Predicting Stock Prices using Social Media

Team: Stock Brokers

| CMPT 733 - Programming for Big Data 2 | April 08, 2018 |

By

Mihir Gajjar, Gaurav Prachchhak, Tommy, Betz, Veekesh Dhununjoy

1.0 Motivation and Background

Stock market prediction is the act of trying to determine the future value of a company's stock traded on a stock exchange. The successful prediction of future stock prices could yield significant profits.

Stock market prediction has long been practiced by investors around the world and many predictive analytic techniques are currently being used. However, very few of those methods or models have proven to be consistently accurate and reliable. The high level of inaccuracy in predicting stocks comes from the uncertainty in predicting future events pertaining to specific companies as well as a certain randomness of stock prices. Nevertheless, we believe that the level of accuracy of stock prediction can further be improved using the proper datasets, enhanced features and advanced machine learning models.

The motivation for this project arises from the fact that investors are constantly looking for an advanced model that is accurate and adaptable to market fluctuations.

There have been various attempts using different machine learning tools and techniques to predict future stock prices. We can classify these techniques into two categories:

1. **Econometric models**: This includes classical econometric models for predicting the future stock prices. Some of the common methods are the autoregressive method (AR), the moving average model (MA), the autoregressive moving average model (ARMA), and the autoregressive integrated moving average (ARIMA).

2. **Soft computing-based models**: The term soft computing covers artificial intelligence. These techniques include artificial neural networks (ANN), fuzzy logic (FL), support vector machines (SVM) and others. Various methods using deep neural networks to extract abstract features from the data have been developed.

2.0 Problem Statement

- How can a user decide where to invest?
- How do different factors like news and social media affect the stock price?
- Which feature has the most weight/importance for predicting future stock prices?
- Which news topics affect the prices/trend of the stocks for different companies?
- Which was the tweet related to a specific company that had the highest followers count for a specific day? (so that we can know the news/tweet that had the most impact for the change in the stock price)

Challenges

- The stock market is very volatile and depends on lots of different parameters hence it is very difficult to predict the future stock prices.
- The text in the news/twitter data can have sentences that contain information about various companies and not just one company. In these situations, it becomes very difficult to extract the sentiment for a

specific company. Complex natural language processing techniques are needed to get the sentiment for one company.

- To improve upon the existing models, we need to create/extract new features that would increase the accuracy of the prediction. Various statistical techniques need to be incorporated to come up with new features from the datasets.
- There have been numerous attempts using different machine learning models to predict the future stock prices. It is very important to select the right machine learning model and discover the best performing parameters for this task.

3.0 Data Science Pipeline

The data science pipeline is shown in Figure 1 below. The pipeline consists of 3 main parts namely:

1. Data Collection/Processing
2. Sentiment Analysis
3. Model Selection/Evaluation

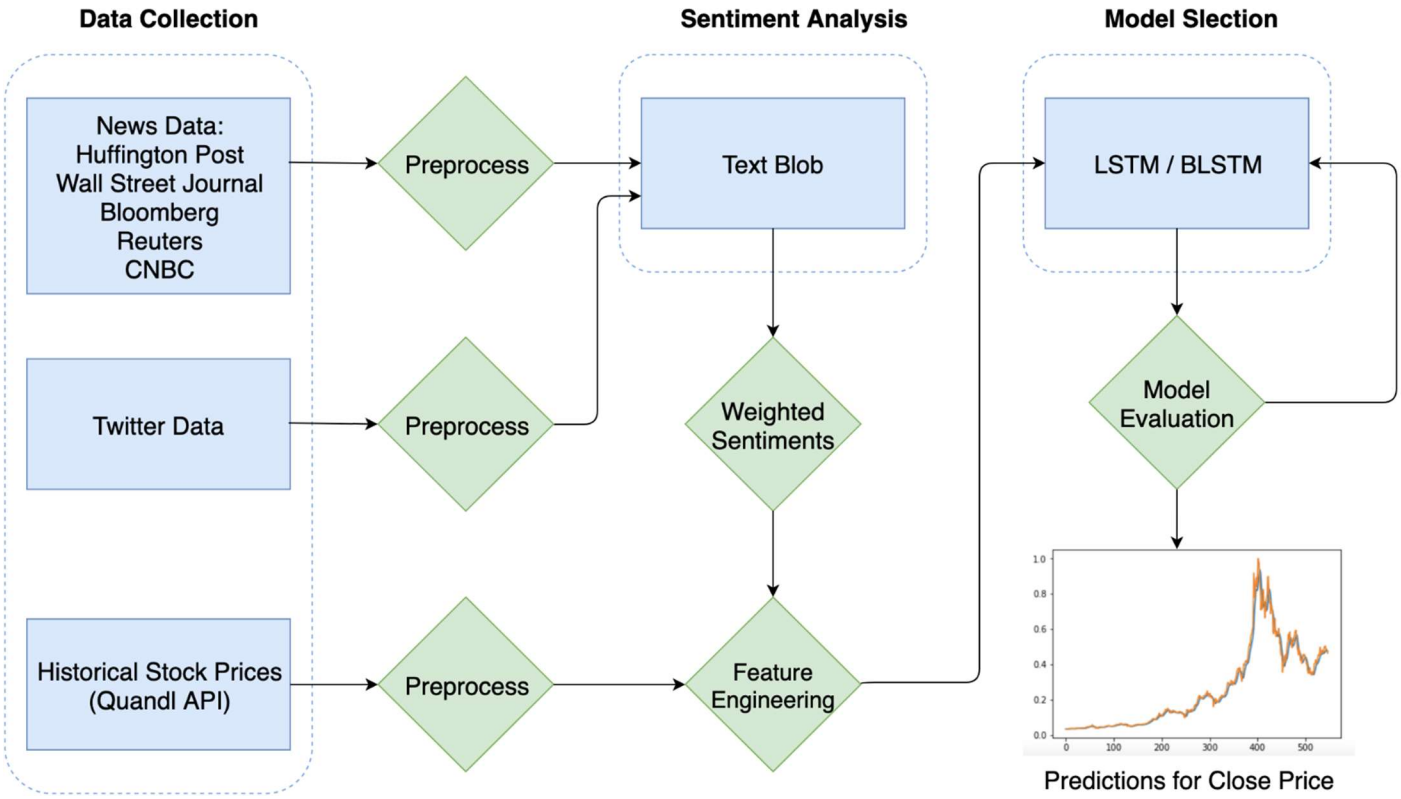


Figure 1: Data-science pipeline

3.1 Data Sources

3.1.1 Historical Stock Prices Data

We have used Quandl API to get stock data from January 2012 to March 2018. We required data for 2019 until March which was not available using Quandl hence we used Yahoo Financials wrapper to get the data for 2019 until March. The dataset contains following fields: Date, Open, High, Low, Close and Volume.

