways to forecast the Closing price of Amazon based on its historical trade data. Here, both LSTM and GRU have been used to forecast the closing price. For simplicity sake, Keras using Tensor flow has been used to build the RNN models.

This project has visualized the different technical features used along with charts to showcase the closing price with the trading days.

1.2. Problem Statement

In this project, the main problem is to predict the stock closing price based on the knowledge of historical trade data. There are several machine learning algorithms out there in market which can be leveraged to calculate the closing price. Here, RNN specially LSTM and GRU (known to be very effective and perform well on continuous time series data) has been leveraged to calculate the price.

As the stock data is continuous and we are trying to predict the price, hence, this is a classic example of regression i.e. y=a0+a1*x. In higher dimensions when we have more than one input (x), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g. a0 and a1 in the above example).

In higher dimensions when we have more than one input (x), the line is called a plane or a hyper- plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g. a0 and a1 in the above example).

In this case, Y >> Stock Closing Price (final label) and Xi >> The different technical (Adj.Open, Adj.High, Adj. Close etc.). Our models will be trained based on historical stock data imported from Quandl data sets. Then, compare the results of the models, visualize the results and conclude based on the findings.

1.3. Metrics

In this project, the model evaluation metrics will be measured using Mean Square Difference & Root Mean Square ($RMSE = \sqrt{\sum{(Y_{actual} - Y_{predicted}) ^2} / n}$) for the predicted verses the actual. And, then both GRU and LSTM model would be compared based on the accuracy than that of the Regression model. Will visualize the actual price with the predicted price based on the learning.

---

https://www.quora.com/Can-machine-learning-algorithms-models-predict-the-stock-prices-If-yes-which-are-the-best-machine-learning-algorithm-models-to-predict-the-stock-prices

https://www.microsoft.com/developerblog/2017/12/04/predicting-stock-performance-deep-learning/

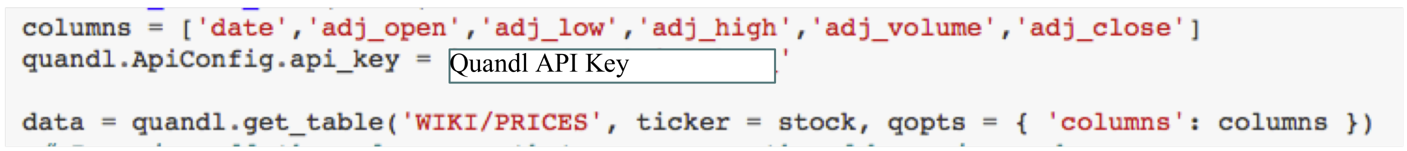
https://medium.com/@vivekvyas_24202/stock-market-predictions-using-machine-learning-d03098dd1957

2. Analysis

2.1. Data Exploration

As discussed before, this project uses Amazon's stock index to measure the model performance. One of the reasons of choosing this index as it price increased very fast in last 4-5 years. I wanted to see, how the model can learn the trade data effectively and the predict the rising peak. I have used Quandl python module to download the stock historical data. The prime reason to use Quandl as the data is mostly neat and clean and removed the non-trading days.

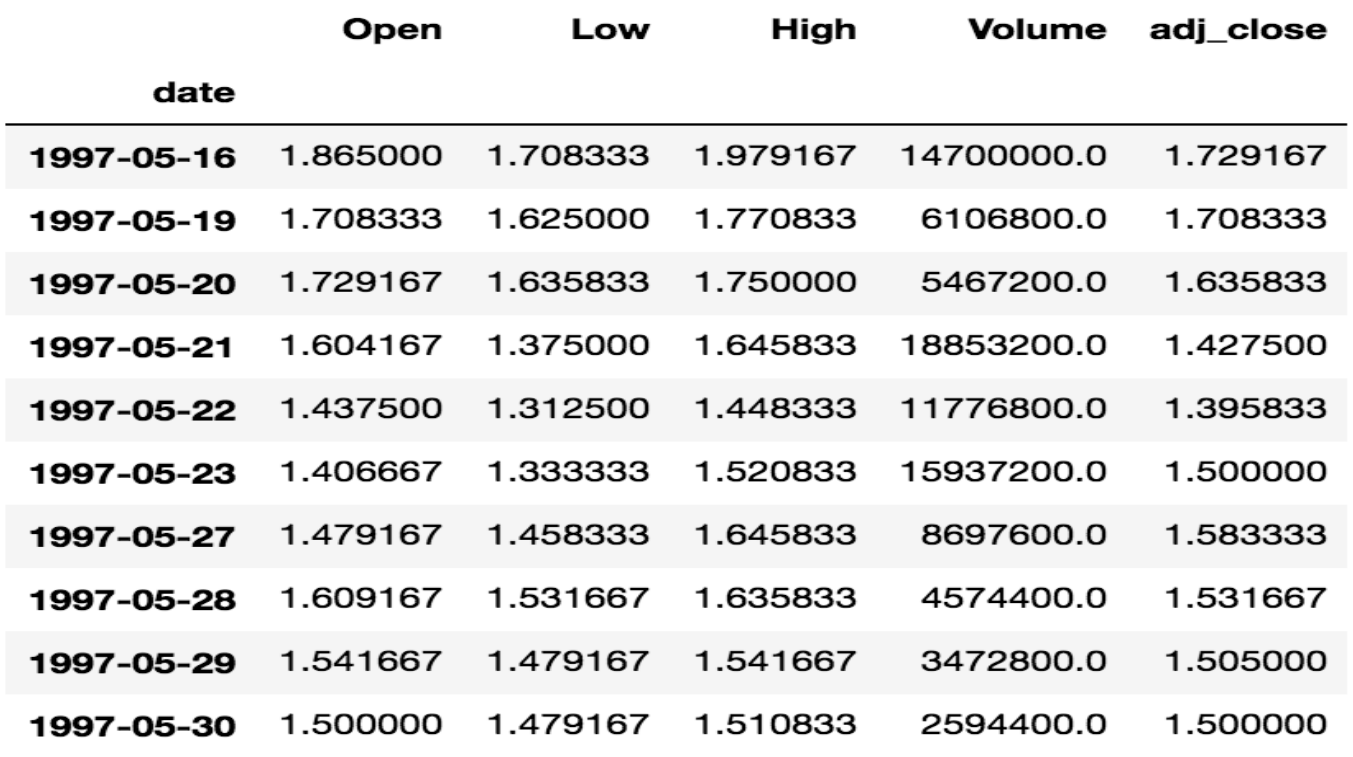
Below is code snippet on how easy to download the day in needed python data frame format.



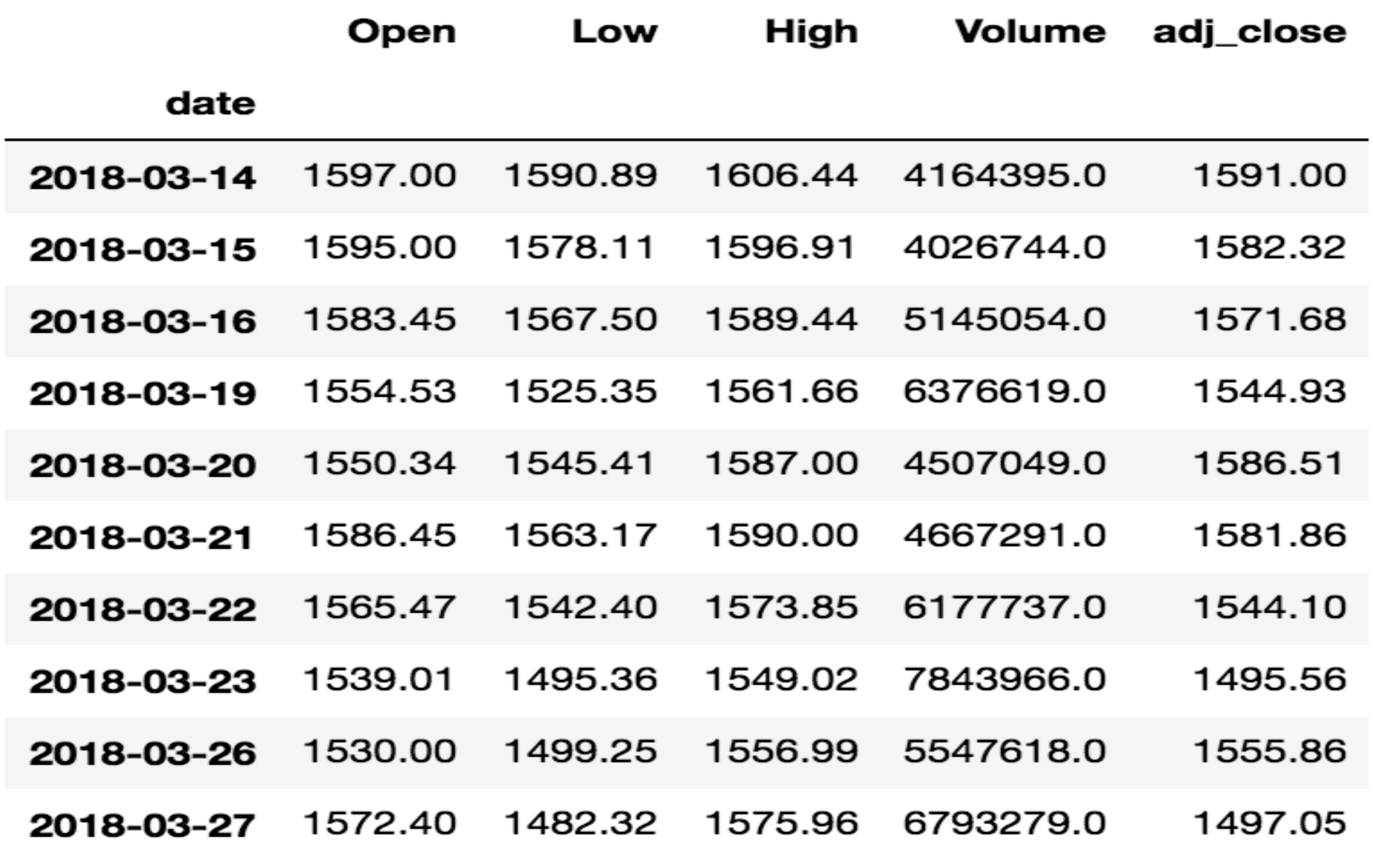
```
columns = ['date','adj_open','adj_low','adj_high','adj_volume','adj_close']
quandl.ApiConfig.api_key = Quandl API Key'

data = quandl.get_table('WIKI/PRICES', ticker = stock, qopts = { 'columns': columns })
```

