Bayesian Data Analysis
Project Report
on
Stock Price Prediction
using Naïve Bayes Model
by
Shivika Malik

URL to the video: https://drive.google.com/file/d/1WblcQD5qT9_Hs1VzmxsMVbyJ_iB1wRoO/view

ABSTRACT:

The stock market is the most popular investing places for users. Because of its expected high profit. Recently forecasting stock market returns gaining more attention. The prediction of stock markets is regarded as a challenging task. Data analysis is the way of predicting future value. if future stocks prices will increase or decrease. The main objective of this paper is to predict future stock price using prediction concept. In that Parse Records then calculate predicted value and send to user. And automatically perform operations like purchase and sale shares using Automation concept. For that use Naïve Bayes Algorithm. There is Real time Access by Download log forms yahoo finance website and Store in dataset.

PROBLEM STATEMENT:

The current methods of predicting stock market prices are not as accurate as investors might need them to be. These current methods only incorporate linear regression in their prediction. The Efficient Market Hypothesis (EMH) states that stock market prices are largely driven by new information; thus, there is need to incorporate public sentiments when coming up with a prediction model for stock market prices.

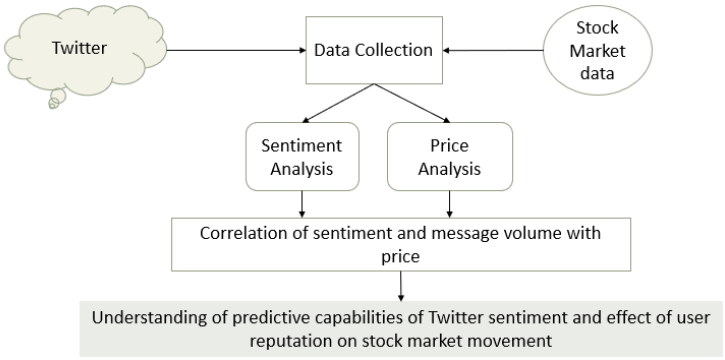
PROPOSED APPROACHES:

The problem statement stated before was briefly discussed, here in this section we will discuss about how to tackle those problem and will discuss about proposed solution for problem statement. Our project mainly has goals.

1. Collect data from twitter and finance data from national stock exchange.
2. Process and clean tweet corpus.
3. Generate feature vector and train data from given text document.
4. Analyze and predict tweets sentiment using support vector machine, Vader sentiment analyzer & Naïve Bayes techniques.
5. Merge data for both tweets sentiment and stock data.
6. Predict the stock market trend from twitter sentiment using linear regression technique and fining correlation between sentiment and stock prices.

SYSTEM DESIGN & IMPLEMENTATION:

- **Architecture**



The architecture design for the system is simple and robust which does its purpose without any problems. This architecture design was chosen for the project because it is most commonly used design and has simple and easy to implement functions and libraries which will not cause any problem when running on older computer versions.

- **Datasets**

Our system mainly consists of two datasets, one from twitter which are the tweets for companies which were saved in a separate csv file, tweets are used for analyzing emotion of users about companies and how market is talking about it, and next dataset is from National Stock Exchange which provides data about open and close price of companies, in both datasets task are done separately and analyzed until they are ready for merging, then these two datasets are merged as a single csv file in which final prediction is done and later correlation is found between them.

- **Major Components**

a) Tweets Collection

For tweets collection, Twitter provides robust API, there are two ways to get this done via Twitter streaming API and Twitter REST API, we used REST API as it allows to find tweets related to a query of recent tweets. The request of JSON object contains the tweets and their metadata, which includes many information that is time of tweets, location where tweets was written, retweets etc., our focus was on tweets text and time when it was made. API requires users to have API key which can be obtained by twitter developer website. The text of tweets contains too much extraneous words which are not consider in part of sentiment they are mainly URLS, tags, to obtain correct sentiment we have to filter those tweets or remove noisy words.

b) Tweets Pre-Processing

There are number of steps we can achieve this, and they are as follows:

- First step is to split the text by space, which forms a list of words for each text and they are called feature vectors or most occurring word in a tweet, which will be used later to train data for support vector model,
- Next step is to remove stop words from tweets, python library known as NLTK is used for this purpose.