3.1.2 Twitter Data

We collected the twitter data from Internet Archive collection of twitter stream.

It contains the tweets in a nested structure as: Year → Month → Date → Hour → Minute. It is a massive dataset which contains around 3 TB of data. The average size of twitter data for one month is about 50 GB. We have spent considerable amount of time to process this data.

The dataset contains the following fields: contributors, coordinates, created_at, delete, display_text_range, entities, extended_entities, extended_tweet, favorite_count, favorited, filter_level, geo, id, id_str, in_reply_to_screen_name, in_reply_to_status_id, in_reply_to_status_id_str, in_reply_to_user_id, in_reply_to_user_id_str, is_quote_status, lang, place, possibly_sensitive, quote_count, quoted_status, quoted_status_id, quoted_status_id_str, quoted_status_permalink, reply_count, retweet_count, retweeted, retweeted_status, source, text, timestamp_ms, truncated, user.

3.1.3 News Data

We collected the News Dataset from Kaggle News Category Dataset and Kaggle US Financial News Articles.

Kaggle News Category Dataset contains approximately 200k news articles of various categories from 2012 to 2018 obtained from Huffington posts. The fields in this dataset include date, authors, category, headline, short description and link.

Since the headlines and the short description fields did not provide enough information regarding the articles, we used the link in the dataset to scrape the articles' host's website to extract the body of the news and combined it with the existing dataset.

Kaggle US Financial News Articles consists of 306242 news articles present in JSON format, dating from Jan 2018 up to end of May 2018. The news articles are from the following news publishers: Bloomberg.com, CNBC.com, reuters.com, wsj.com, fortune.com.

3.2 Exploratory Data Analysis (EDA)

3.2.1 Stock Data

False API Calls

From Figure 2 (a) below, it can be clearly seen that there are inconsistencies in the data. There is a sudden dip in the stock prices between 2014 and 2015. Figure 2 (a) contains open price of apple stock including dividend information which is not required while Figure 2 (b) contains adjusted open price without the dividend information which is required.

Generally, adjusted prices for stock market prediction are used because normal prices might have other influences such as stock splitting which may lead to a poorly performing model.

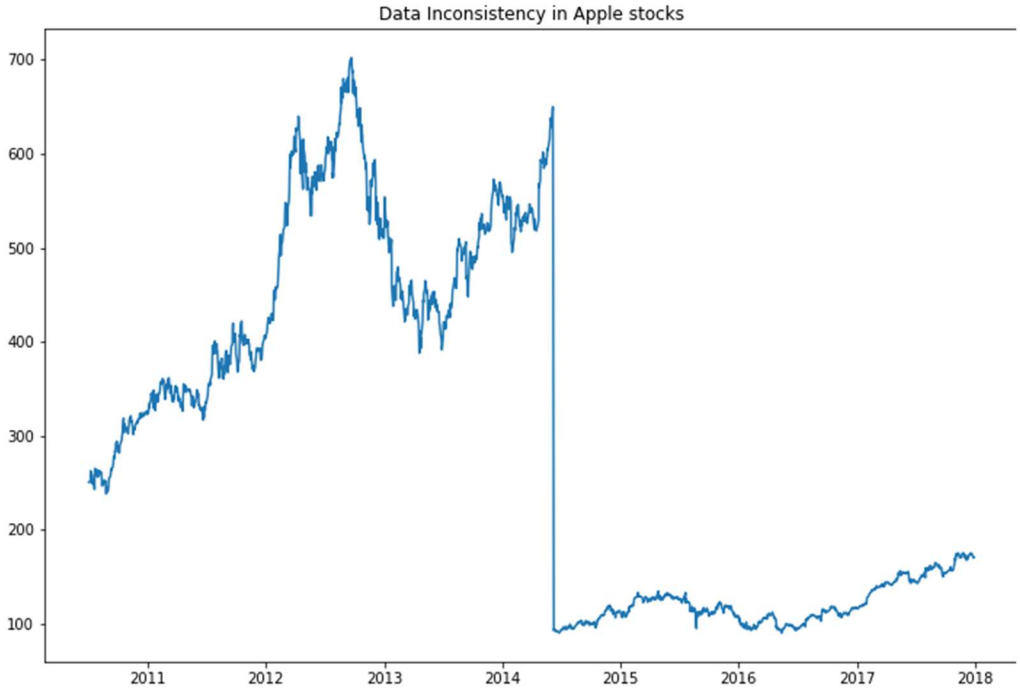


Figure 2 (a): Open price of Apple stock including dividend information from 2010 – 2018 (Inconsistencies in the data)