As, one can see the stock data is retrieved by just using 1-2 lines of code. And, in this project, I am focused on few technical features like date, adj_open, adj_low, adj_high, adj_volume and adj_close to train the model for simplicity sake.

I have downloaded all the historical stock data for Amazon since its inception till March 2018 (around 20 years). Below is the initial head snapshot of the data.



	Open	Low	High	Volume	adj_close
date					
1997-05-16	1.865000	1.708333	1.979167	14700000.0	1.729167
1997-05-19	1.708333	1.625000	1.770833	6106800.0	1.708333
1997-05-20	1.729167	1.635833	1.750000	5467200.0	1.635833
1997-05-21	1.604167	1.375000	1.645833	18853200.0	1.427500
1997-05-22	1.437500	1.312500	1.448333	11776800.0	1.395833
1997-05-23	1.406667	1.333333	1.520833	15937200.0	1.500000
1997-05-27	1.479167	1.458333	1.645833	8697600.0	1.583333
1997-05-28	1.609167	1.531667	1.635833	4574400.0	1.531667
1997-05-29	1.541667	1.479167	1.541667	3472800.0	1.505000
1997-05-30	1.500000	1.479167	1.510833	2594400.0	1.500000



	Open	Low	High	Volume	adj_close
date					
2018-03-14	1597.00	1590.89	1606.44	4164395.0	1591.00
2018-03-15	1595.00	1578.11	1596.91	4026744.0	1582.32
2018-03-16	1583.45	1567.50	1589.44	5145054.0	1571.68
2018-03-19	1554.53	1525.35	1561.66	6376619.0	1544.93
2018-03-20	1550.34	1545.41	1587.00	4507049.0	1586.51
2018-03-21	1586.45	1563.17	1590.00	4667291.0	1581.86
2018-03-22	1565.47	1542.40	1573.85	6177737.0	1544.10
2018-03-23	1539.01	1495.36	1549.02	7843966.0	1495.56
2018-03-26	1530.00	1499.25	1556.99	5547618.0	1555.86
2018-03-27	1572.40	1482.32	1575.96	6793279.0	1497.05

All the columns except Date (indexed column) are adjusted. There are several other columns available for download but for now I am interested on these for consistency sake and to avoid unnecessary values.

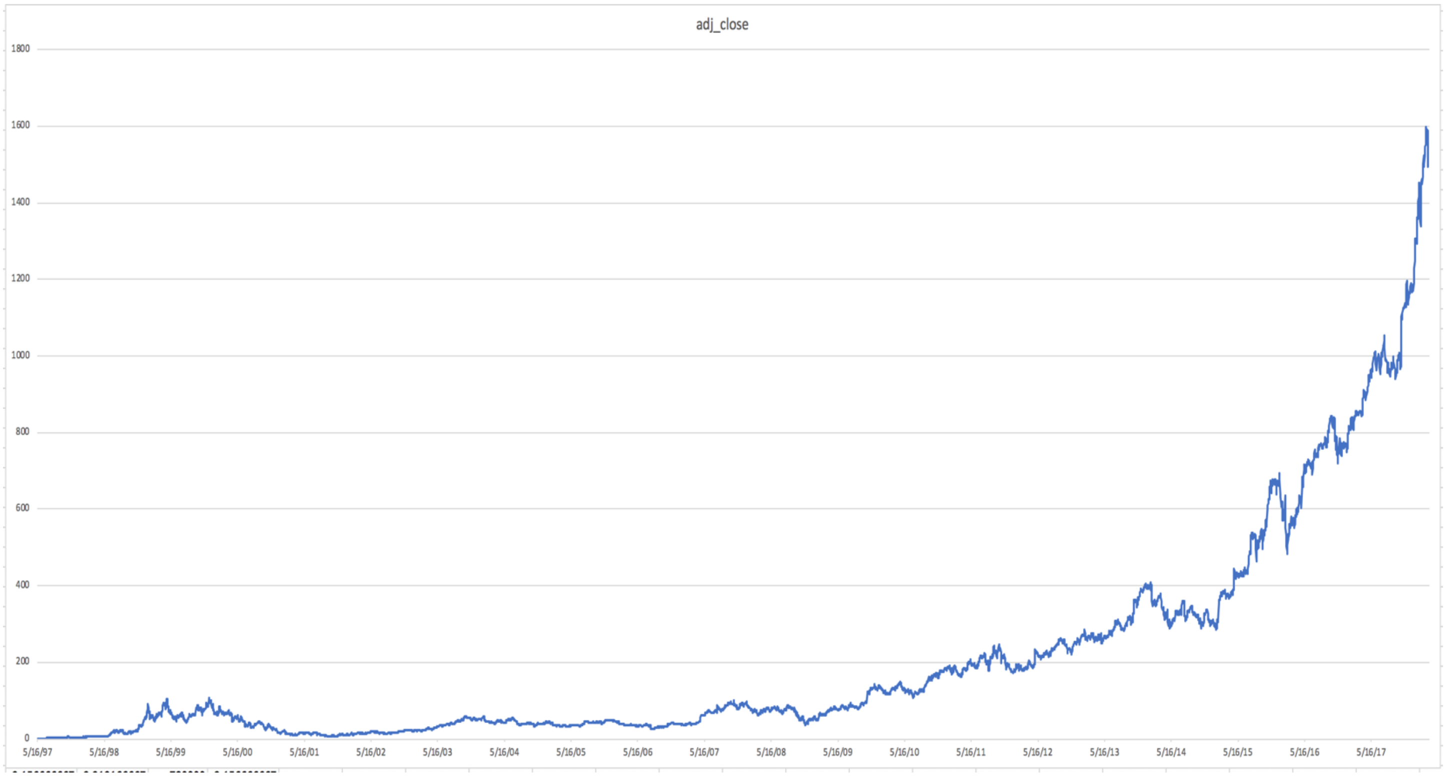
From the above data set, we have a good variety of historical information of Amazon's performance. In our model, our main intension would be to predict the closing price. And, there can be several strategies to like intra-day (very famous but has regulatory concern to maintain liquidity) like buy at day's low and sell at day's high and then Day-in and Day-out i.e. enter at day's open and sell at Day's close. In our case, I am mostly focus on Adj_close. So, technical, Open and Volume make more sense which can impact the day's close. This should cover several key trading factors for each day e.g. sudden drop or sudden high, earnings etc.

