Capstone Project

Machine Learning Engineer
Nanodegree

Nitish Kumar
Nov 21, 2018

Stock Closing Price Prediction

Definition

1. Project Overview

This project is all about exploring the data of stocks and trying to predict the Adjusted Closing Price of the stocks by analysing the previous Opening price of stock and news related to that stock.

Stock market is a well known field of large data, and collection of this data and using complex algorithms on it to predict future prices is not something new. For example in 1997 , an attempt was made to predict stock price using prior knowledge and neural networks [1]. In 2011, a functional link ANN architecture was used for both long and short term stock forecasting, which utilised a standard least mean square algorithm with search-then-converge scheduling to compute a learning rate parameter that changes temporally and required less training experiments [2]. And now in 2018, popular machine learning algorithms were used to predict stock price such as pattern graph [2], convolutional neural network [3], recurrent neural network [4].

For this project we are focussing on two of the worlds biggest companies, Microsoft and Apple. For the news data, New York Times turned out to be the best option as it provides an API access to all its news articles. In this project we will also observe the effect of one companies's stock and news on the other companies's stock price.

2. Problem Statement

" Predicting the Closing Price of the Stock on the basis of Opening price and News related to that stock, on the given day. "

In this project we are going to build a predictive model using Recurrent Neural network (RNN) made using Long Sort-Term Memory (LSTM) nodes, to predict the closing price of the stock with just the news data and the Opening price of the stock on that particular date, and also try to beat the prediction of a Polynomial Regression model(Benchmark model) which also takes the same data as the input.
The strategy for the solving this problem can be outlined as follows:

1. Collect and analyse the data. It will be straight forward to get the stock data, but after that we have to get the new from API for every date we have in the stock data.
2. Process the news data according to the need of the model, as to make the model to understand the news easily we will convert it into sentiment score using Natural Language Toolkit (NLTK).
3. Refine and normalise the complete data set to bring all the into a single range.
4. Make and analyse the performance of the benchmark model.
5. Make the final model using RNN and tune the model till a satisfactory result is obtained in all of the 3 models.

To make the benchmark model I have used a single perceptron with linear activation and polymerised data set to perform polynomial regression. The final model contain LSTM layer to remember the sequence in the data and a set of Dense layer to make effective prediction of the data.

3. Metrics

To measure the performance of the model, both benchmark and the final, I have used the final score as the sum of relative error of all the test data points .i.e

score = (| predicted price - actual price | ÷ actual price) . sum()

Minimum the score the score better the model. I have also plotted the data to see that how much they actually differ from the actual price of the stocks and see if the model is able to recognize the pattern or not, these graph can be seen in the final result representation.
Overall performance of all the model on the training data can also be observed using the loss and the validation loss graph made by the 'tensorboard', these are shown in the Methodology section. These graph were helpful in deciding the

final set of parameters like learning rate for the final models and batch size for the training.

Analysis

1. Data Exploration

This section and the Exploratory Visualisation section will present the finding from the data_visulisation_and_exploration.ipynb file present in the Data subfolder of this project.
This project is done using two data sets, stock data of the company and the news published on the New York Times related to that company. The main data of the company stocks is downloaded from the the kaggle. The data is stored in the form of tables inside the respective csv files of each company. Each table contain 'Date', 'Open', 'High', 'Low', 'Close', 'Adj Close' and 'Volume' column of the given stock. Following are the top 10 rows from the row data file of Stock data:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2006-12-01	13.114285	13.190000	12.871428	91.320000	13.045714	198769900
1	2006-12-04	13.125714	13.150000	12.928572	91.120003	13.017143	177384200
2	2006-12-05	13.092857	13.190000	12.981428	91.269997	13.038571	165709600
3	2006-12-06	12.948571	13.055715	12.810000	89.830002	12.832857	159546100
4	2006-12-07	12.861428	12.928572	12.414286	87.040001	12.434286	251206900
5	2006-12-08	12.461429	12.770000	12.428572	88.259995	12.608571	196069300
6	2006-12-11	12.700000	12.757143	12.578571	88.750000	12.678572	124945100
7	2006-12-12	12.658571	12.691428	12.218572	86.139999	12.305715	256655000
8	2006-12-13	12.564285	12.724286	12.450000	89.050003	12.721429	214263000
9	2006-12-14	12.721429	12.857142	12.608571	88.549995	12.650000	208082700

Apple stock data (top 10 rows from AAPL.csv file)

	Date	Open	High	Low	Close	Adj Close	Volume
0	2006-12-01	29.230000	29.299999	28.900000	29.120001	29.120001	72257000
1	2006-12-04	29.230000	29.520000	29.170000	29.330000	29.330000	55123400
2	2006-12-05	29.360001	29.400000	29.030001	29.129999	29.129999	45606000
3	2006-12-06	29.100000	29.129999	28.870001	28.990000	28.990000	48564100
4	2006-12-07	28.959999	29.070000	28.809999	28.850000	28.850000	46831100
5	2006-12-08	28.820000	29.400000	28.799999	29.400000	29.400000	108854900
6	2006-12-11	29.190001	29.750000	29.110001	29.540001	29.540001	107712000
7	2006-12-12	29.559999	29.629999	29.219999	29.430000	29.430000	68529400
8	2006-12-13	29.600000	29.600000	29.320000	29.549999	29.549999	46002500
9	2006-12-14	29.540001	30.080000	29.520000	30.070000	30.070000	85866500

Microsoft stock data (top 10 rows from MSFT.csv file)

Then these stocks data were used the get_news.ipynb in the Data subfolder to gather the news of the company for the respective date from the stock data field. In the process if the API search returned the data then I extracted the headline and the summary of all the news, made a single string from it, and stored it into a separate file. After this process we have two new files .i.e AppleNewsStock.csv and MicrosoftNewsStock.csv. Again these files can be read using the pandas, and top 15 rows of the news collected looks like following:

```
micro['News'][:15]
```

```
0                                                   NaN
1     The Retooling of a Search Engine : Ask.com is ...
2                                                   NaN
3     Combat as Usual? Not With These Games : A few ...
4     Vista Is Ready. Are You? : Why it might be a b...
5                                                   NaN
6                                                   NaN
7     Take the Hotel Room Home : Hotels have become ...
8     Google to Offer Variation on Stock Options : G...
9                                                   NaN
10    Law Firms' Merger Will Expand Reach : Kirkpatr...
11                                                  NaN
12    Finally, a Way to Catch a Flight Without Shedd...
13    Google Steps More Boldly Into PayPal&#8217;s T...
14                                                  NaN
Name: News, dtype: object
```

Microsoft News data (top 15 rows)

```
apple['News'][:15]
```

```
0     WHAT'S ON TONIGHT : 8 P.M. (TLC) ASHLEY JUDD A...
1     More on Housing Prices : The broadest governme...
2                                                   NaN
3     Honoring R.W. Apple in Words and Food : About ...
4     Homebuilders, and Worries Over Jobs, Lead a De...
5     Homebuilders, and Worries Over Jobs, Lead a De...
6     Sales of iPods and iTunes Not Much in Sync : T...
7                                                   NaN
8     Op-Art; iDentities : Op-Art by Seymour Chwast ...
9     Adam, She's Ms. Madam : Kristin Chenoweth uses...
10    Adam, She's Ms. Madam : Kristin Chenoweth uses...
11    Apple Delays Earnings Statement Due to Options...
12    Maple's Got Some New Tricks : Maple has new di...
13    Recent Openings  : New shows in New York inclu...
14                                                  NaN
Name: News, dtype: object
```