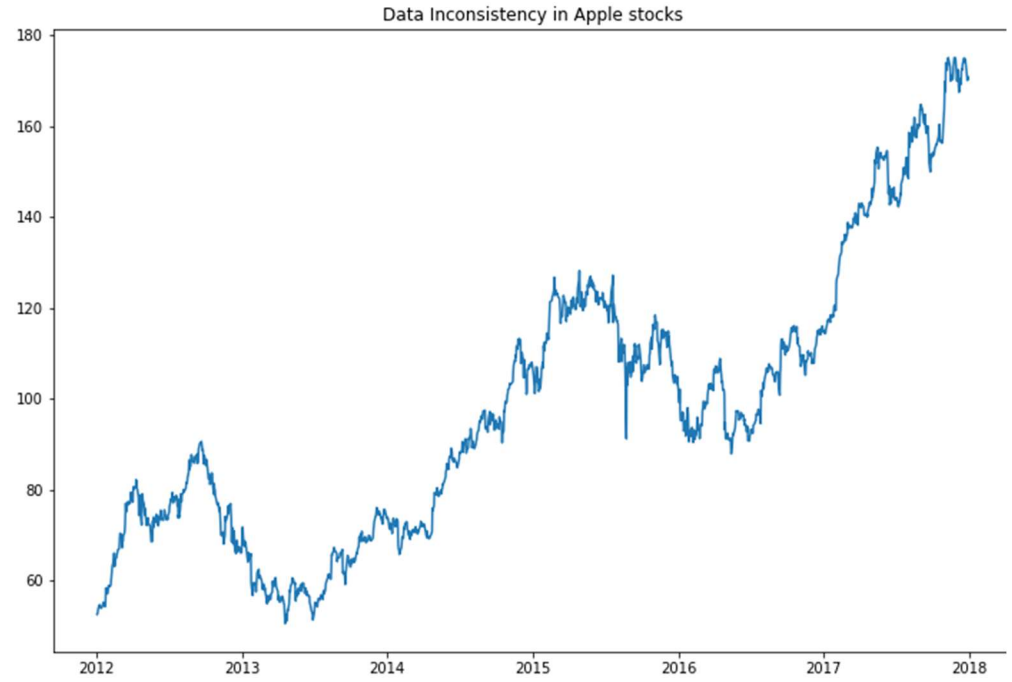


Figure 2 (b): Adjusted Open price of Apple stock excluding dividend information from 2010 – 2018

3.2.2 News Data

Count of articles based on News Categories

The Bar Chart below shows the count of articles by each category for the Kaggle News Category Dataset.

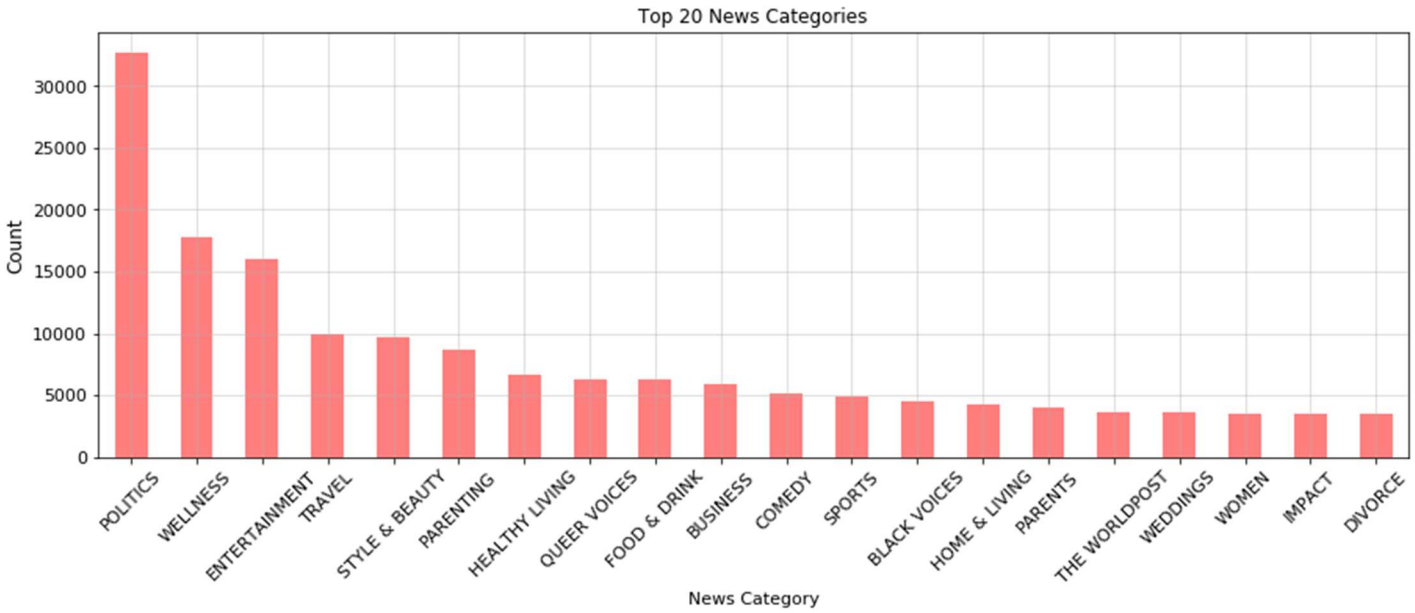


Figure 3: Top 20 news categories based on article count for Kaggle News Category Dataset

3.3 Data Pre-Processing

The data from the above datasets were not in the format that could be used for our analysis. We had to aggregate the vast amount of daily data from each individual dataset in order to obtain one record per day with only the relevant information needed. Then we merged the data using Date as the join key. The data pre-processing stage involved the following:

3.3.1 Twitter Data – Pre-Processing

The downloaded twitter datasets were uploaded to Google DataProc cluster. We read the twitter data using Spark. We processed the twitter data to extract sentiments for Apple and Facebook data only.
The data cleaning process is as follows:

1. Remove unwanted fields. We retained only the following fields: 'Date', 'screen_name', 'followers_count', 'text'.
2. We keep only the tweets in the english language and remove the language column.
3. Convert the timestamp (ms) in to Date and Time.
4. Replace the null values with 0 or empty string values so that the data can be processed further.
5. Clean tweets to remove unwanted texts, characters, symbols and emojis using regular expressions.
6. Filter tweets to retain only those tweets containing only words from APPLE_Word_List and joint_words and tweets not containing words in the APPLE_unwanted_list. This step ensures that we only get tweets that are related to Apple technologies and not apple the fruit.

APPLE_Word_List = ['apple','aapl', 'airpod', 'app store', 'earpods', 'homepod', 'imac', 'itablet', 'magic trackpad', 'magic mouse', 'iphone', 'ipad', 'ipod', 'macbook', 'ios','itunes', 'iwatch']

joint_words = ['steve jobs', 'tim cook']

APPLE_unwanted_list = ['fruit', 'cherries', 'berries', 'berry', 'food', 'pie', 'nutrition', 'pear', 'citrus', 'mango', grape', 'peach', 'apricot', 'mango', 'plum', 'avocado', 'ripe', 'melon', 'cherimoya', 'tropical', 'pineapple', 'orange', 'vegetable', 'tomato', 'fig']

1. The same logic was applied for Facebook tweets. However, we did not have negative words for Facebook since this term is not associated with any other meaning.
2. The tweets' text was then passed to a function to extract the sentiments polarity (-1 to 1) using the Python TextBlob library.
3. We then performed multiple aggregations by Date to generate the following features:
    a. Median_sentiment
    b. Count_tweet
    c. Avg_sentiment
    d. Sum_followers_count
    e. Max_followers_count
    f. Sum_favourite_count
    g. Sum_retweet_count
    h. Avg_weighted_sentiment
        i. Rather than using a plain average sentiment we put more weights on tweets with a higher follower count to put more importance on tweets that had a higher audience exposure.
    i. Sum_weighted_sentiment
4. Then we wrote the output to a csv file to be merged with the other datasets later.