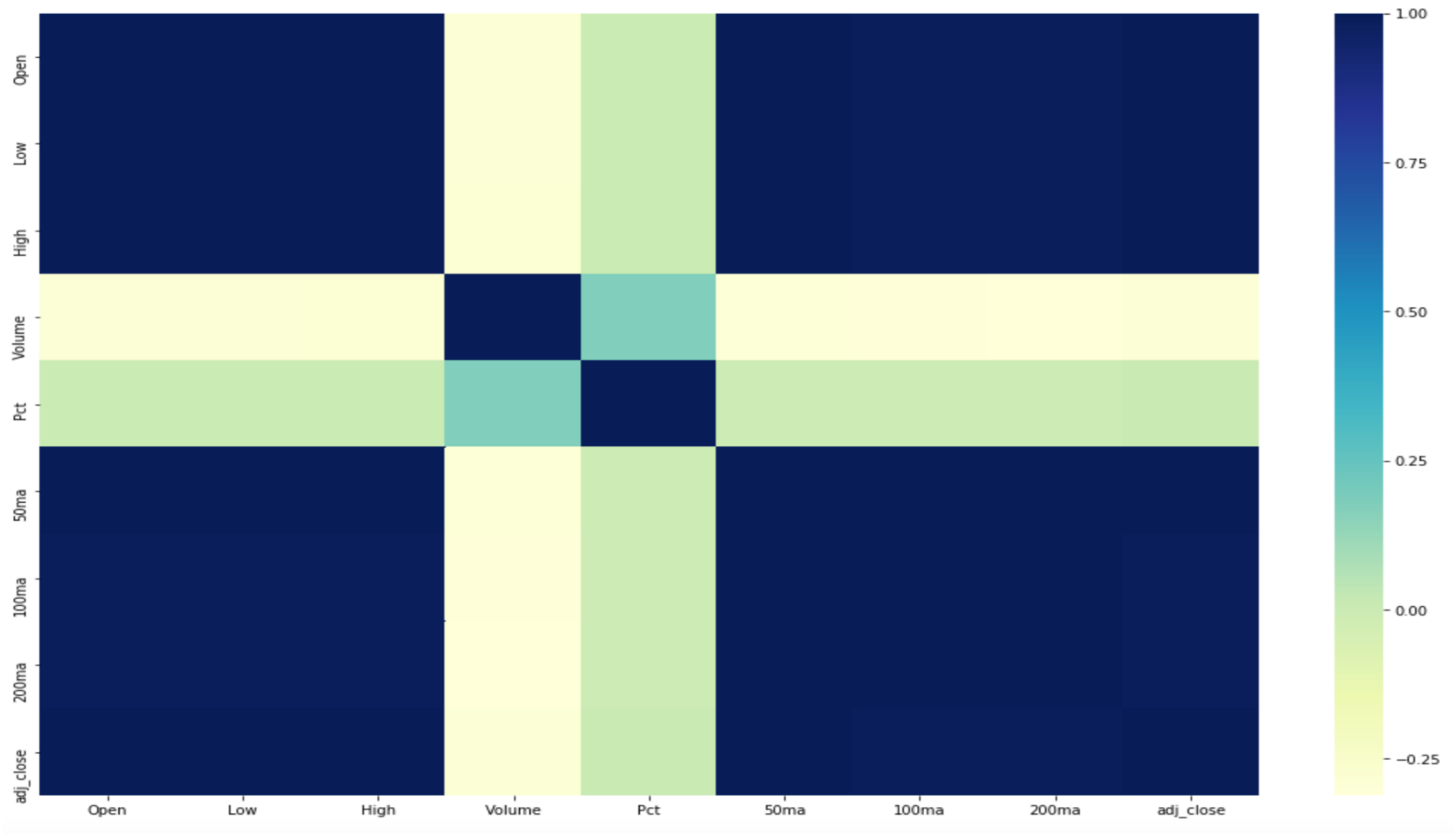
Below are Mean, Standard deviation, Max and Min of the whole data set. The min and max are far away from Mean and hence signifies the stock is volatile.

	Mean	STD	Max	Min
Open	201.48	281.84	1615.96	1.41
Low	198.96	279.15	1590.89	1.31
High	203.82	284.02	1617.54	1.45
Close	201.50	281.75	1598.39	1.40
Volume	7803634.29	7493733.37	104329200.00	487200.00

2.2. Exploratory Visualization

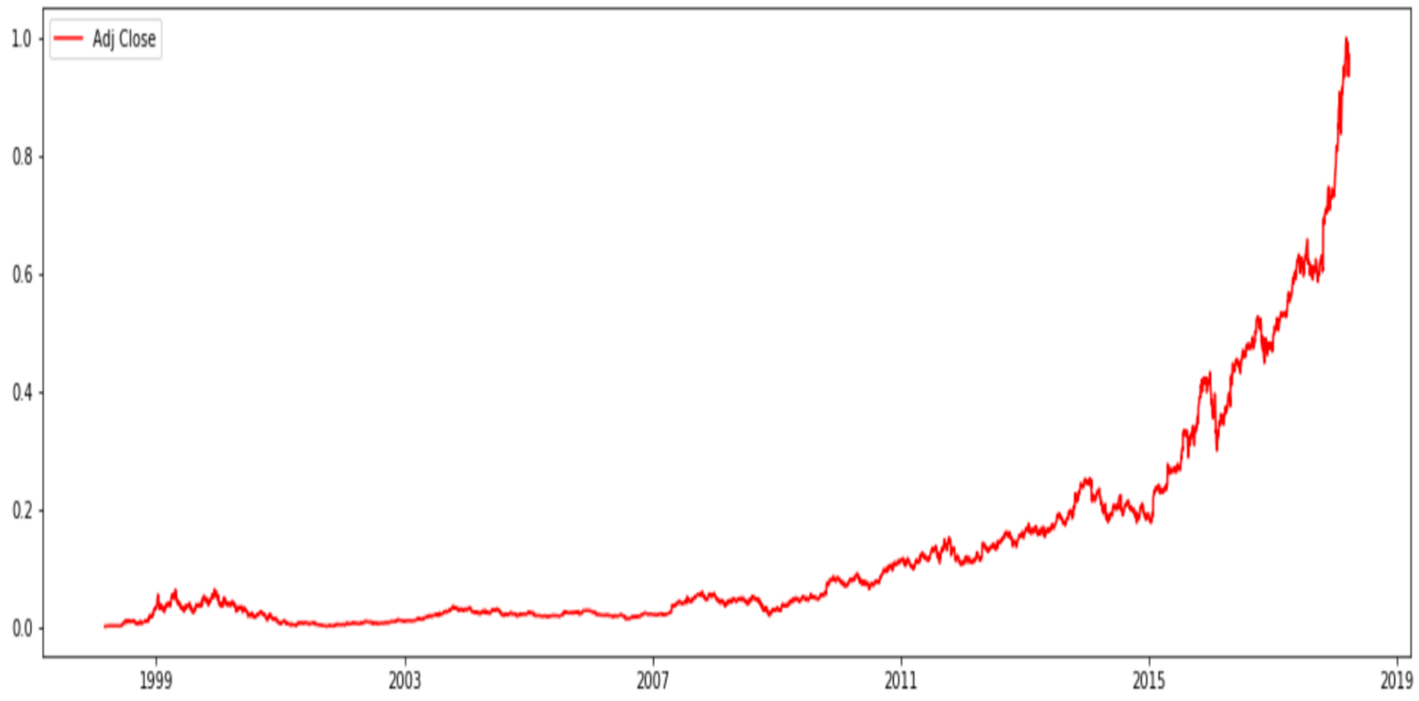
Thought the project, I have use matplotlib and seaborn library to visualize the data set. Below are figures with actual, normalized, percentage change and feature correlation charts.