- Stop words contains articles, punctuation, and few other words, which do not pose any sentiment in a tweet and should be removed. Stop words list is stores in a dictionary which check each tweet with the stop words and if tweets contain those words it will be removed immediately, and tweets will be filtered.
- Tweets also have some extra symbols like "@", "#", and URLS, any text next to "@" symbol is a username of user who is writing a tweet which does not add any purpose in sentiment and therefore should be removed or stored in different file.

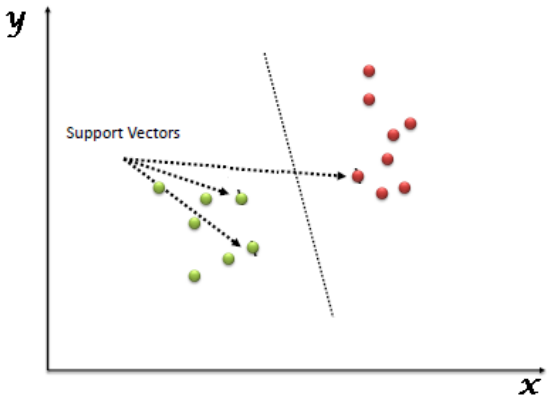
c) Sentiment Analysis

Sentiment analysis was first major part of our system as this will allow us to compare stock price. Sentiment analysis is simple process of determining whether a piece of text is positive, negative or neutral. Major approaches take one of two form, polarity based, or valance based, where intensity is considered. For example, the words 'good' and 'excellent' would be treated the same in a polarity-based approach, whereas 'excellent' would be treated as more positive than 'good' in a valence-based approach. Both have their own function which make them very effective tool for sentiment analysis. We will discuss about both in detail.

- **Vader sentiment analyzer** is known as Valence aware dictionary and sentiment reasoner. It is a powerful tool in python. It belongs to the type of sentiment analysis that is based on lexicons of sentiment-related words. In this approach, every word in the lexicon is rated as positive negative. Vader analyze a text and then check for words which are in lexicon and they have a rating which is given automatically. Vader produces four sentiment metrics from these words ratings which are positive, negative, neutral and compound. Vader is great for social media text which that is because tweet text contains words or symbols which can produce unnecessary information and can change sentiment of tweet from one way to other, Vader handles this sort of problem by adding such symbols in lexicon. Firstly, Vader package was installed into the project folder and then using one function called as SentimentIntensityAnalyser, it is an object from Vader package. Finally, we will use polarity_scores() method which again from Vader sentiment which provides metrics for a piece of text.

```
def sentiment_cal(tweet):
    value = SentimentIntensityAnalyzer()
    score = value.polarity_scores(tweet)
    score = float(score['compound'])
    return score
```

- **Support vector machine** (SVM) is a supervised machine learning algorithm, which can be used for both regression and classification, however it is used mainly in classification problems. In the SVM model we plot data points on a n-dimensional space where n represents the number of features you have and then we perform classification by finding the hyper-plane that differentiate the two classes.



Support vectors are best way to separate two classed. Scikit-learn is a widely used and popular python library. Gather a large tweet corpus is the foremost task as we need to separate classes in there, after collecting large set of tweets we need to manually label tweet text and store them in a text file. After gathering tweets, we must build and train a classifier for sentiment analysis, for classifier we then find feature vector for each piece of text. To process each text, we process and find meaningful feature whose frequency is more than 20 and create an array call X= [] and y= []. Next, we use 10-fold cross validation to determine best w and b for the support vector equation which will determine the how two class will be get separated. Similarly, we need a testing data set which will help us to analyze data from training set and predict tweet sentiment. Lastly, we used metrics classification report to generate result for precision and recall.