The 2 Python programs used for the above preprocessing are:

1. SentimentAnalysis.py
2. SentimentAnalysis_Facebook.py

3.3.2 News Data – Pre-Processing

Once we have imported the body field from the actual link, we process the news data as follows:

1. We use the same logic as above to clean the text and check for words in the Apple and Facebook word list. We assign 'AAPL' for Apple related news and 'FB' for Facebook related articles.
2. Pass the body text into a function to extract sentiments using TextBlob.
3. Retain only the relevant news articles i.e. 'AAPL' for Apple and 'FB' for Facebook
4. Carry out aggregations by Date to generate useful features:
    a. Median_sentiment
    b. Count_news
    c. Avg_sentiment
    d. Sum_company_frequency
    e. Max_company frequency
    f. Sum_weighted_sentiment
    g. Avg_weighted_sentiment

The output data was then saved as csv files.

3.3.3 Stock Data – Pre-Processing

Since the data was already pre-filtered with important features, there was no need for any pre-processing. Both Apple and Facebook's stock data was stored in separate .csv files.

3.4 Sentiment Analysis

We performed sentiment analysis on our news data sets (Huffington Post, Wall Street Journal, Bloomberg, Reuters, CNBC) as well as the twitter dataset using the Python library TextBlob. To obtain a sentiment score we passed the whole body of a news article or the whole text of a tweet through TextBlob's API with the sentiment polarity function. The resulting score lies within a range from –1 to +1 and has the following meaning:

Score Range	Sentiment	Comment
- 1 to 0	negative	a lower score means a more negative sentiment
0	neutral	neither negative nor positive sentiment
0 to 1	positive	a higher score means a more positive sentiment

3.5 Feature Engineering

We start by having three datasets with the following features as shown in the table:

Dataset	Number of Features	Features
News	6	date, authors, category, headline, short description and link.
Twitter	37	contributors, coordinates, created_at, delete, display_text_range, entities, extended_entities, extended_tweet, favorite_count, favorited, filter_level, geo, id, id_str, in_reply_to_screen_name, in_reply_to_status_id, in_reply_to_status_id_str, in_reply_to_user_id, in_reply_to_user_id_str, is_quote_status, lang, place, possibly_sensitive, quote_count, quoted_status, quoted_status_id, quoted_status_id_str, quoted_status_permalink, reply_count, retweet_count, retweeted, retweeted_status, source, text, timestamp_ms, truncated, user.
Stock	5	Open, High, Low, Close, Volume

Then we start to generate features from the existing features. We first check whether the new feature we created are correlated with closing price or not. If the feature is correlated with the closing price, we keep the feature in the final vector, else drop this feature.

3.6 Correlation Analysis

Historic Stock Data

1) VWAP (Volume Weighted Average Price)

VWAP is calculated by adding up the dollars traded for every transaction (price multiplied by the number of shares traded) and then dividing by the total shares traded.

VWAP = Σ Price x Volume / Σ Volume

2) DOW (Day of Week)

Day of week is a feature that we thought would give good insights as there may be more activity on certain weekdays. Eg. the last trading day of week may have certain behavior that is specific to that particular day.

3) US/CAD

One other feature that we came up with was US/CAD dollar exchange price for each date. We thought as we are dealing with NASDAQ listed companies why not use the US dollar price as well because the volume and the price together results for profit or loss. And we were correct in our intuition as the results we got showed positive correlation between US dollar price and closing price.

4) Running Difference

Running difference is simply high price of a stock on a date minus the lowest price of that stock on that day. This feature captures the variation of price per day.

Spearson's correlation with Close Price	Correlation score	Correlation meaning / direction
DOW (Day of Week)	-0.001226	very weak - negative
High	0.999722	very strong - positive
Low	0.999744	very strong - positive
Open	0.999425	very strong - positive
Volume	0.204484	weak - positive
Running Difference	0.343693	weak - positive
US/CAD exchange rate	0.720408	strong - positive
VWAP(Volume Weighted Average Price)	0.856878	very strong - positive

Spearman's correlation with Close Price	Correlation score	Correlation meaning / direction
DOW (Day of Week)	-0.003859	very weak - negative
High	0.999579	very strong - positive
Low	0.999599	very strong - positive
Open	0.999114	very strong - positive
Volume	0.434197	moderate - positive
Running Difference	0.378883	weak - positive
US/CAD exchange rate	0.712201	strong - positive
VWAP(Volume Weighted Average Price)	0.915756	very strong - positive

News Data

5) Count_News

The Count_News is the count of the number of news for a company (AAPL) grouped by date.

6) Avg_Sentiment

Avg_Sentiment is the average of the body sentiments for 1 company (AAPL) grouped by day.

Spearman's correlation with Close Price	Correlation score	Correlation meaning / direction
news_avg_sentiment	-0.003421	very weak - negative
count_news	-0.199476	very weak - negative

Spearman's correlation with Close Price	Correlation score	Correlation meaning / direction
news_avg_sentiment	-0.099409	very weak - negative
count_news	-0.190386	very weak - negative

Twitter Data

7). Tweet_Count

Tweet_count is the count of the number of tweets for 1 company grouped by date.

8). Tweet_Avg_Sentiment

Tweet_Avg_Sentiment is the average sentiment of tweets for AAPL grouped by date.