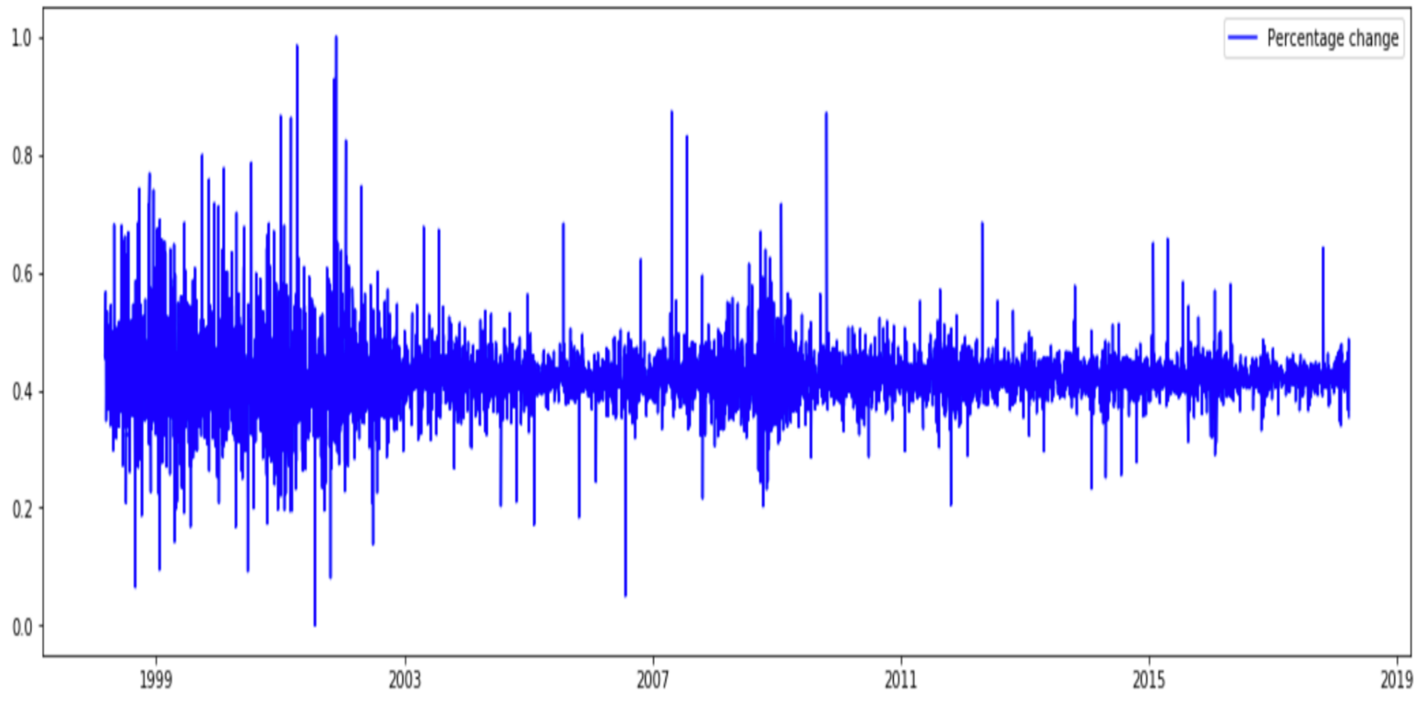




During mid 1998, the stock had a good rise till 2000 (Dot Com period) it started falling then went into a stead rise till 2008 financial crisis and stock has increased rapidly many folds after 2016. From PCT chart, we can see the stock have high volatility, Dot Com period uncertainty at the beginning then during 2008 financial crisis, which make as the short players went for more intraday trading and institutional investor might have stayed quiet.

Normalized Charts



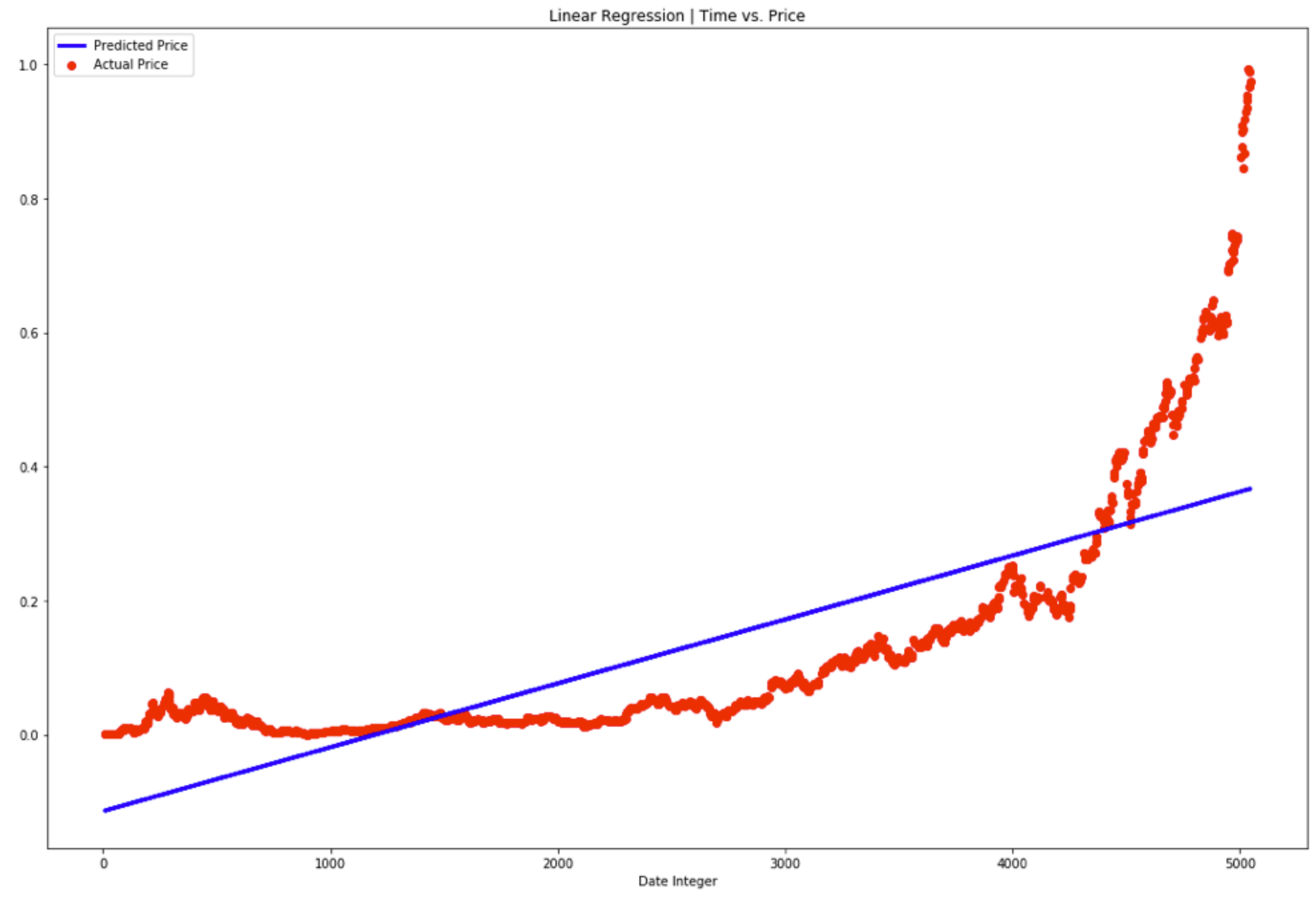


2.3. Algorithms and Techniques

In this project, I would leverage the Recurrent Neural Network (RNN) namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) as the model architecture. I will use all the relevant technical features for training these advanced models to showcase its capability. As the data set is continuous time series data, that's why I will leverage the sequence to better split of the data set. The full historical data will be preprocessed and normalized using MinMaxScaler before being fed to the model for learning. I will train the model using the training set and then will evaluate the model performance on the testing set and will measure the errors. I will plot the model's prediction on the chart and measure the error rate for each. Then, finally will tune the model so as gain max performance using hyper-parameters.

2.4. Benchmark

The benchmark model of this project is Linear Regression as it would be best fit for the continuous time series data. Below the plot chart of the predicted vs actual Closing price.



As you can see, the linear regression model did a fair job in having a constant upward trend r.w.t the actual price based on the trading days. As one can see the model predicted below the actual for the 1000 days than it actually predicted more than the actual but after 4500 days it cannot map the exponential upward trend.

Below is the error rate MSE and RMSE for regression model for training and testing set:

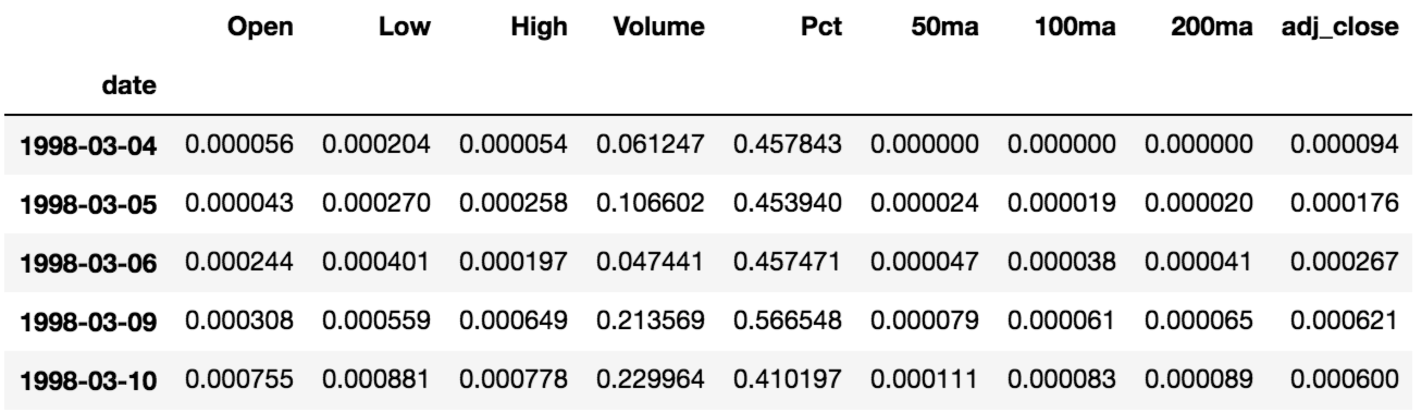
```
Train Score: 8534412.4325 MSE (2921.3717  RMSE)
Test Score:  0.01290567   MSE (0.11360313 RMSE)
```