Apple News data (top 10 rows)

After gathering the data get_news.ipynb notebook also calculates the sentiment data of the news using the nltk library which makes the following final data sets:

	Date	Open	High	Low	Close	Adj Close	compound	neg	neu	pos
0	2006-12-01	29.23	29.30	28.90	29.12	29.12	0.0000	0.000	0.000	0.000
1	2006-12-04	29.23	29.52	29.17	29.33	29.33	-0.7783	0.137	0.863	0.000
2	2006-12-05	29.36	29.40	29.03	29.13	29.13	0.0000	0.000	0.000	0.000
3	2006-12-06	29.10	29.13	28.87	28.99	28.99	0.9003	0.051	0.847	0.102
4	2006-12-07	28.96	29.07	28.81	28.85	28.85	0.6597	0.000	0.810	0.190
5	2006-12-08	28.82	29.40	28.80	29.40	29.40	0.0000	0.000	0.000	0.000
6	2006-12-11	29.19	29.75	29.11	29.54	29.54	0.0000	0.000	0.000	0.000
7	2006-12-12	29.56	29.63	29.22	29.43	29.43	0.0000	0.000	1.000	0.000
8	2006-12-13	29.60	29.60	29.32	29.55	29.55	0.5574	0.000	0.833	0.167
9	2006-12-14	29.54	30.08	29.52	30.07	30.07	0.0000	0.000	0.000	0.000

Microsoft Final data set (top 10 rows from MicrosoftFinalData.csv file)

	Date	Open	High	Low	Close	Adj Close	compound	neg	neu	pos
0	2006-12-01	13.1143	13.1900	12.8714	91.32	13.0457	0.7707	0.032	0.905	0.063
1	2006-12-04	13.1257	13.1500	12.9286	91.12	13.0171	0.8720	0.011	0.904	0.085
2	2006-12-05	13.0929	13.1900	12.9814	91.27	13.0386	0.0000	0.000	0.000	0.000
3	2006-12-06	12.9486	13.0557	12.8100	89.83	12.8329	0.6858	0.029	0.878	0.093
4	2006-12-07	12.8614	12.9286	12.4143	87.04	12.4343	-0.6712	0.091	0.869	0.040
5	2006-12-08	12.4614	12.7700	12.4286	88.26	12.6086	-0.1796	0.084	0.848	0.069
6	2006-12-11	12.7000	12.7571	12.5786	88.75	12.6786	-0.8743	0.105	0.852	0.042
7	2006-12-12	12.6586	12.6914	12.2186	86.14	12.3057	0.0000	0.000	0.000	0.000
8	2006-12-13	12.5643	12.7243	12.4500	89.05	12.7214	0.9360	0.018	0.900	0.082
9	2006-12-14	12.7214	12.8571	12.6086	88.55	12.6500	0.9620	0.026	0.880	0.094

Apple Final data set (top 10 rows from AppleFinalData.csv file)

For the final price I have used the 'Adj Price' for both of the stocks. One important point to note is that, the news data is in range of -1 to 1 but the stock data is not, hence before plotting graphs, I Normalised the stock data one column at a time, using following formula :

Normalised Data = (Raw data - Min of Range) ÷ (Max of Range - Min of Range)

It's a known fact that these companies have fixed yearly events and they affect their stock values, to make the model understand I also added the month in the final data frame before using it for the training of the model.

2. Exploratory Visualisation

All the visualisation of the data in this project is done using the "matplotlib". Following are all the graphs in the sequence that I plot to understand the data and the conclusion I made on the basis of them at that time:

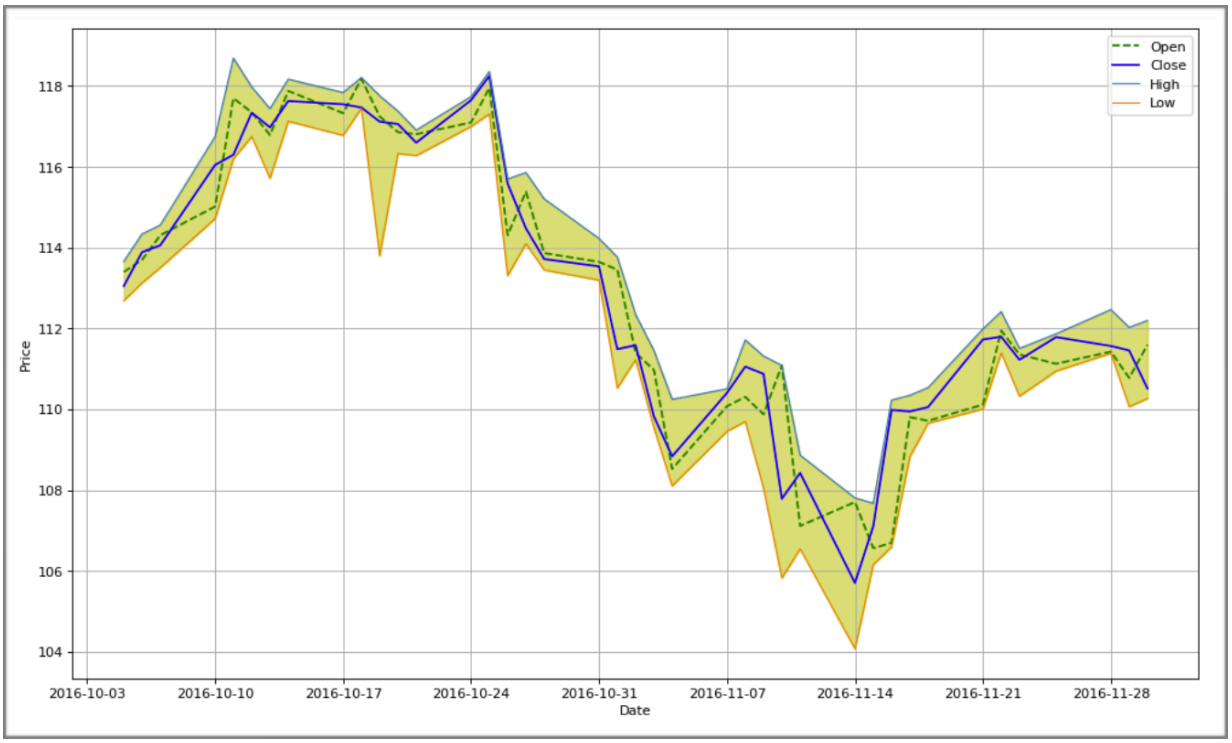
1. Stock's "Adj Close" for Both the companies: From this graph we can see that both the companies have almost same pattern in the price rise and fall although they have their separate stock price range, this was the reason that I chose to add a new column containing only the month number so that the model can also

Apple and Microsoft Raw Adj Closing Prices

understand the years pattern in these data.

2. For the further analysis I used the apple stock data. Hence the 2nd plot is the combined plot of Open, Close, High and Low of the apple stocks. In the following

graph the yellow area is the range of the price range, dashed line is opening price and solid blue line is the Closing price.