Pearson's correlation with Close Price	Correlation score	Correlation meaning / direction
tweet_count	-0.091321	very weak - negative
twitter_avg_sentiment	-0.08826	very weak - negative

Spearman's correlation with Close Price	Correlation score	Correlation meaning / direction
tweet_count	-0.424871	moderate - negative
twitter_avg_sentiment	-0.086189	very weak - negative

Finally, we use the following fields as features for our model:

Dataset	Number of Features	Features
News	2	'news_avg_sentiment', 'count_news'.
Twitter	3	'tweet_count', 'twitter_sum_followers_count', 'twitter_avg_sentiment'.
Stock	9	'DOW', 'Date', 'High', 'Low', 'Open', 'Running Difference', 'US/CAD', 'VWAP', 'Volume'.

Using the above features, we predict the Closing Stock Price.

3.6 Model Selection

Various approaches such as Support Vector Machines (SVM), Convolutional Neural Networks (CNN) etc. are used to solve this problem. However, we have used Long Short-Term Memory (LSTM) for prediction because we found through research that it gives best results for time-series data.

3.7 Visualization

We have used Power BI to plot the closing price predictions of our model using different features. The graphs can be seen in Section 6.0.

4.0 Methodology

TextBlob

We have used TextBlob, a Python library as it is an API which is easy to integreate with Python and gives good results for sentiment analysis. Furthermore, TextBlob is built on NLTK, the leading platform for building Python programs to work with human language data.

Google Cloud Dataproc (Cloud-native Apache Hadoop and Apache Spark)

Google Cloud Dataproc offers a cluster preinstalled with Apache Hadoop and Apache Spark for large scale data processing. We used Dataproc's cloud infrastructure to create a cluster on multiple accounts to process more than 3TB of data.

Pandas

Pandas' Python library offers an alternative to Spark for data structures and data analysis. It is simple to use and performant when data does not exceed 3 GB in size. Furthermore, we wanted to used pandas to add an additional tool to our set of skills.

NumPy

We used Numpy for doing mathematical operations and to reshape various feature vectors into appropriate shape that works for the model.

Keras

It is high level API which uses Tensorflow as backend for deep learning stuff. It works seamlessly on CPU as well as GPU. The advantage of Keras is that it provides simple understanding of the underlying model. We used Keras to create an LSTM and BLSTM model for predicting the closing price of the stock. We found that while creating the model with Keras, the focus lies on the parameters of the model rather than on the core building blocks of any deep learning model.

Google Colaboratory (Colab)

Most of our deep learning code was written on Google colab as it provides the computing resourses (GPU) preconfigured with the notebook. It also provides most of the libraries installed in the Python environment.

Power BI

We used Power BI for Data Visualization as it is easy to setup and it supports mostly all types of visualizations.

5.0 Evaluation

The trend of the predicted stock prices is close to the trend of actual stock prices. We use average weighted sentiment than just the sentiment because average weighted sentiment adds more importance to the data that has a wider audience exposure.

In the first instance we joined all three datasets (historic stock data, news data, twitter data) with an inner join, meaning the joint table only has data for dates that are common for each individual dataset. Using this joint table, we created four different test cases as seen in the table below. We then selected only the features mentioned for each test case and fed these features to our model. This means that we used a combination of datasets to see which sources have the greatest impact on a closing stock price. Since it is an inner join table, no source has more datapoints than the other sources which ensures a fair measurement and comparison. As can be seen in the table below, using a combination of historic stock data with news data has the lowest RMSE, which means that a closing price can most accurately be predicted using these two sources.

This table show the best minimum RMSE achieved with for all four cases using the best performing parameters. In addition to the parameters listed in the table, we used these common parameters which had the best performance for all four cases: batch size = 1, dense = 1, verbose = 1, optimizer = adam, model = LSTM, dropout = no

Case	Stock Data	Twitter Data	News Data	Look Back	Learning Rate	Epochs	Min RMSE
1	✓			12	0.002	100	**5.64**
2	✓	✓		15	0.002	100	**5.78**
3	✓		✓	15	0.002	100	**4.72**
4	✓	✓	✓	12	0.002	100	**6.25**

Having found out that using historic stock data with news data has the lowest RMSE, we proceeded with training our model with as much data as we have for each test case / combination of data sources. The results are shown in the tables below. However, since the size of the datasets has changed now and the different sources will have a different number of data points, the resulting RMSE will vary depending on the number of data points. Hence, it would be unfair to compare the performance of different combination of features.

Case 1: Stock Data

Sr. No.	Epochs	Batch size	Hidden layers	Neurons	Look Back	LSTM/BLSTM	Dropout	Learning rate	Train RMSE	Test RMSE
1	100	2	1	8× 32 × 1	15	LSTM	0	0.001	2.51	13.17
2	100	1	1	8 × 32 × 1	30	LSTM	0.5	0.002	3.94	33.78
3	100	4	2	8 × 32 × 32 × 1	15	BLSTM	0	0.001	2.42	8.62
4	100	2	2	8 × 32 × 32 × 1	30	LSTM	0	0.001	2.14	22.48
5	100	4	3	8 × 64 × 32 × 32 × 1	15	BLSTM	0	0.001	5.58	7.45
6	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0.5	0.002	2.02	23.61
7	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0	0.002	2.93	21.94

Case 2: Stock Data and News Data

Sr. No.	Epochs	Batch size	Hidden layers	Neurons	Look Back	LSTM/BLSTM	Dropout	Learning rate	Train RMSE	Test RMSE
1	100	2	1	8× 32 × 1	15	LSTM	0	0.001	3.26	14.39
2	100	1	1	8 × 32 × 1	30	LSTM	0.5	0.002	4.38	27.80
3	100	4	2	8 × 32 × 32 × 1	15	BLSTM	0	0.001	2.93	11.76
4	100	2	2	8 × 32 × 32 × 1	30	LSTM	0	0.001	2.34	16.31
5	100	4	3	8 × 64 × 32 × 32 × 1	15	BLSTM	0	0.001	2.65	13.44
6	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0.5	0.002	2.14	21.29
7	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0	0.002	2.27	13.90

Case 3: Stock Data and Twitter Data

Sr. No.	Epochs	Batch size	Hidden layers	Neurons	Look Back	LSTM/BLSTM	Dropout	Learning rate	Train RMSE	Test RMSE
1	100	2	1	8× 32 × 1	15	LSTM	0	0.001	3.22	7.28
2	100	1	1	8 × 32 × 1	30	LSTM	0.5	0.002	4.25	12.41
3	100	4	2	8 × 32 × 32 × 1	15	BLSTM	0	0.001	3.73	9.89
4	100	2	2	8 × 32 × 32 × 1	30	LSTM	0	0.001	3.52	10.52
5	100	4	3	8 × 64 × 32 × 32 × 1	15	BLSTM	0	0.001	4.20	6.74
6	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0.5	0.002	1.83	11.10
7	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0	0.002	2.51	10.04