3. Methodology

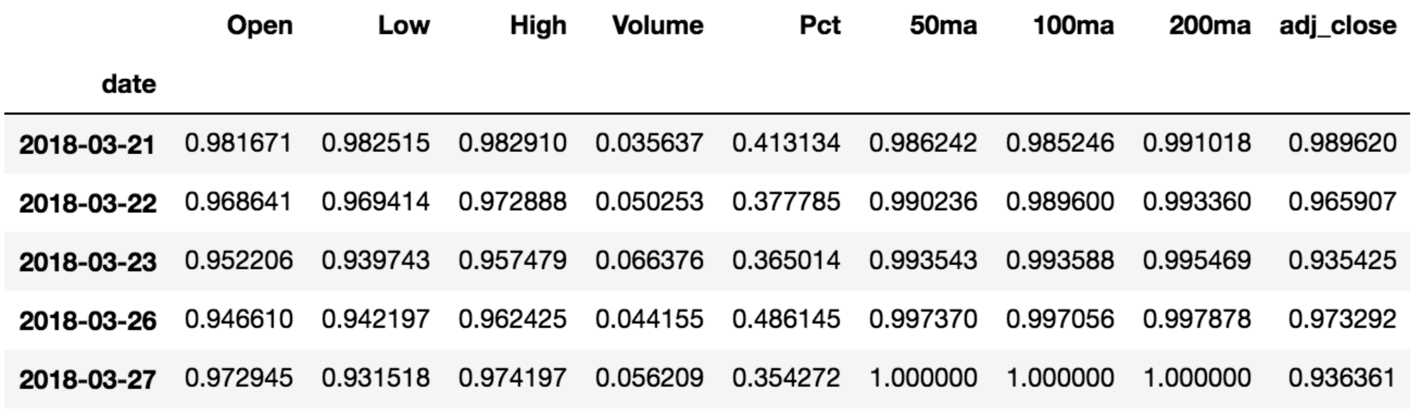
3.1. Data Preprocessing

There are various important parts for this data preprocessing namely technical feature selection, create more important feature, normalization using min-max scaler and removed the null values where ever possible. Then, changed the date into days of trade for ease of processing.

Before normalizing the data, I have created few more technical features namely Pct (% Change), Moving averages.



date	Open	Low	High	Volume	Pct	50ma	100ma	200ma	adj_close
1998-03-04	0.000056	0.000204	0.000054	0.061247	0.457843	0.000000	0.000000	0.000000	0.000094
1998-03-05	0.000043	0.000270	0.000258	0.106602	0.453940	0.000024	0.000019	0.000020	0.000176
1998-03-06	0.000244	0.000401	0.000197	0.047441	0.457471	0.000047	0.000038	0.000041	0.000267
1998-03-09	0.000308	0.000559	0.000649	0.213569	0.566548	0.000079	0.000061	0.000065	0.000621
1998-03-10	0.000755	0.000881	0.000778	0.229964	0.410197	0.000111	0.000083	0.000089	0.000600



date	Open	Low	High	Volume	Pct	50ma	100ma	200ma	adj_close
2018-03-21	0.981671	0.982515	0.982910	0.035637	0.413134	0.986242	0.985246	0.991018	0.989620
2018-03-22	0.968641	0.969414	0.972888	0.050253	0.377785	0.990236	0.989600	0.993360	0.965907
2018-03-23	0.952206	0.939743	0.957479	0.066376	0.365014	0.993543	0.993588	0.995469	0.935425
2018-03-26	0.946610	0.942197	0.962425	0.044155	0.486145	0.997370	0.997056	0.997878	0.973292
2018-03-27	0.972945	0.931518	0.974197	0.056209	0.354272	1.000000	1.000000	1.000000	0.936361

I have divided the Test and Training as the below split for the various Model

- **Linear Regression:** Training 70% and Testing 30 %

```
    o  ('x_train', (3532, 1))
    o  ('y_train', (3532, 1))
    o  ('x_test', (1515, 1))
```

```
o  ('y_test', (1515, 1))
```

➢ **LSTM & GRU:** Training ~90% and Testing ~10%

```
o  Amount of training data = 4522.5
o  Amount of testing data = 502.5
```

3.2. Implementation

In the actual Jupiter notebook, there are namely three sections as follows:

➢ **1st Section:** Benchmark Model – Linear Regression
➢ **2nd Section:** LSTM Model & Optimization
➢ **3rd Section:** GRU Model & Optimization

And, each section follows similar steps/flows from starting till its end. Following are the different steps for the implementation:

1. Feature Selection
2. Data Preprocessing and cleansing
3. Data Split
4. Design the model architecture
5. Build the model
6. Training the model
7. Prediction based on testing set.
8. Errors Measurement
9. Chart/Visualization of Actuals vs Prediction
10. Tuning the hyper-parameters & Optimized Charts

I will not go through the code as it is pretty much self-explanatory from the notebook based on the aforesaid structure, but I try to discuss about the different sections in general.

The overall project was done using Keras module with Tensor flow as backend. For this project, I have leverage the Anaconda 2.7 macOS version and then loaded the aforesaid modules in the runtime environment. During my exercise, I have used input features as Trading days as **Item, Low, High, Volume, Pct, 50ma, 100ma, 200ma.**

➢ **1st Section:** Benchmark Model – Linear Regression:
In this section, I have used the **Close** as label and **Item (**trading days**)** as the input. Then, used the **sklearn's train_test_split** function to split data into testing and training sets. I wanted to keep the benchmark model simple so as to depict the level of complexity we can go during the further optimization. Then, used the **sklearn's LinearRegression** to fit the model with **X_train, y_train** which constitute 70% of the data set. Then, used the same model to predict using **X_test** for the prediction part.

- **2nd Section:** LSTM Model & Optimization:
  In the LSTM model, all the 9 features are being used as input to the model. The data set was divided into 90/10 ratio for training and testing set. For the LSTM, the initial model has 4 layers (**1 sequential, 2 hidden LSTM & 1 Dense node**) with **"Linear" activation function** as the model architecture. The hidden LSTM nodes has **256 and 128 neurons** with **shape (22, 9)** respectively. I have use **"MSE"** as the loss function with **"Adam"** as the Optimizer. The initial model was trained for 1 epoch, with default batch size and having 0.2 as the validation set.

  In the optimized model, I have added a **Dropout of 0.2** between each layer, **batch size of 100** and trained it for **epoch 20.**

- **3rd Section:** GRU Model & Optimization:
  In the GRU model, similar to LSTM all the 9 features are being used as inputs and used the same split function logic for extracting the testing and training set. For GRU, the initial model has total 5 layers (**1 sequential, 3 hidden GRU & 1 Dense node**) as the model architecture. The hidden GRU layers has 2048, 1024, 512 neurons with Linear activation function, rest are same as the LSTM model.

  In the optimized model, I have added a **Dropout of 0.2** between each layer, **batch size of 100** and trained it for **epoch 5.**

3.3. Refinement

For the final optimized models for both GRU and LSTM, I have fine-tuned few hyper – parameters and they are as follows;