- **Naïve Bayes Model**

  Multinomial Naive Bayes classification algorithm tends to be a baseline solution for sentiment analysis task. The basic idea of Naive Bayes technique is to find the probabilities of classes assigned to texts by using the joint probabilities of words and classes.

  Given the dependent feature vector (x_1, …, xn) and the class Ck. Bayes' theorem is stated mathematically as the following relationship:

$$P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k)P(x_1, \ldots, x_n \mid C_k)}{P(x_1, \ldots, x_n)}$$

  According to the "naive" conditional independence assumptions, for the given class Ck each feature of vector xi is conditionally independent of every other feature xj for i≠j..

$$P(x_i \mid C_k, x_1, \ldots, x_n) = P(x_i \mid C_k)$$

  Thus, the relation can be simplified to

$$P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i \mid C_k)}{P(x_1, \ldots, x_n)}$$

Since P(x_1, …, xn) is constant, if the values of the feature variables are known, the following classification rule can be used:

$$P(C_k \mid x_1, \ldots, x_n) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_k P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

We used MultinomialNB from sklearn.naive_bayes package of scikit-learn for Naive Bayes classification. We used Laplace smoothed version of Naive Bayes with the smoothing parameter α set to its default value of 1. We used sparse vector representation for classification and ran experiments using both presence and frequency feature types. We found that presence features outperform frequency features because Naive Bayes is essentially built to work better on integer features rather than floats. We also observed that addition of bigram features improves the accuracy. We obtain a best validation accuracy of 79.68% using Naive Bayes with presence of unigrams and bigrams.

```
Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

             precision    recall  f1-score   support

          0     0.7397    0.7941    0.7659     37078
          1     0.7759    0.7183    0.7460     36792

avg / total     0.7577    0.7564    0.7560     73870
```

d) Stock Price Analysis

We simply took data from National stock exchange and proceed further with it, as NSE data is robust and we can trust it without any issues as that data is used by many users and companies.

e) Correlation

Correlation, in the finance and investment industries, is a statistic that measures the degree to which two values move in relation to each other. Correlation is calculated which is know as correlation coefficient, whose value falls between -1 to 1. 1 mean a perfect positve correlation which means that if one asset moves up or down, other asset will move in lockstep with it, in the same direction. -1 means a perfect negative correlation which means that if one asset move up or down, other asset will move in opposite direction from it, and there is a zero correlation sometimes which means there is no relation between two.

This was the final step in our system, after completing all the data collection and pre-processing, we have to merge both the data from sentiment analysis and NSE data which is quiete a rigorous task as both the data does not have any thing in common except the date, so we decided to find some relation among two data and got a hit. Stock market opens at 9AM and closes at 4PM, so we made a simple calculation that we can compute average

sentiment for each day that is when is the first tweets came in morning about 9AM and when was the last tweet for the same company that is 4PM, we later calculate open_score and close_score for tweets that were written between these two timelines and then it became easily to relate two file.

f) Implementation Details

Our application has mainly four files which needs to be run one at a time. The program will run in following manner.

• Get tweets from twitter using REST API, and store data in csv file which is easy to work on, the csv file contains time, username, tweet text, company name, sentiment, confidence and date, initially confidence and sentiment will be NULL as we will compute it in next program code.

• Next step is to implement sentiment analysis on csv file made earlier, tweets are process and the result is stored in separate csv file which contains date, time, username, sentiment.

• We find average sentiment for each day where we change the tweets sentiment data to form a relation with other dataset which is from NSE and this data is stored in different csv file, which has three columns, open_score, close_score and data.

• Next, we have a merge data from both data files, this was done manually, and this data is saved in a separate csv file, which contains data from both average sentiment file and NSE data.

• Process tweets remove stop words and create feature set for training data.

• Classify training data and test classified data model on test data for tweet sentiment analysis.

• Find correlation between sentiment of tweets and stock prices.