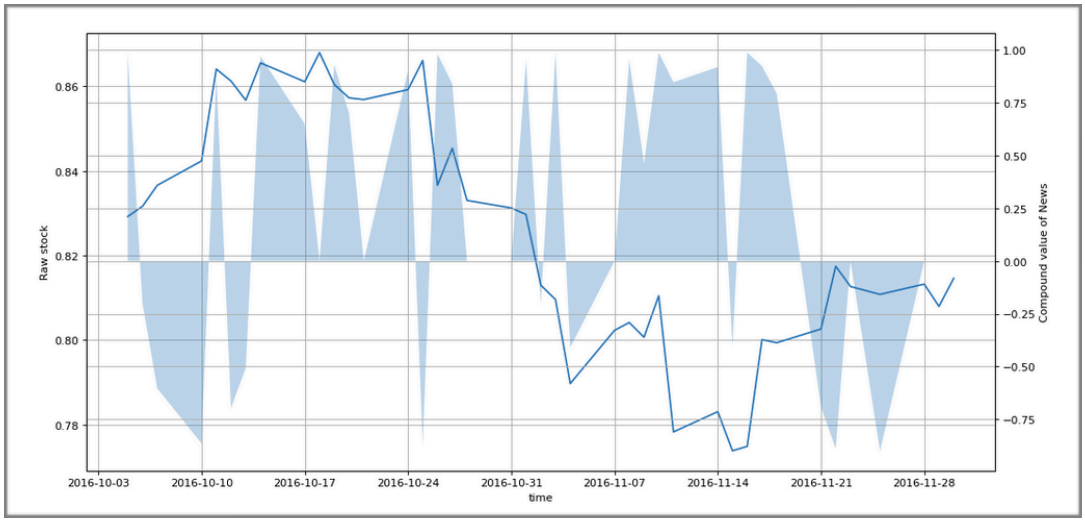
Apple Stock data for the last 2 month

3. Visualisation of the Normalised data to check that the final data still contain all the patterns after the normalisation.

Apple Raw and Normalised Closing Stock data

4. The final data set also contain the sentiment analysis of the news related to both of the companies. In this graph we can see the relation between the rise and fall

of the price based on the news in that region. In the following graph blue line is the final normalised stock prices of apple and the area fill is made on that graph.



Apple Closing data (normalised) overlapped by the Compound sentiment value of the news related to Apple

5. In the last, after visualising all the data I plotted the training, validation and testing data set. Validation data set is 10% of the training data set.

Apple Opening data into 3 data sets.

Microsoft Opening data into 3 data sets.

3. Algorithms and Techniques

Following are the algorithms for benchmark and final model, that were used:

1. Benchmark

   Benchmark model is using the Polynomial Regression to predict the closing price of these stock. To do so I have used a single node single layer neural network .i.e a perceptron. This perceptron is trained on the same Normalised data set but with few more extra features generated through Polymerisation.

   The perceptron is made using Dense layer of Keras with single output and linear activation function. To optimise this during the training I used the 'adam' optimiser with default settings.

2. Final model

   For the final prediction I made 3 different model, 2 of them are trained on both of the companies data separately with the news data of each and the 3rd one is trained on the combined data of the companies with their news data. The 3rd model was made to check the effect of stock due to competitor companies in the same market.

   All of the these models contain 3 LSTM layers and 3 Dense layers. The separate prediction models have single node in these output node and the combined one has 2 node to predict the price of both of the company. The LSTM layers are a kind of Recurrent Neural network which work great with the sequential data, and these LSTM layers do not suffer from the gradient vanishing problem like the RNN does with long sequential data sets.

3. Training

   This is a interesting data set due to the time sequence, hence I have use following 2 different training process:

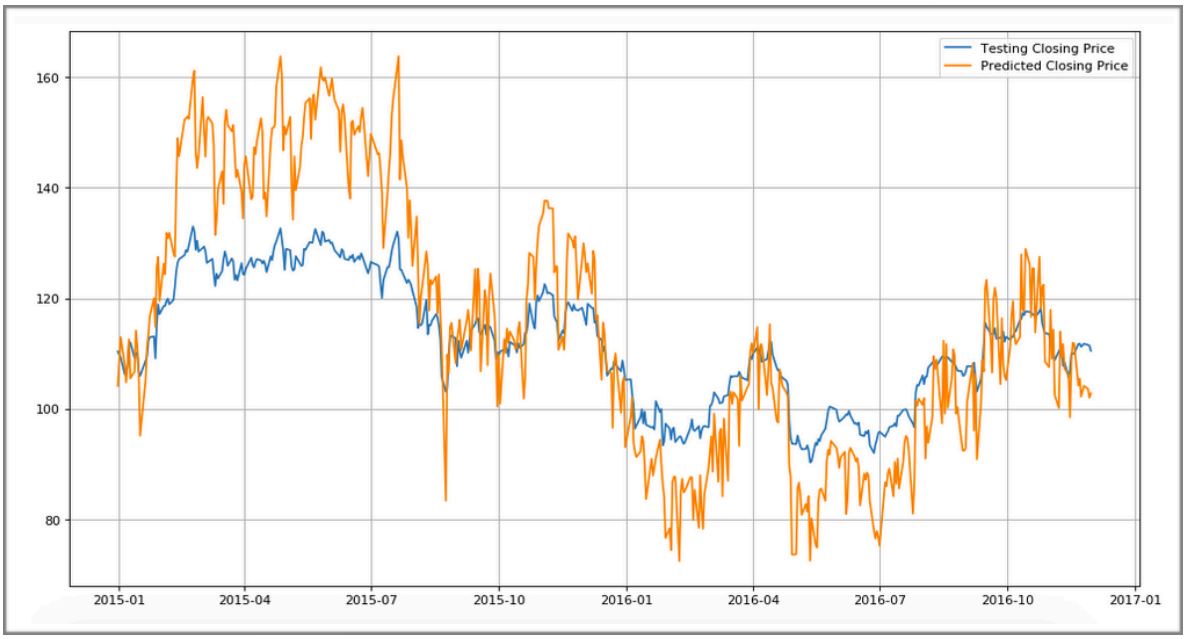
   - Simple single training: it a conventional way of training, in which the model if provided with complete data and it trains on the complete data during all the epochs.
   - Framed training: The problem with the above approach is that it sends the complete data of last 10 years to the model and we are trying to predict the closing price of the next day. Hence in this framed way I made a data frame of 40% and a shift size of 10% of the training data set. Then at the time of training the model 1st get the starting 40% of data and next 10% as validation, after training the frame is shifted forward by a shift length of 10% and the training starts again. This process repeats itself till the frame reaches its end. This framed training gives preference to the latest data more than the older data and by overwriting the weights of the model.