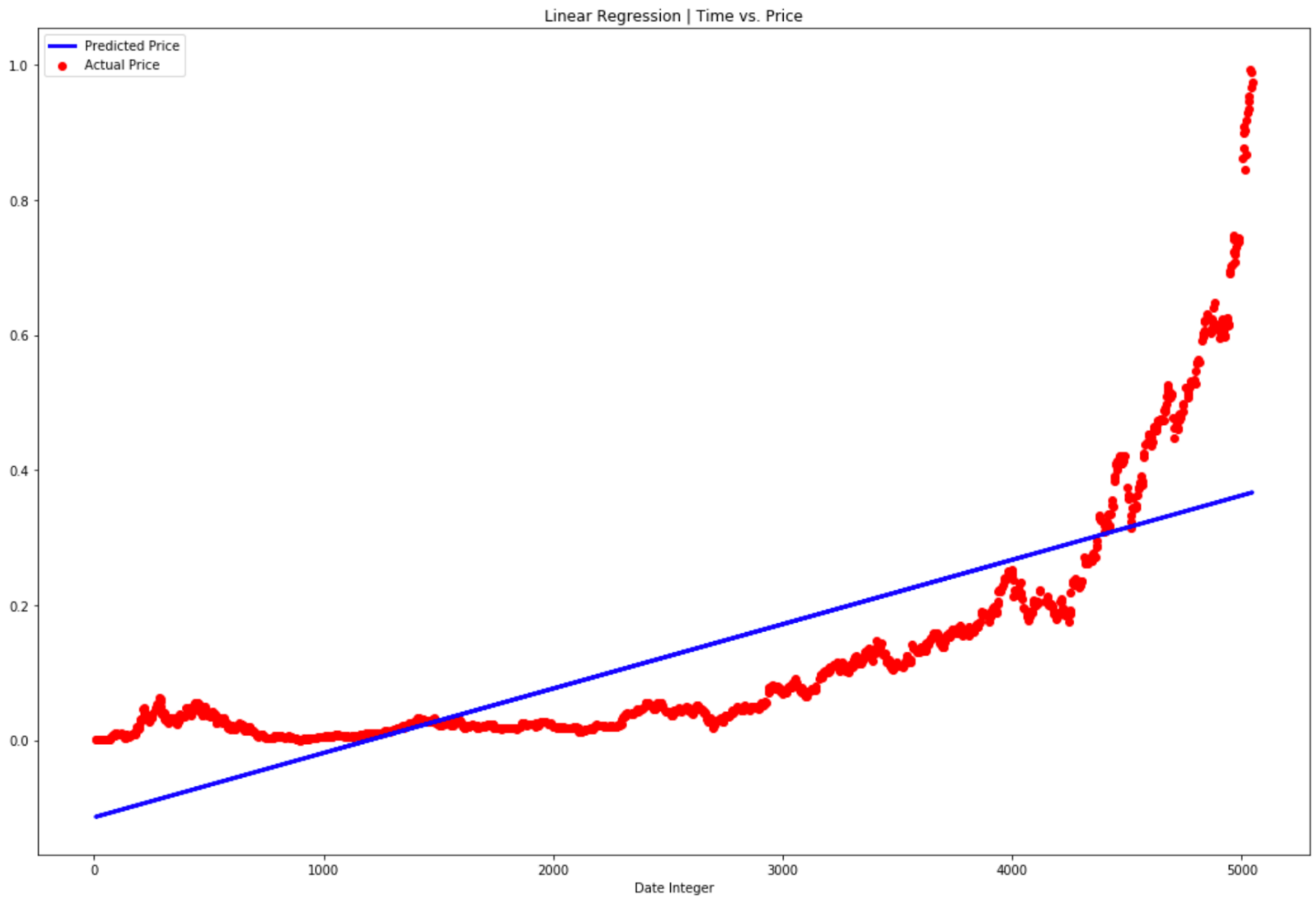
1. Added dropouts of 0.2 between each node.
2. Saw from several trial and run both LSTM and GRU performs well on "Linear" activation function.
3. Batch size of 100 from 1.
4. Epoch from 5 (GRU) and 20 (LSTM). Could have gone more but had Compute power limitation.
5. Optimizer function of "adam" was pretty good in compared to "RMSProp" in general in the trial and run phase.

From general observation, increase the no.of hidden layers drastically deteriorates the performance and error increases. Seen BatchNormalizer did not helped much.

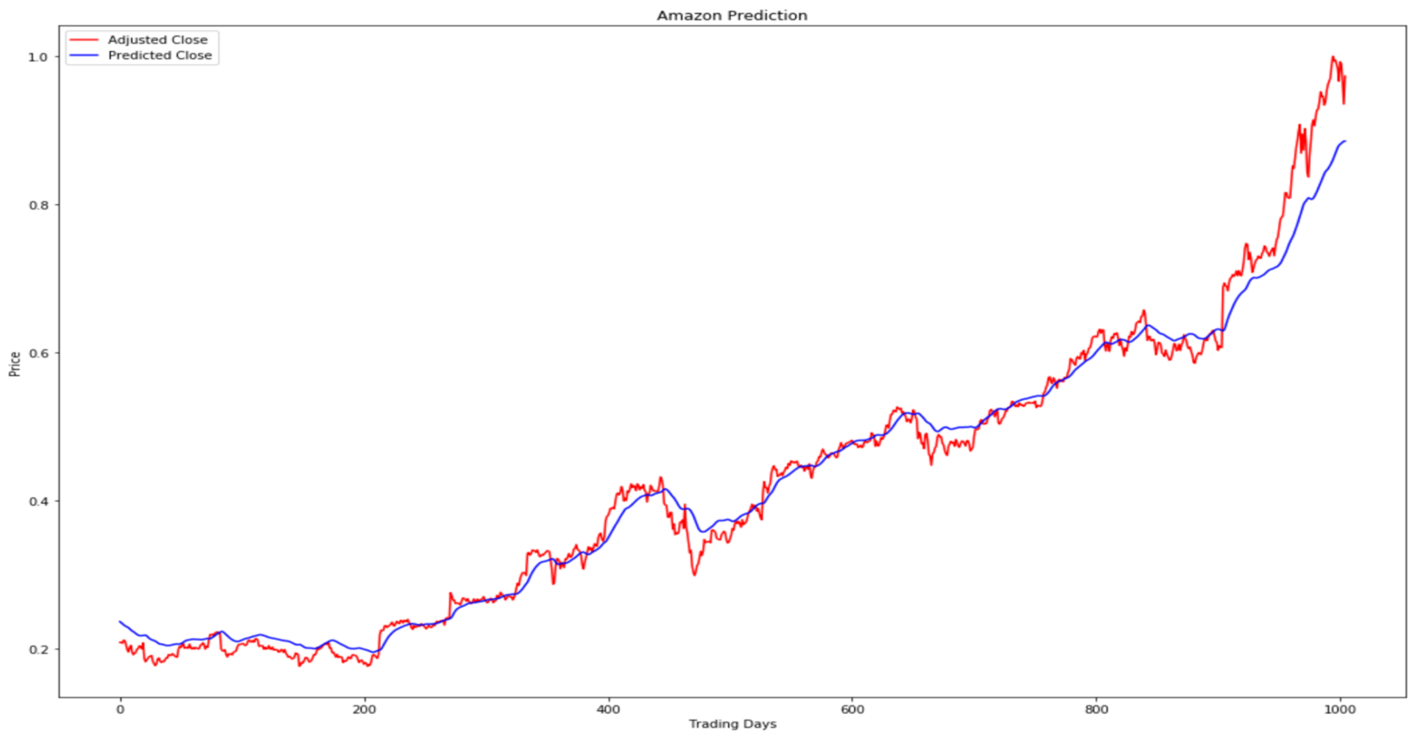
4. Results

4.1. Model Evaluation and Validation

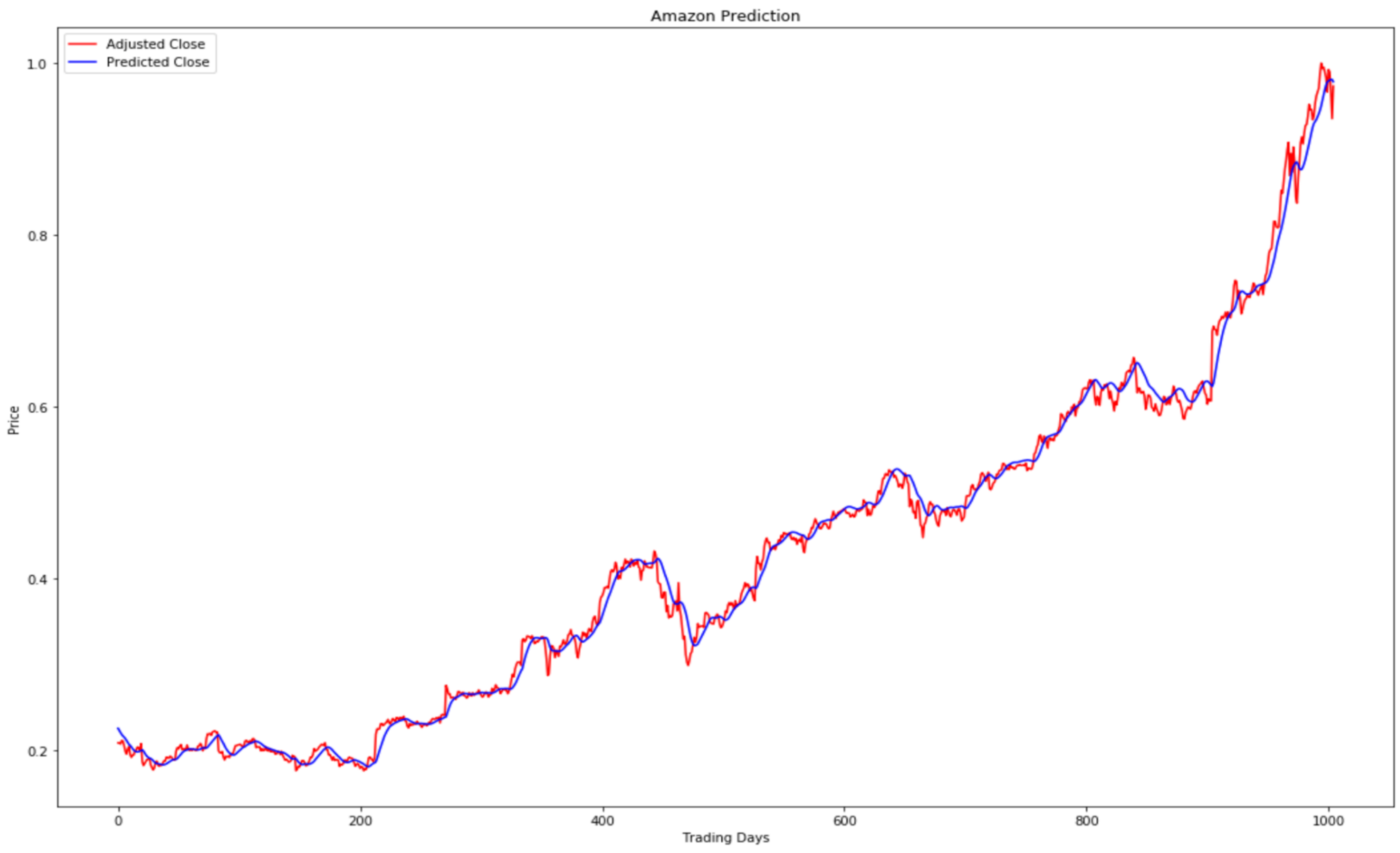
Following are the model performance starting from Benchmark – Linear regression to sophisticated RNN models. Here, I have mentioned both the basic and optimized models with their respective errors.