RESULTS & ANALYSIS:

We first collected tweets using twitter REST API, which responses in JSON format, we need authentication to fetch data from twitter, and need following configuration for it.

```
{
        “consumer_key”: “”,
        “consumer_secret”: “”,
        “access_token”: “”,
        “access_token_secret”: “”
}
```

After fetching data from twitter, we need to store that data in a file for which we used comma separated value file by Microsoft office excel. The next step was to process tweet for which we had some basic steps and they are as follows:

1. Tokenize tweets
2. Remove extra keywords from tweets
3. Remove stop words from tweets

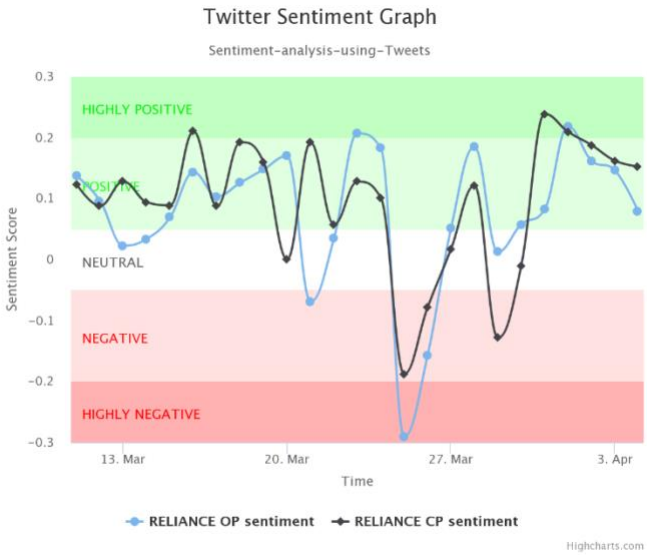
After fetching data and preprocessing, now comes task to do twitter sentiment analysis and storing the data in a separate CSV file which can be used later for computing average sentiment for tweets. Following is a small portion from that file. The generated tweets csv file contains company name, date, time, username and sentiment.

	company	date	time	username	sentiment
0	Reliance	########	6:01	RshDvvrma	0.1027
0	Reliance	########	6:03	NewsBossIndia	0.3818
0	Reliance	########	6:04	EtemaadDailyNew	0
0	Reliance	########	6:05	shyamjiagrawal	0.3818
0	Reliance	########	6:05	Rampraw4121	0.1027
0	Reliance	########	6:06	AjNewspaper	0.3818
0	Reliance	########	6:06	thus_spake	0.1027
0	Reliance	########	6:06	alan_abraham8	0.1027
0	Reliance	########	6:06	ravindarthakur0	0.1027
0	Reliance	########	6:08	Rampraw4121	0.1027
0	Reliance	########	6:09	BroderkMarshall	0.3818
0	Reliance	########	6:10	zithastechno	0
0	Reliance	########	6:11	IndiLeak	0.3818
0	Reliance	########	6:12	SubratN	0

We then started implementing our next task which was to find average sentiment for each day which was stored in a separated CSV file, following is an example from that file.

close_score	date	open_score
0.12312973	########	0.136453704
0.087479012	########	0.095978947
0.127274074	########	0.022238614
0.093629201	########	0.033530404
0.088488556	########	0.068960784
0.209550361	########	0.142469877
0.086928977	########	0.103239024
0.191771865	########	0.125527311
0.158684524	########	0.1474
0	########	0.169215217
0.191236019	########	-0.069225828
0.057127092	########	0.034364609
0.127134896	########	0.206134987
0.100737838	########	0.182294865
-0.187957585	########	-0.290353573
-0.078752041	########	-0.157434361

The above results indicated that the expected output is same what was expected, and we can move further with our application, we made a graph for this example which was easy to visualize.



Here we can see sentiment score for open price and close price.