Case 4: Stock Data, Twitter Data and News Data

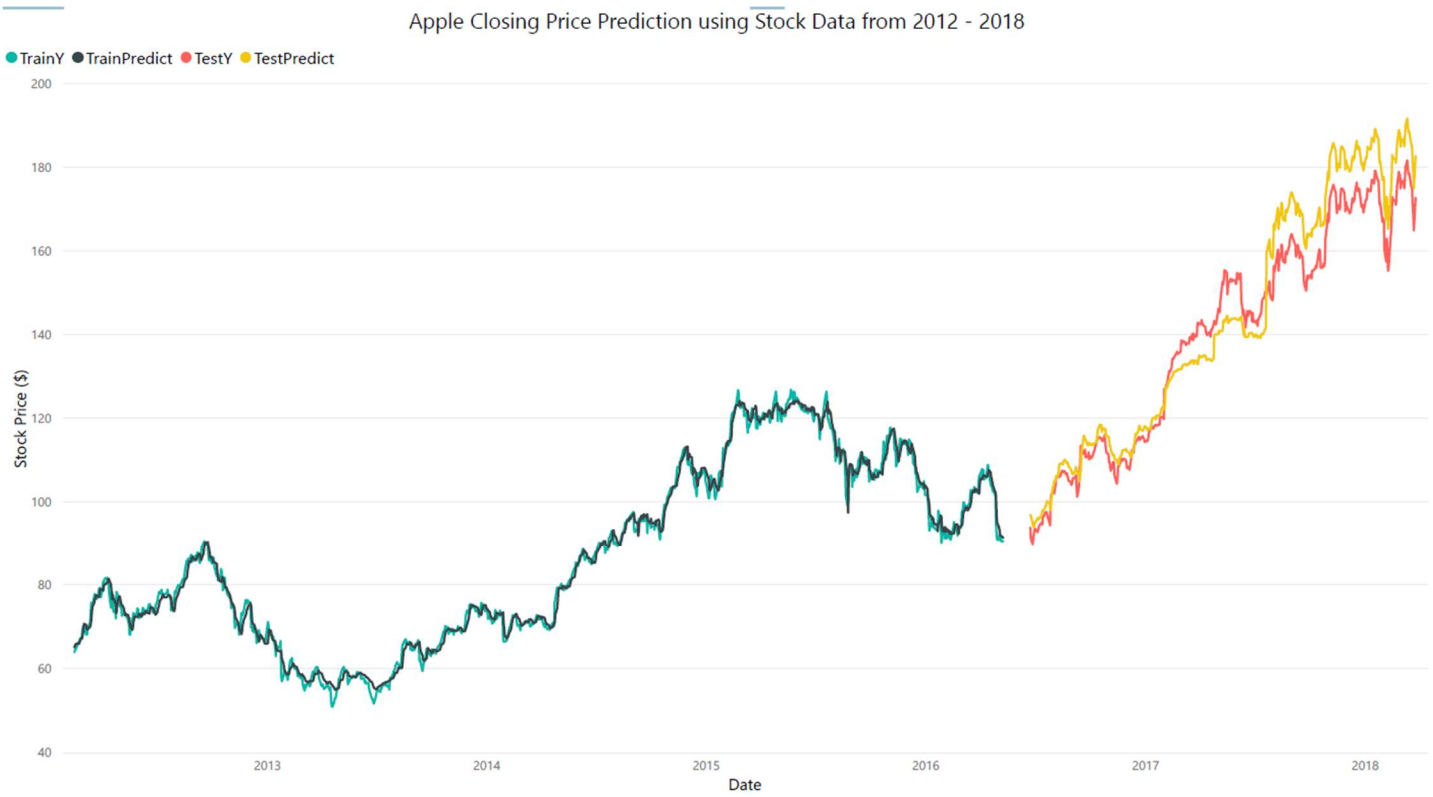
Sr. No.	Epochs	Batch size	Hidden layers	Neurons	Look Back	LSTM/BLSTM	Dropout	Learning rate	Train RMSE	Test RMSE
1	100	2	1	8× 32 × 1	15	LSTM	0	0.001	4.11	12.48
2	100	1	1	8 × 32 × 1	30	LSTM	0.5	0.002	4.47	16.49
3	100	4	2	8 × 32 × 32 × 1	15	BLSTM	0	0.001	4.59	12.40
4	100	2	2	8 × 32 × 32 × 1	30	LSTM	0	0.001	4.20	13.75
5	100	4	3	8 × 64 × 32 × 32 × 1	15	BLSTM	0	0.001	4.01	9.35
6	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0.5	0.002	1.45	9.76
7	100	4	3	8 × 64 × 32 × 32 × 1	30	BLSTM	0	0.002	3.90	15.77