4. Benchmark

The benchmark model is present in the 'Benchmark.ipynb' notebook. I have used the above benchmark algorithm to predict the closing price of the stocks. The bench mark used is made using mesa Dense layer and is trained using "Adam" optimiser. Following are the results of the benchmark for the respective company:

- Apple:

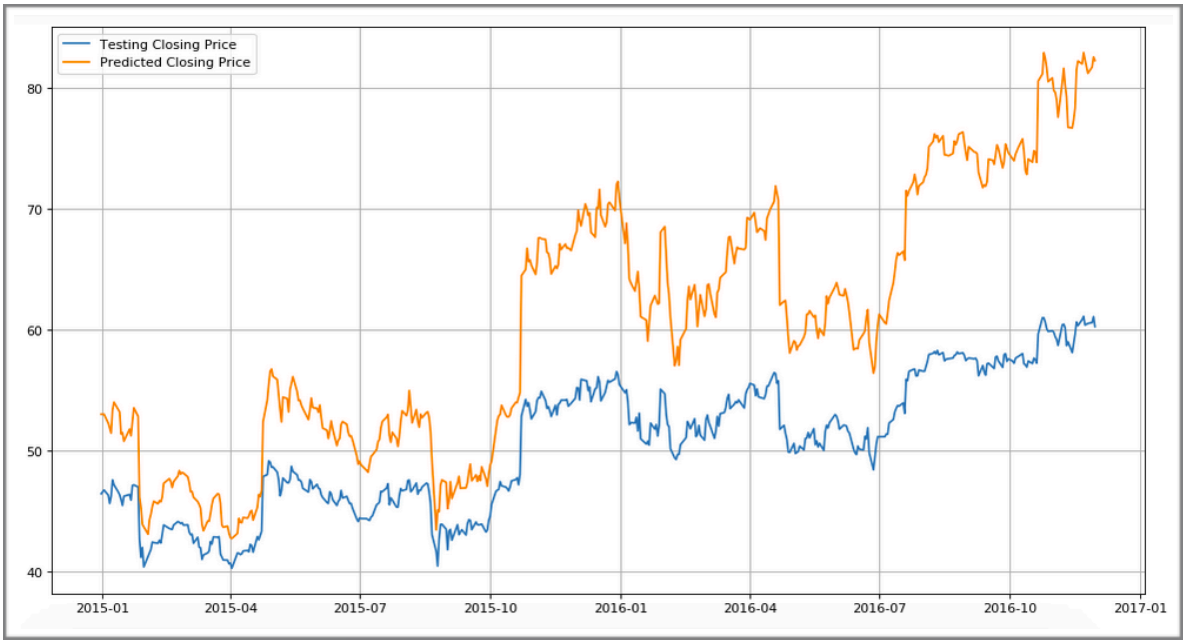
  Final score of the benchmark is: 44.3202 on testing data set of apple



Benchmark prediction on Apple stocks

- Microsoft:

  Final score of the benchmark is: 90.6457 on testing data set of Microsoft



Benchmark prediction on Microsoft stocks

These benchmark models are able to identify the pattern in the abstract manner .i.e they can predict the either the cost will rise or fall, but these are lacking in the accuracy of the data.

Methodology

1. Data Processing

Data processing is done on every notebook that uses the stock data. Following are the common steps taken :

i) Arranging Columns: After accessing the "csv" file, I renamed the Adj Close column as Price and removed the Close column. This was helpful in plotting the graphs. After that I created a new field of Month and extracted the month number from the Date column and store it into the Month as an integer, so that they could be used in the training. And finally I arranged the column in way that all the Feature set columns were before the month column and the target column were after the Date.

ii) Data Normalisation: Values in the column related to the stocks have different range than that of the data in the sentiment columns, so I split the main data into two different data sets, "stock" and "news". And now we have the stock data into separate set we can implement the following Normalisation formula on all of the columns:

$$stock=(stock-min(stock)) \div (max(stock)-min(stock))$$

iii) Data Polymerisation : After Normalisation of the data I polymerised the stock data using the "sklearn" lib. This is done to increase the final feature set that will be used to train the model. For the benchmark the degree of polymerisation is 3 and for the main model it is 2.

iv) Make the Training and Testing Data: Once the data is completely processed I combined the news and stock data to make the final training and testing sets. The training data set is of 2033 rows in length for every model and the testing dat have 408 rows in it.

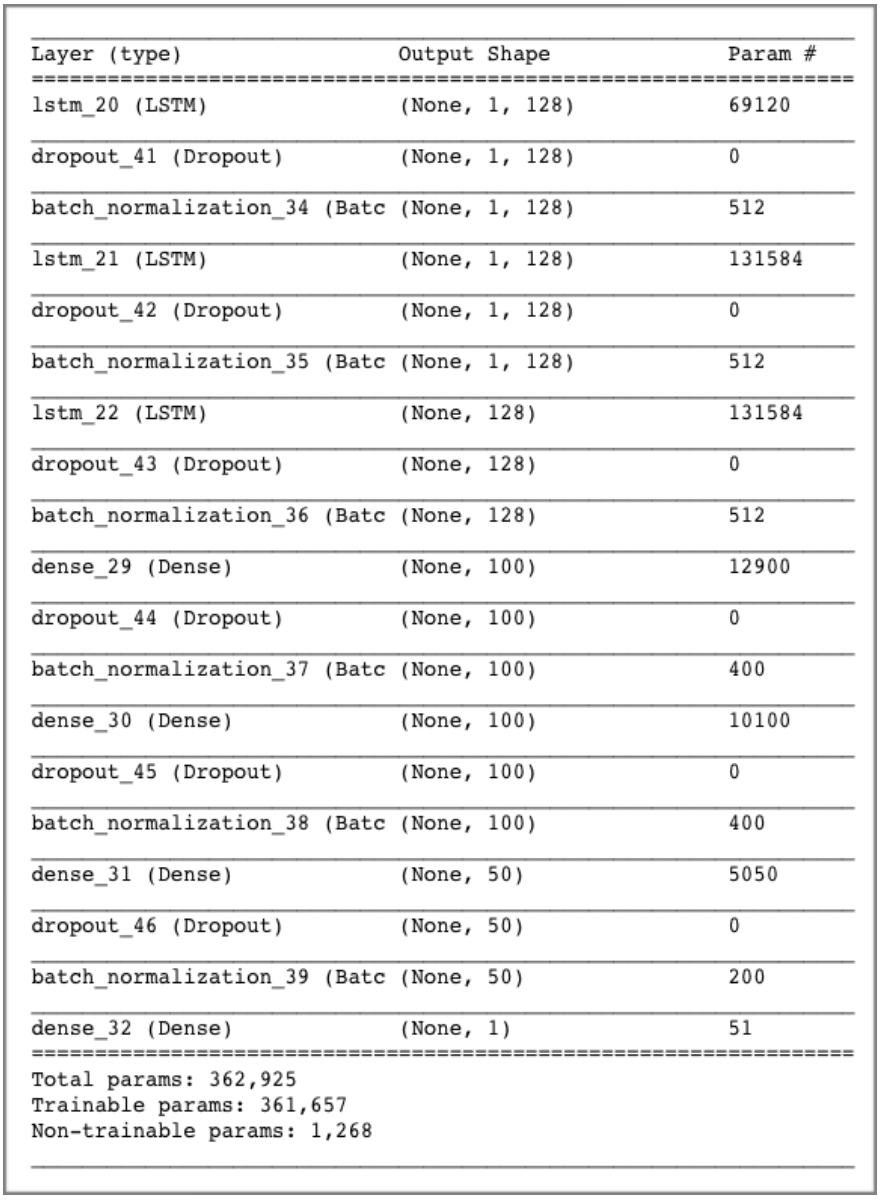
For the training of the main model I have only used the Opening Price of the stock from the stock price and its squared value as the polynomial feature, making the final feature set of 7, two from the stocks and five from news.