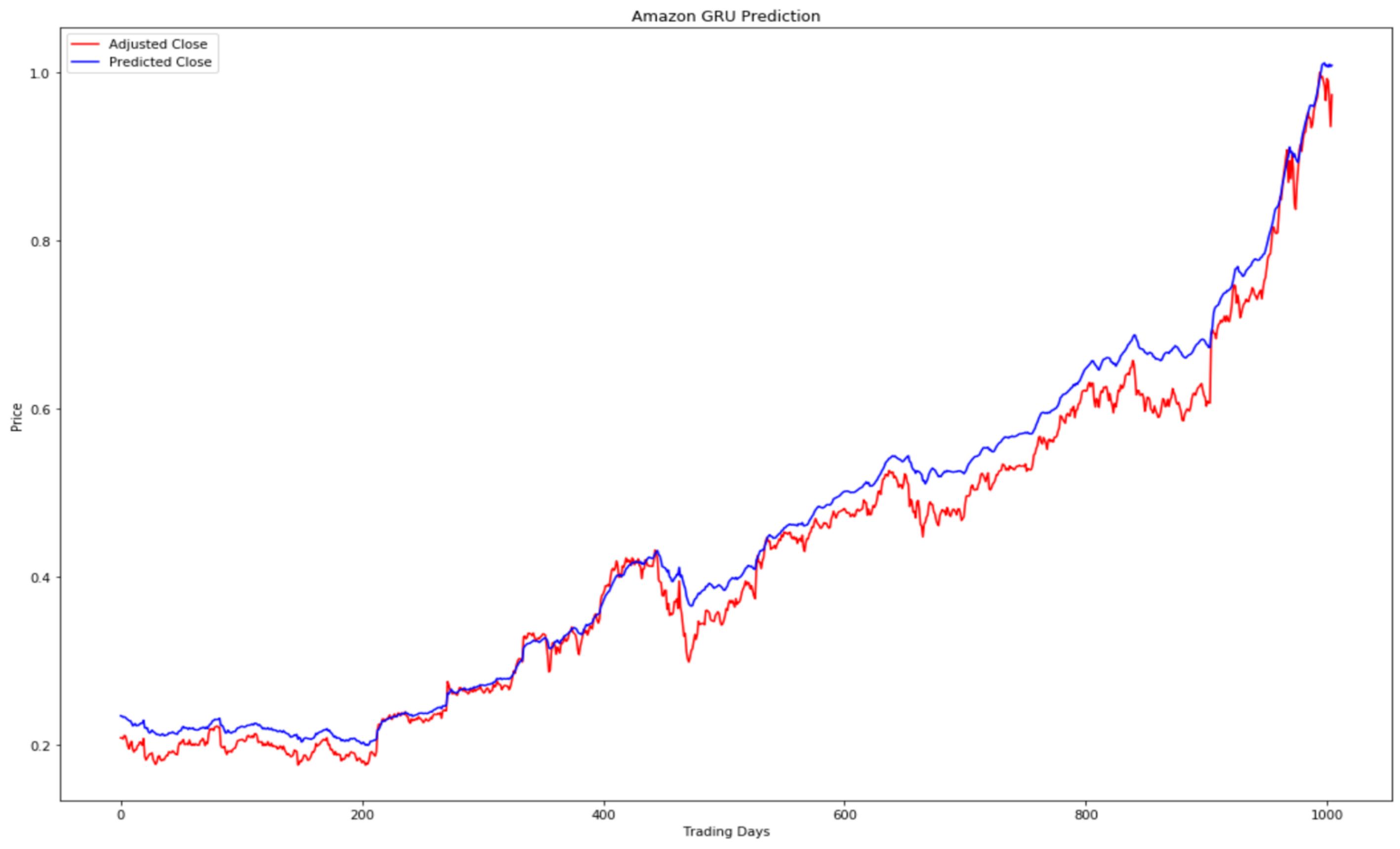
```
Linear Regression - Train Score: 8534412.4325 MSE (2921.3717 RMSE)
Linear Regression - Test Score: 0.01290567 MSE (0.11360313 RMSE)
```



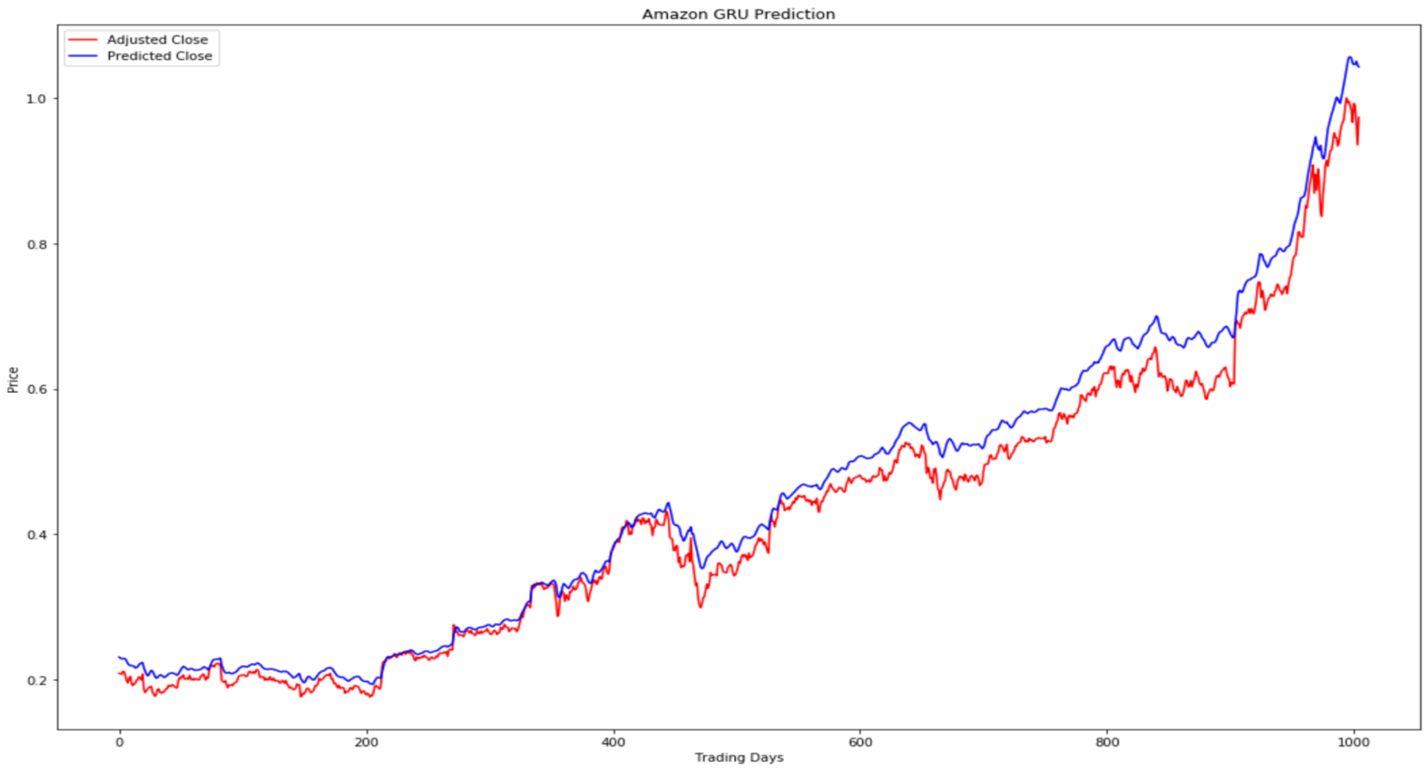
LSTM Basic - Train Score: 0.00004412 MSE (0.00664239 RMSE)
LSTM Basic - Test Score: 0.04161486 MSE (0.20399721 RMSE)