6.0 Data Product

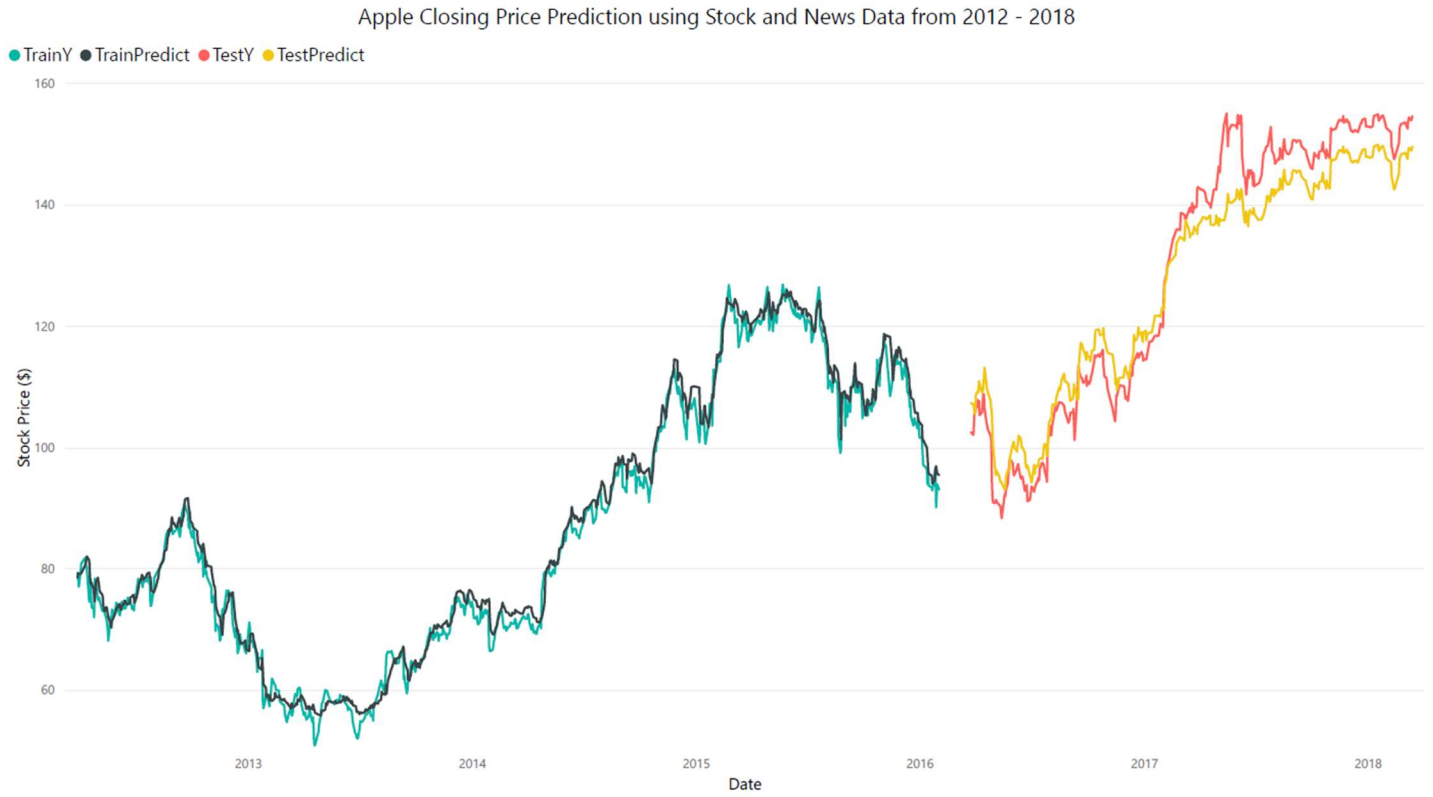
We trained a LSTM/BLSTM model which takes as an input the generated features from the Stock, Twitter and News dataset and predicts the stock prices for the next day by using the past n days information.

As the model is trained for the past years, the model can predict the future stock prices. Finally, closing price predictions using combination of different features can be seen in the below figures.