LSTM in the main model requires one more step after these processing as it accepts a 3D array [samples, time steps, features] and the current data is in 2D array shape [samples, features]. Hence the data was converted before using it into the training of the LSTM layers. In this we do not have to change the

dimension of the target data. This step is really important to perform as it causes a lot of errors and due to just one reason.

2. Implementation

The 1st model that is implemented is for the apple stocks, and it is present in the “apple.ipynb” notebook. This notebook 1st reads and preprocess the apple stock data as described in the above data processing section. This model contain 3 LSTM layer where 1st two layers has return_sequence = True, and the 3rd layer with return_sequence = False as after that there are 3 Dense layer and one final output layer of one node. All the layers have ‘relu’ as their activation function and are followed by a pair of Dropout layer and a BatchNormalization layer to have a generalised output. Hence the final summary looks like this:



Layer (type)	Output Shape	Param #
lstm_20 (LSTM)	(None, 1, 128)	69120
dropout_41 (Dropout)	(None, 1, 128)	0
batch_normalization_34 (Batc	(None, 1, 128)	512
lstm_21 (LSTM)	(None, 1, 128)	131584
dropout_42 (Dropout)	(None, 1, 128)	0
batch_normalization_35 (Batc	(None, 1, 128)	512
lstm_22 (LSTM)	(None, 128)	131584
dropout_43 (Dropout)	(None, 128)	0
batch_normalization_36 (Batc	(None, 128)	512
dense_29 (Dense)	(None, 100)	12900
dropout_44 (Dropout)	(None, 100)	0
batch_normalization_37 (Batc	(None, 100)	400
dense_30 (Dense)	(None, 100)	10100
dropout_45 (Dropout)	(None, 100)	0
batch_normalization_38 (Batc	(None, 100)	400
dense_31 (Dense)	(None, 50)	5050
dropout_46 (Dropout)	(None, 50)	0
batch_normalization_39 (Batc	(None, 50)	200
dense_32 (Dense)	(None, 1)	51

Total params: 362,925
Trainable params: 361,657
Non-trainable params: 1,268

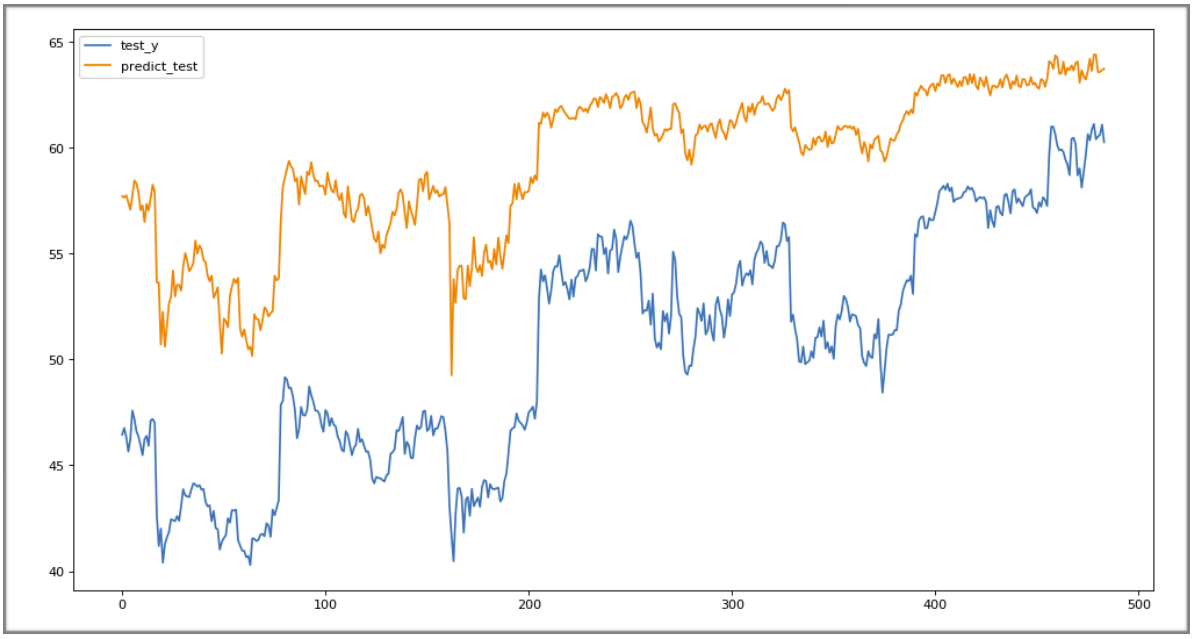
Model summary

For the debugging I am also using the tensorboard callback to generate the log file of the model during the training and ModelCheckpoint to store the weights of the model when it has minimum validation loss, just to skip the overfitting, if occurs. Initially RMSprop optimiser is user with 0.001 learning rate is used.

Then the model is trained using model.fit(). Initially I trained it for 30 to 40 epochs and I started with the batch size of 10. I kept the verbose = 1 so that I can see where it got the minimum loss in validation. This setup produced the score of 109.6592 with apple data and prediction looks like:

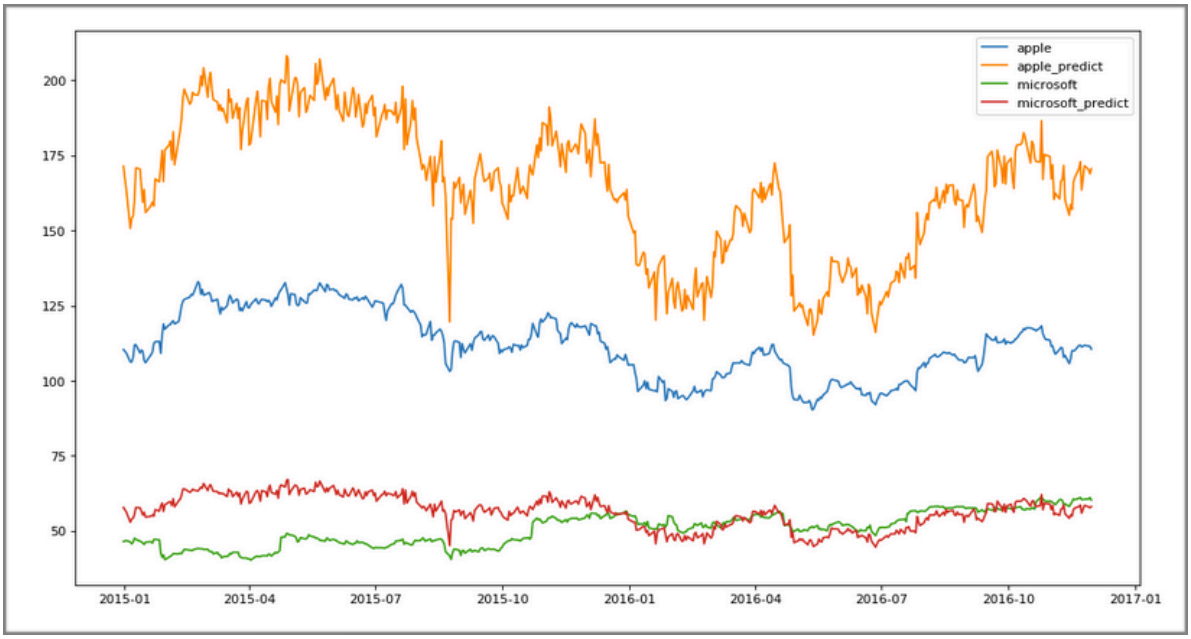
Prediction by Baseline Structure implementation on Apple data

The model is able to predict the sequential pattern in the stock data, it just has to improve the final price values, hence it need modification in the Dense layers. This model was also implemented with the Microsoft data, and it gave the score of 87.5499 and the following prediction graph.