For the next task we had to manually merge two files which as when trying to do it via code we ran into some problems and somehow data could not be merged so we decided to do it using manually. Doing this was not a difficult task as we just must match data and put data accordingly into their field and this was done quickly, following is the example from that file.

close_score	date	open_score	Close	Date	High	Low	Open
0.093629201	########	0.0335304	1289.5	########	1319	1285.25	1318.75
0.088488556	########	0.0689608	1304.95	########	1316.3	1290.4	1291.05
0.209550361	########	0.1424699	1297.65	########	1310.4	1293.6	1310
0.086928977	########	0.103239	1300.7	########	1319.95	1298.05	1308
0	########	0.1692152	1280.8	########	1306.25	1278.35	1306
0.191236019	########	-0.069226	1263.8	########	1283.9	1259.3	1282.2
0.057127092	########	0.0343646	1259.7	########	1265.9	1246.55	1252
0.127134896	########	0.206135	1273.3	########	1277.55	1258	1263.15
0.100737838	########	0.1822949	1286.75	########	1292	1268.45	1274.1
0.016734194	########	0.0506032	1251.1	########	1278.75	1247.2	1271.1
0.120644571	########	0.1841203	1245.75	########	1264	1242.1	1258
-0.12878068	########	0.0120903	1256.65	########	1260	1233.35	1251.7
-0.01074244	########	0.0571418	1270.65	########	1274.75	1253	1255
0.238220802	########	0.0816654	1320.9	########	1337.65	1266	1266
0.161634081	4/3/2018	0.1458255	1374.65	4/3/2018	1380.5	1337.05	1342

Left three columns are from average sentiment file and right five columns from National Stock Exchange dataset. This was the final step for all the data collection and processing which was completed for checkpoint 2 and now we can start using python techniques for predicting stock market movement using twitter sentiment analysis.

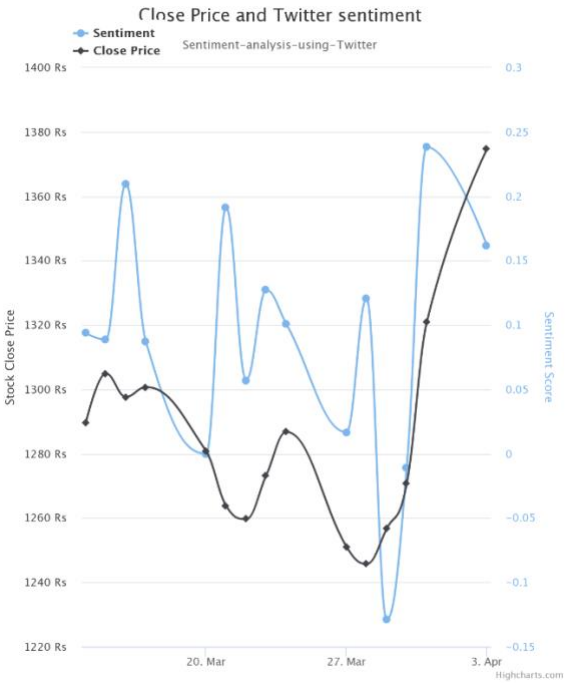
Last step was to analyze this data, and this was achieved by using linear regression, but first we found correlation between two values that is open_score from sentiment analysis and open_price from NSE dataset and similarly for close price and following are result from that analysis.

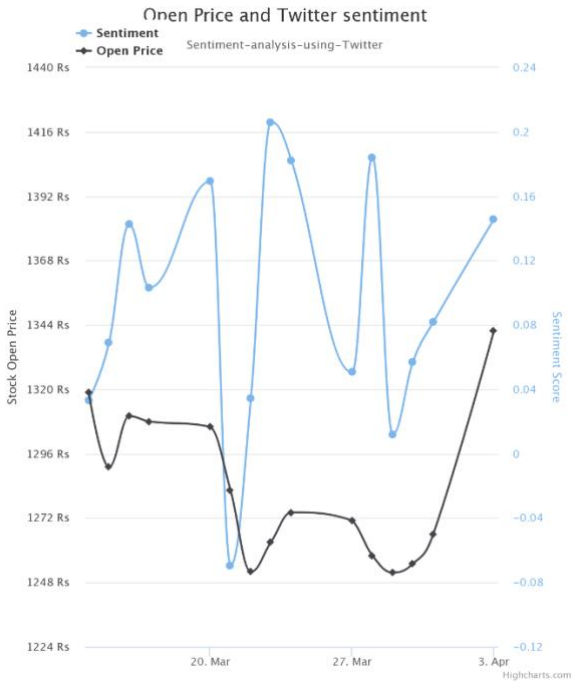
```
('Open price correlation:', 0.16501586120961761)
('Close price correlation:', 0.4749685257404374)
```