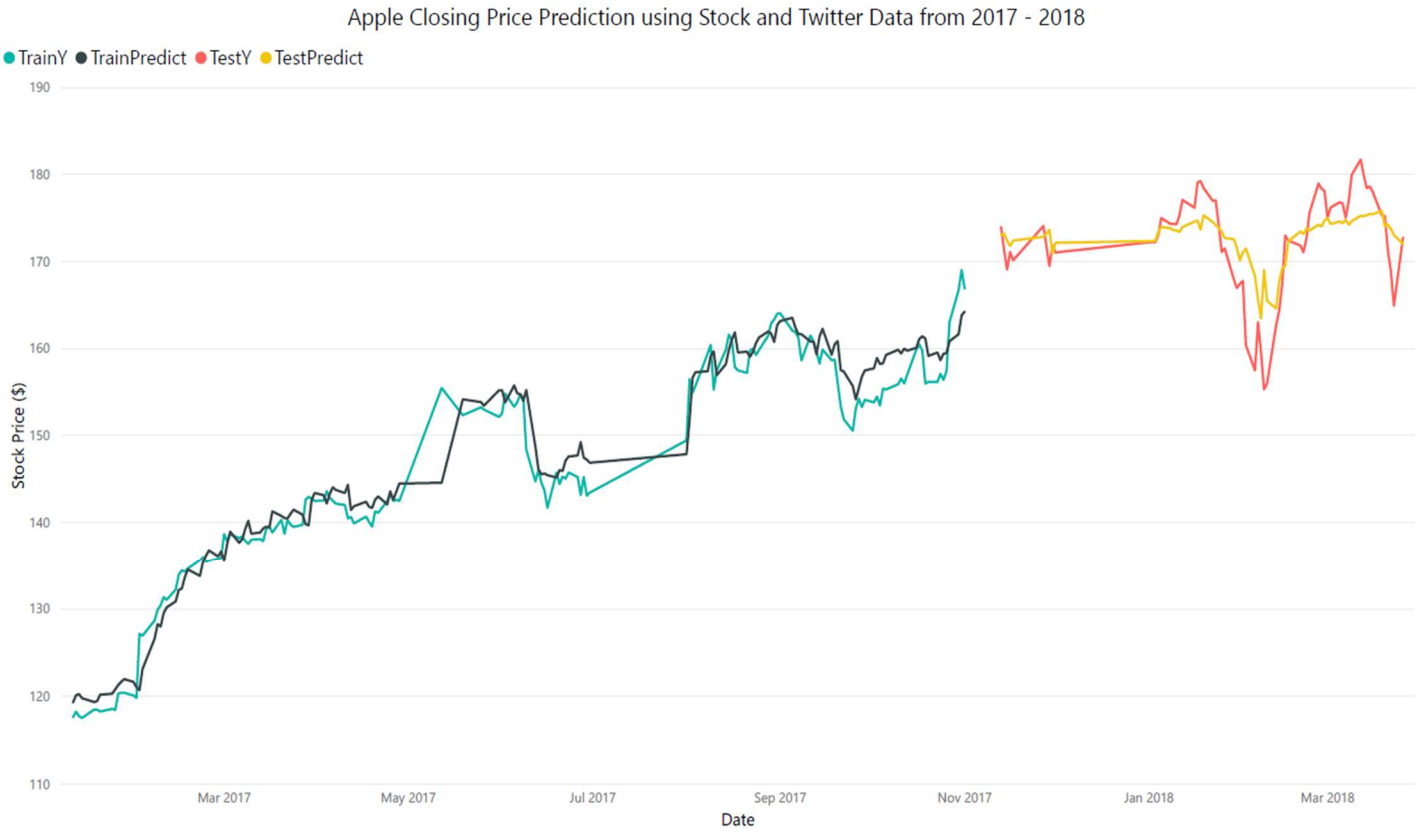
Case 1: Historical Stock Data



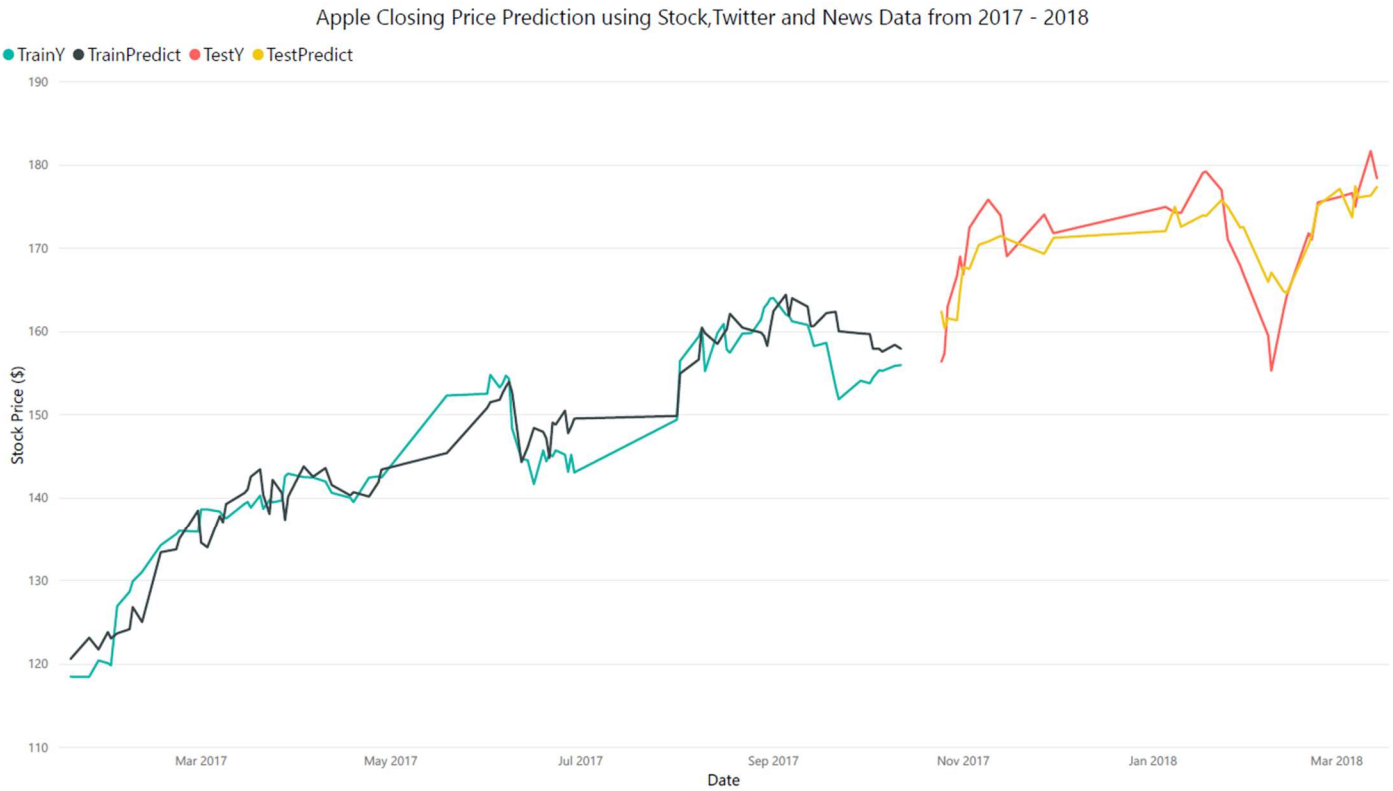
Case2: Historical Stock Data & Twitter Data



Case 3: Historical Stock Data & News Data



Case 4: Historical Stock Data, Twitter Data & News Data



7.0 Lessons Learnt

1. Processing massive datasets (3 TB) in a distributed environment using Google Dataproc cluster.
2. Extracting features and aggregating large datasets using various data wrangling techniques.
3. Creating features that are important for predictions for a model in the stock market domain.
4. Performing feature engineering to increase the value of a feature by combining multiple features.
5. NLP techniques to extract features like sentiments from text data.
6. Creating LSTM and BLSTM models over time series data for future price prediction using Keras.
7. How to decide which feature has the most impact on the prediction.
8. With the help of our Instructors and TAs we were able to learn more about the stock market field and how the data of this domain is dependent on various features such as VWAP and sentiments.

8.0 Summary

We predict the future closing stock price using historical stock data in combination with the sentiments of news articles and twitter data. We collected the historical stock price, twitter and news data by web scraping and through various data sources.

In the preprocessing stage, we filter the unwanted records and carry out aggregations to extract useful features. Sentiment analysis has been performed using TextBlob on the news and twitter data to generate weighted average sentiments. By using various statistical techniques, we generate new features using the stock data. Using the 'Date' field we combine all the features. The usefulness of the features is validated by performing correlation analysis. After performing feature engineering, we provide these features as an input to our LSTM model and predict the future closing stock prices.

The best results were obtained by using the stock data along with the new data. On the other hand, when including twitter sentiments, the error was higher which indicates that the vast number of tweets were not directly related to Apple's success which can interfere with predictions.

To summarize our results:

- We got the best result when we used a combination of Stock and News Data.
- Twitter Data did not provide better stock predictions.
- Including Twitter data with Stock and News data combined had a negative effect on RMSE.
- A Learning Rate of 0.002 was found to be most effective.
- A lookback of 15 was found to be the best given our data sources.
- Advanced NLP techniques can be used to improve the model predictions using twitter data.

9.0 References

- Advances in Financial Machine Learning
- Datacamp – Stock Market Prediction
- Github Repositories – LinuxIsCool, Bidirectional LSTM