Prediction by Baseline Structure implementation on Microsoft data

These results are also similar to the prediction of the apple one as in this they are able to predict eh yearly pattern but not able to exactly calculate the price. Now it's time for the 3rd implementation of model with both data combined. For this also I have use used the same model structure with 2 nodes in the final output layer. With both of the companies data this model performed even worse as it gave the score of 224.8493 with the apple stock prediction and 85.6477 with the Microsoft data and crated the following graph.



Prediction by Baseline Structure implementation on combined data of Apple and Microsoft.

The baseline model shows remarkable results in remembering the yearly pattern of the stock price, but lacks in the prediction of the exact value for the individual data sets.

3. Refinement

First thing that can be seen from the prediction graph is that the model is not able to calculate the price close to the actual Price, hence I have to improve the Dense layers. To test I started with the higher values like 500/200/100 nodes in three consecutive layers. Results from this structure was better than the baseline but it was still suffering from the overfitting as if the data was out of the range of the training data it was not performing well.

I also tested the model with different training setups, that is with different size of batch and epochs. It was taking not able to identify the general pattern with smaller batch size hence to improve the performance I increased the batch size and tested the model with batch size near 50. And to decide the number of epoch I trained the model with 10 epochs repeatedly till it started overfitting and noted down the final sets. For most of the parameter sets, it took nearly 60 to 80 epoch with batch size near 50 to get the model to overfit.

For the sake of exploration I also tested the model with different optimisers like Adam and Adadelta, but the RMSprop had the best results. For the error function I also tried “mean_absolute_error” and “mean_square_error”. For the training I also tested the model with the following ways:

1. Simple training : in this all the data is provided to the model at the same time.
2. Framed training : in this a data frame is moved from one end of the data set to the other end and the model is trained on each frame with the weights of the previous frame. This process also created a new window for more exploration as what should be the frame size and the shift size to shift the frame. I started with 60% frame size and 10% shift size.

At the time of testing different parameters I first trained the model with simple training and if the parameter set showed improvement, then I used the framed training with the same parameter set. At the end of tuning the final model got 11.56 and 11.77 score with Apple and Microsoft data respectively, on simple training and 200/100/50 nodes in Dense layer.

After the improvement in the separate model I started working on the combined model and took the same steps to get the result. As the combined data model had a bigger feature set it took more nodes in the dense layers to see some significant results nearly 22.73 and 15.05 on Apple and Microsoft data respectively with the simple training and 300/150/50 nodes in the Dense layers.

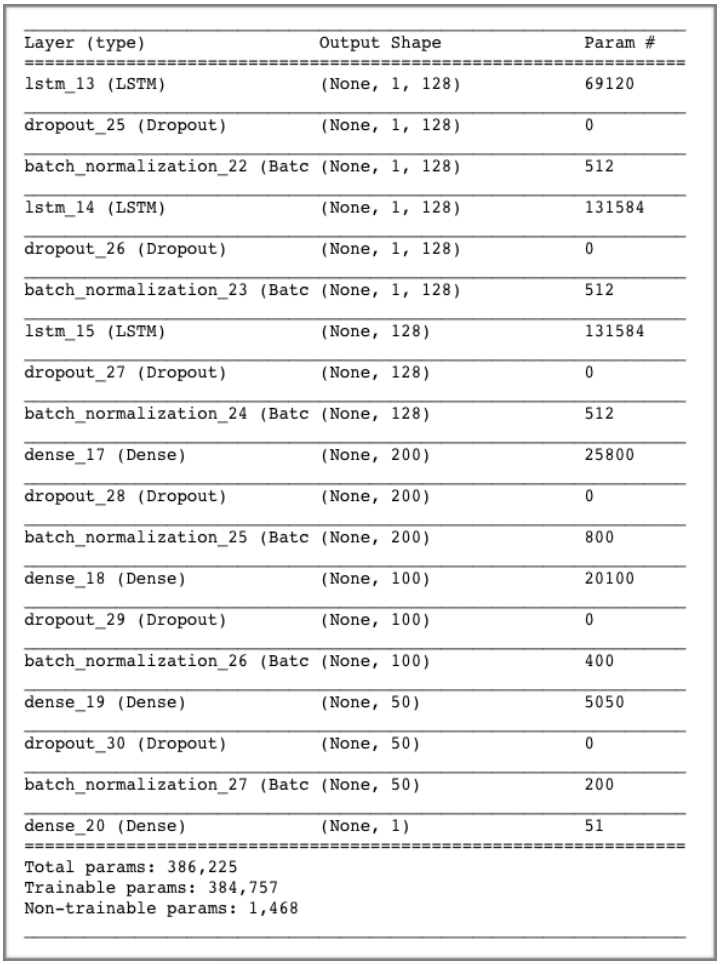
Results

1. Model Evaluation and refinement

Following are the final set of parameters and model structure that I used

Separate Model (For Apple and Microsoft both):

- Model Parameter:
  Separate model have 3 LSTM layers, 3 Dense layer and 1 final output node. Each layer is followed by a combination of Dropout layer and BathNormalization layer. I used ‘relu’ activation function for every layer, except the last output layer, it docent have one. Final structure of the model is as follows:



Layer (type)	Output Shape	Param #
lstm_13 (LSTM)	(None, 1, 128)	69120
dropout_25 (Dropout)	(None, 1, 128)	0
batch_normalization_22 (Batc	(None, 1, 128)	512
lstm_14 (LSTM)	(None, 1, 128)	131584
dropout_26 (Dropout)	(None, 1, 128)	0
batch_normalization_23 (Batc	(None, 1, 128)	512
lstm_15 (LSTM)	(None, 128)	131584
dropout_27 (Dropout)	(None, 128)	0
batch_normalization_24 (Batc	(None, 128)	512
dense_17 (Dense)	(None, 200)	25800
dropout_28 (Dropout)	(None, 200)	0
batch_normalization_25 (Batc	(None, 200)	800
dense_18 (Dense)	(None, 100)	20100
dropout_29 (Dropout)	(None, 100)	0
batch_normalization_26 (Batc	(None, 100)	400
dense_19 (Dense)	(None, 50)	5050
dropout_30 (Dropout)	(None, 50)	0
batch_normalization_27 (Batc	(None, 50)	200
dense_20 (Dense)	(None, 1)	51

Total params: 386,225
Trainable params: 384,757
Non-trainable params: 1,468

Structure of the final Separate Model

- Model Compilation:

  Final mode is using RMSprop as its optimiser with learning rate of 0.0005, I left the remaining parameter to default value .i.e rho=0.9 and decay=0 . It uses " mean_squared_error " as its final loss function, with 'accuracy' as metrics.

- Training:

  To save the data log of the training I have used the TensorBoard and ModelCheckpoint. And the remaining parameters are as follows:

  Simple training :

  Training X= (2033,7), .i.e complete data set

  Training Y = (2033,1)

  epoch=80

  Batch size=50

  Validation split= 0.1 .i.e 10%

  Framed training:

  Training X = (40% of 2033,7), this frame id shifted with 10% shift size after every training of 30 epochs.