LSTM Optimized - Train Score: 0.00000976 MSE (0.00312377 RMSE)
LSTM Optimized - Test Score: 0.00020909 MSE (0.01445986 RMSE)



```
GRU Basic - Train Score: 0.00002767 MSE (0.00525995 RMSE)
GRU Basic - Test Score: 0.00147107 MSE (0.03835451 RMSE)
```



```
GRU Optimized - Train Score: 0.00001411 MSE (0.00375587 RMSE)
GRU Optimized - Test Score: 0.00107527 MSE (0.03279123 RMSE)
```

4.2. Justification

Below is the summarization of all the models score for training and testing set:

	MSE		RMSE	
Model	**Training Score**	**Testing Score**	**Training Score**	**Testing Score**
Linear Regression	8534412.433	0.01290567	2921.3717	0.11360313
Basic - LSTM	0.00004412	0.04161486	0.00664239	0.20399721
Optimized - LSTM	0.00000976	0.00020909	0.00312377	0.01445986
Basic - GRU	0.00002767	0.00147107	0.00525995	0.03835451
Optimized - GRU	0.00001411	0.00107527	0.00375587	0.03279123

Model Error Summary

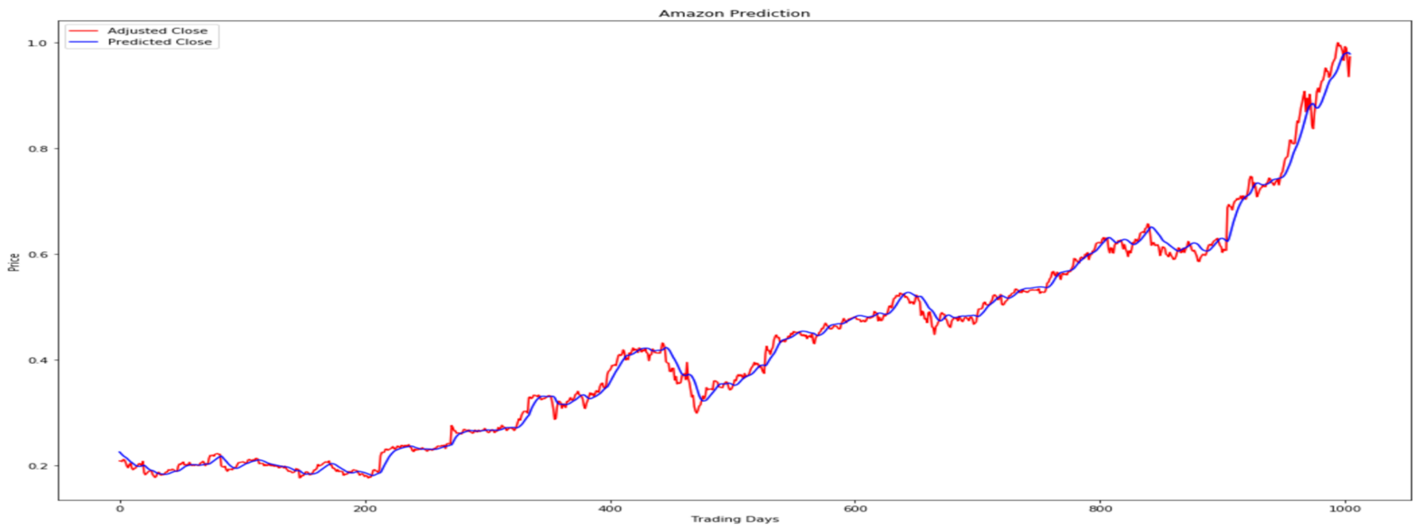
From the above dashboard, it's clear that LSTM model outperformed the other two models.

5. Conclusion

5.1. Free-Form Visualization

One of the important aspects for these models is loss function, which should be minimum as possible. I have already discussed all these errors for all the model in detail in above section. Now, after studying all the charts and the model error summary, it's very clear that the optimized LSTM did a very good job to outperform the other models. I ran these models several time even I had to cost a mac laptop for some graphics logic failure while training the GRU model. But, finally I got the desired output which I was looking for some time for this project.

Below is the final chart and error for the Optimized LSTM model:



LSTM Optimized - `Train Score:  0.00000976 MSE (0.00312377 RMSE)`
LSTM Optimized - `Test Score:   0.00020909 MSE (0.01445986 RMSE)`

5.2. Reflection

In this project, I have used machine learning to predict the stock adjusted closing price based on the stock historical trading data. Below are different details regarding the project.

- **Project Setup & Environment**
  - Anaconda python 2.7 based ipython Jupiter Notebook.
  - ML libraries like Keras, Tensor flow, NumPy, Pandas, Matplotlib etc.
  - Github repository
- **Data Preparation**
  - Use Quandl API to load the Stock data as it is mostly clean Load from Quandl
  - Focus on the FAANG's Amazon stock for analysis.
  - Generated more technical features and removed the null rows from the data frame.
  - Data normalization using the desired function.
  - Leveraged Pandas data frame for the data as it easy to manipulate.
  - 80/20 Data split for preparing the Training and Testing set or the other aforesaid approach.
- **Benchmark Model**
  - Creation of the model based on Linear Regression.
  - Add the technical features for prediction.
- **RNN Model**
  - Prepare 3-layer GRU Model to start with and tuned hyper parameter for optimization.
  - Prepare 2-layer LSTM Model to start with and tuned hyper parameter for optimization.
  - Feed the necessary Technical features.
- **Optimized RNN Model**
  - Added dropouts of 0.2 between each node.
  - Saw from several trial and run both LSTM and GRU performs well on "Linear" activation function.
  - Batch size of 100 from 1.
  - Epoch from 5 (GRU) and 20 (LSTM). Could have gone more but had Compute power limitation.
  - Optimizer function of "adam" was pretty good in compared to "RMSProp" in general in the trial and run phase.
- **Metrics & Result**
  - Visualize the results of Actuals, benchmark and the RNN Models.
  - Analyze and the summarize the result based on the performance and metrics

While working on this project, I felt LSTM performs much better than GRU whole training/testing the neural network. GRU tends to take more time in my laptop. But, then in a proper Cloud environment with good compute power the model can be much faster.

I am a Java J2ee programmer for long time and currently doing real time project in SAP Commerce stack. Python and Machine learning was completely new things for me. But, honestly, I am very much thankful to Udacity ML nano-degree program where I got ample time to learn and complete projects of different categories.

I have gone through multiple sites and research paper but got a good insight form Kaggle more than any others.

My company is heavily investing on Machine learning, I can definitely take this knowledge to explore the Machine Learning Engineering projects for brining automation at large scale for example, Customer Char Bots, Product Recommendation, Cross Selling, Campaign Management etc.

5.3. Improvement

There can be some improvement into this project which crossed my mind while doing this project.

- ➢ Feature selection plays a very important role in this project. All the features might not be of great importance. So, proper correlation diagram can give a better picture. Addition of more fundamentals based on the technical can really make big difference. Then, sentiment analysis with news buzz for each day can have a significant impact.
- ➢ Historical Data range also play an important role. All the data during the market collapse during economic breakdown like Dot Com, Wars etc. should ideally be removed.
- ➢ Test all the hyper parameters and come up with an optimum value. This would need huge compute power to train and test model. Then, apply those in the final model.

Given a chance with proper compute power support and data set, I will surely try to apply these in my future project.

Academia Research paper and reference on this subject –

- https://ssrn.com/abstract=3015609
- http://etd.lib.nsysu.edu.tw/ETD-db/ETD-search/view_etd?URN=etd-0012117-194528
- https://dspace.mit.edu/bitstream/handle/1721.1/105982/965785890-MIT.pdf
- https://www.kaggle.com/sensho/lstm-stock-prediction-20170507
- https://www.quandl.com
- http://cs229.stanford.edu/proj2012/BernalFokPidaparthi-FinancialMarketTimeSeriesPredictionwithRecurrentNeural.pdf