  Training Y = Same as the X .i.e (40% of 2033,1)

epoch=30 for all frames and 20 for the extra end frame
Batchsize=50
Validation split=0.1, .i.e 10% of the current frame

Combined Model: this model have a bigger feature set and have to predict 2 values on a single run, hence the following parameters:

- Model Parameter:
  Common model also have 3 LSTM layers, 3 Dense layer and 2 final output node. Each layer is followed by a combination of Dropout layer and BathNormalization layer. I used ' relu ' activation function for every layer, except the last output layer, it docent have one. Final structure of the model is as follows:

Layer (type)	Output Shape	Param #
lstm_13 (LSTM)	(None, 1, 128)	73216
dropout_25 (Dropout)	(None, 1, 128)	0
batch_normalization_25 (Batc	(None, 1, 128)	512
lstm_14 (LSTM)	(None, 1, 128)	131584
dropout_26 (Dropout)	(None, 1, 128)	0
batch_normalization_26 (Batc	(None, 1, 128)	512
lstm_15 (LSTM)	(None, 128)	131584
dropout_27 (Dropout)	(None, 128)	0
batch_normalization_27 (Batc	(None, 128)	512
dense_17 (Dense)	(None, 300)	38700
dropout_28 (Dropout)	(None, 300)	0
batch_normalization_28 (Batc	(None, 300)	1200
dense_18 (Dense)	(None, 150)	45150
dropout_29 (Dropout)	(None, 150)	0
batch_normalization_29 (Batc	(None, 150)	600
dense_19 (Dense)	(None, 50)	7550
dropout_30 (Dropout)	(None, 50)	0
batch_normalization_30 (Batc	(None, 50)	200
dense_20 (Dense)	(None, 2)	102

Total params: 431,422
Trainable params: 429,654
Non-trainable params: 1,768

Structure of the final Common Model

- Model Compilation:
  Final mode is using RMSprop as its optimiser with learning rate of 0.0005, I left the remaining parameter to default value .i.e rho=0.9 and decay=0 . It uses " mean_squared_error " as its final loss function, with 'accuracy' as metrics.

- Training:

  To save the data log of the training I have used the TensorBoard and ModelCheckpoint. And the remaining parameters are as follows:

  Simple training :

  Training X= (2033,7), .i.e complete data set
  Training Y = (2033,1)
  epoch=80
  Batch size=50
  Validation split= 0.1 .i.e 10%

  Framed training:

  Training X = (40% of 2033,7), this frame id shifted with 10% shift size after every training of 30 epochs.
  Training Y = Same as the X .i.e (40% of 2033,1)
  epoch=30 for all frames and 20 for the extra end frame
  Batchsize=50
  Validation split=0.1, .i.e 10% of the current frame

2. Justification

Following are the final score** with all model with Apple and Microsoft Stock and news data:

1. Simple training:

Models ↓	Apple Data	Microsoft Data
Benchmark model	44.3202	90.6457
Separate Model	11.5601	11.7765
Combined model	22.7319	15.0587

2. Framed Training:

Models ↓	Apple Data	Microsoft Data
Benchmark model	44.3202	90.6457
Separate Model	6.1561	8.4034
Combined model	14.9829	46.4753

** These scores are calculated using the Relative error.

Conclusion: we can clearly see the effect of using LSTM layers as they have a significant improvement. And although the combined model didn't performed

as expected, but they also have better score than the benchmark and have good prediction graph also in comparison to benchmark.

Conclusion

1. Free-Form Visualisation

Following are the final prediction graph of all the models

A. Separate Apple model with Simple training : model was able to predict the pattern and the price are also almost same, on very close observation we can see that it has problem in predicting the rise in the price but it is good in the falls overall.

Apple Model prediction on simple training

B. Separate Apple model with Framed training :

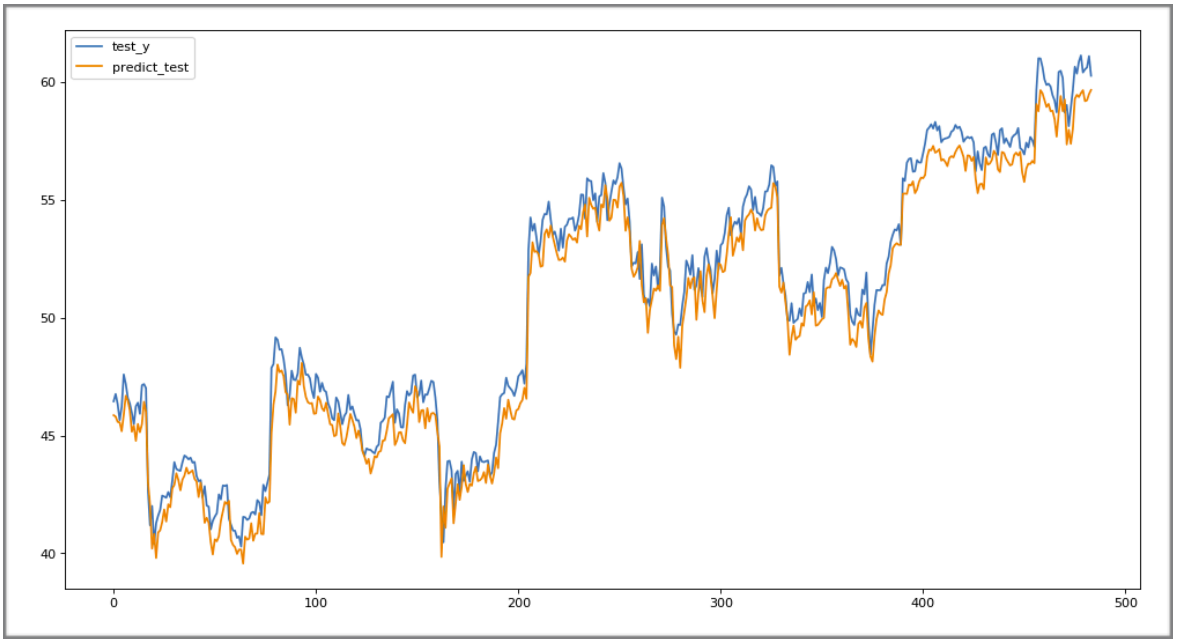
Apple model prediction on framed training

We can clearly see the improvement in the model by using the framed training, it is now able to predict the price even more accurately.

C. Separate Microsoft model with Simple training : Model is working really good in predicting the yearly pattern in the price but it has few problems in predicting the price when the actual price range changes.

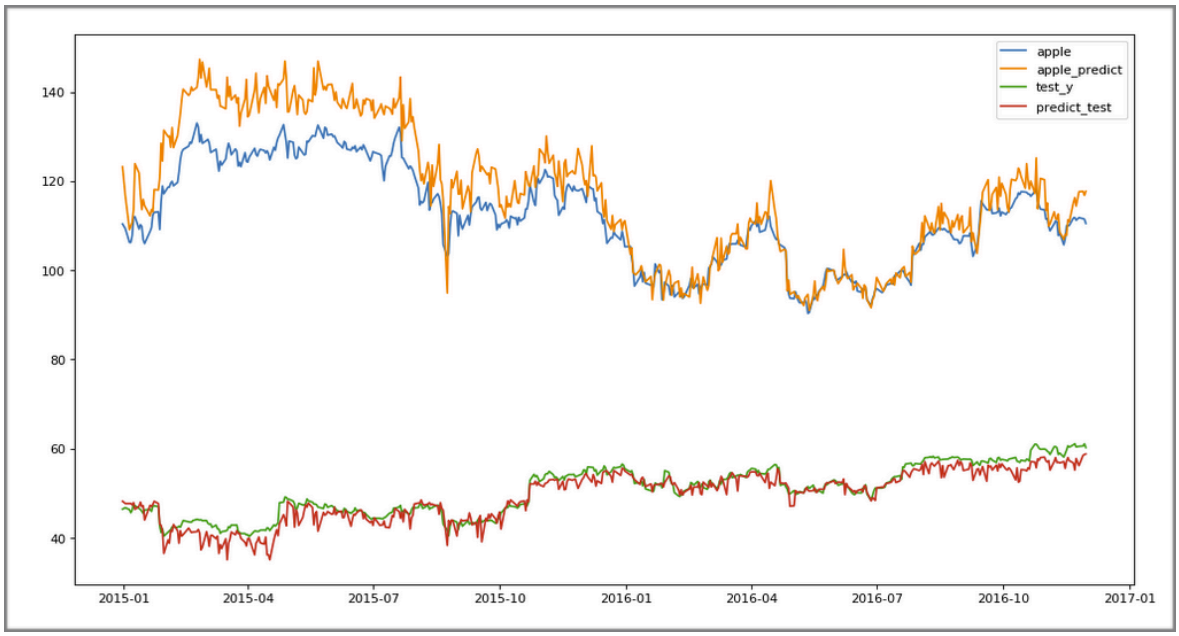
Microsoft model prediction on simple training

D. Separate Microsoft model with Framed training: Similar to the Apple Model the framed training has a better price prediction than the simple one.



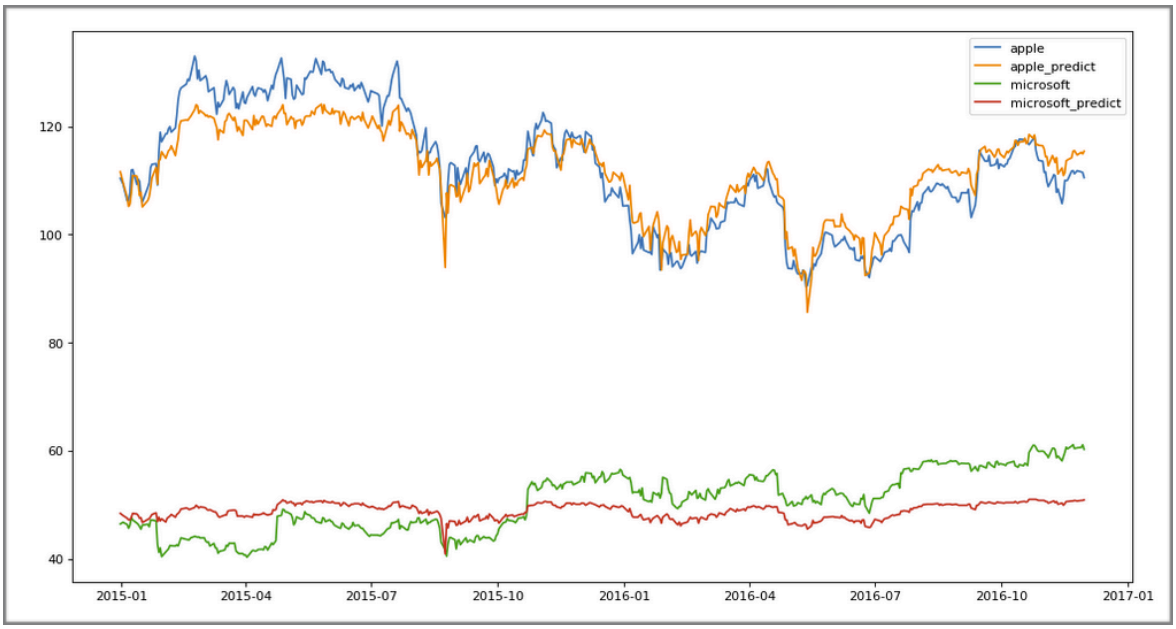
Microsoft model prediction on Framed training

E. Common Model with Simple training: I was expecting that this model will work better than the separate one but it didn't, still at-least it has done better than the benchmark model. On observation we can see that the model is able to predict the price but the predictions are not that close, but got the yearly pattern correct.



Common data model prediction on Simple training

F. Common Model with Framed training: This has even more unexpected prediction and unlike the separate model it performed poor than the simple training. But an amazing thing to notice is that the score of Microsoft prediction is increased, .i.e poor performance, on the other hand the score of Apple predictions decreased, i.e the performance increased.



Common data model prediction on Framed training

2. Reflection

The process used in this project can be summarised using the following steps:

1. Collect the Stock data.
2. Collect the News of the that stock for the specific data from stock data.
3. Preprocess the news data to have a sentiment score for that day.
4. Process the complete data to make final data set.
5. Make the benchmark model (Polynomial Regression Model).
6. Make separate model for both the companies.
7. Make a combined model for both the companies data.

Most of the interesting and time consuming part was in the 2nd, 6th and 7th. In the data collection the New York Times api really gave a hard time, as it took some time to understand the errors of the api and then process the received data.

Most interesting thing happened, while working with separate models to predict the price that is 6th, where I observed that even with sufficient data and training the model was able to predict the pattern but was not able to predict the price in the actual price range, hence I made the framed training so that it can give more preference to the latest data, such that the old data will help in predicting the pattern in the prices and the latest data will help in setting the range of the price. This concept really worked with the separate model, but not with the combined one.

3. Improvement

This is really a very wast field and even after this much of work I really believe that we have just scratched the surface of the field. There are limitless amount of data to explore and hence a lot of places to implement the machine learning.

For this specific project there are many improvements that can be made specifically in the combined data model, as I expected it to work better than the separate data model.

I have made the model to predict the the closing price of the stock on that day using just the opening price and news, and now we can work on it to make it predict the closing price of upcoming weeks or even days, and then we can actually implement it on the live data to see its performance.

In the terms Machine learning I have just used the supervised learning, but we can make complex models using Reinforcement learning to make the model predict even more better and in near future.

References

[1] Kohara, K., Ishikawa, T., Fukuhara, Y., & Nakamura, Y. (1997). Stock price prediction using prior knowledge and neural networks. Intelligent systems in accounting, finance and management, 6(1), 11-22.

[2] Padhiary, P. K., & Mishra, A. P. (2011). Development of improved artificial neural network model for stock market prediction.International Journal of Engineering Science.

[3] Jeon, S., Hong, B., & Chang, V. (2018). Pattern graph tracking-based stock price prediction using big data. Future Generation Computer Systems, 80, 171-187.

[4] Lee, M. S., Ahn, C. H., Kwahk, K. Y., & Ahn, H. (2018). Stock Market Prediction Using Convolutional Neural Network That Learns from a Graph. World Academy of Science, Engineering and Technology, International Journal of Business and Economics Engineering, 5(1).

[5] TORRES, D. G., & QIU, H. (2018). Applying Recurrent Neural Networks for Multivariate Time Series Forecasting of Volatile Financial Data.