Here, we can see that the value for correlation is positive which means that people emotions and actual value are moving in same direction which can be either up or down it depends on the share values of that company in market. Last step was to find accuracy which was achieved by using linear regression and following are the result from it.

```
('Traditional Method Accuracy for Open Price: ', 97.2293482223652)
('Sentimental Method Accuracy for Open Price: ', -67.309776790104)
('Hybrid Method Accuracy for Open Price: ', -1987.9226003079382)
('Traditional Method Accuracy for Close Price: ', 8.630060243282312)
('Sentimental Method Accuracy for Close Price: ', -148.40353450931948)
('Hybrid Method Accuracy for Close price: ', -49.450841170768435)
```

We have three different accuracy methods implemented which makes it easy to determine which method is producing more accurate data and how value can fluctuate.
Following are the graphs for correlation between prices.





RECOMMENDATION SYSTEM:

We also implemented recommendation system for our application which is a subclass of information filtering system that seeks to predict the ‘rating’ or ‘preference’ a user would give to an item. There are many extensive classes of web application which involved predicting user responses to options. Recommendation system uses number of technologies and we can classify these into two broad categories.

1. Content-based system: it examines properties of the item recommended
2. Collaborative filtering is a system which recommend item based on user response to some previous interaction with data source

In recommendation system we do not consider any historical data or data from other user, we only look interaction of a user with itself. For our system we focused on collaborative filtering and the techniques used is call Association rule mining which a rule-based mining machine learning method for discovering interesting relation between variable in a dataset.

Some Useful concept:

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest are used. The best-known constraints are minimum thresholds on support and confidence.

Let be an itemset, an association rule and a set of transactions of a given database.

1) Support

Support is an indication of how frequently the itemset appears in the dataset. The support of with respect to is defined as the proportion of transactions in the dataset which contains the itemset X.

In the example dataset, the itemset X= {beer, diaper} has a support of 0.2 since it occurs in 20% of all transactions (1 out of 5 transactions). The argument of supp () is a set of preconditions, and thus becomes more restrictive as it grows (instead of more inclusive).

2) Confidence

Confidence is an indication of how often the rule has been found to be true. The confidence value of a rule, X=>Y , with respect to a set of transactions T , is the proportion of the transactions that contains X which also contains Y.

Confidence is defined as:

Conf(X=>Y) = supp(XUY)/supp(X).

For example, the rule {butter, bread} => {milk}has a confidence of 1.0 in the database, which means that for 100% of the transactions containing butter and bread the rule is correct (100% of the times a customer buys butter and bread, milk is bought as well).

For our system we used raw data which consist of all the rules.

```
raw_data = ["Bajaj-Auto, Reliance, Heromotocorp, BajFinance",
            "Tata-Motors, Heromotocorp, Maruti, Axisbank",
            "BajFinance, Reliance, Maruti, Cipla",
            "Bajaj-Auto, ITC, Reliance",
            "LT, Heromotocorp, Tata, Cipla, Wipro",
            "Reliance, Maruti, BajFinance, Tata-Motors"]
```

We used orange package from python which is interactive data analysis tool, we stored this raw data into a file whose extension is given as file_name.basket, we then load data from this file and apply function from orange library which is Orange.associate.AssociationRulesSparseInducer(), which will produce result and below is the result from it.

```
Supp Conf  Rule
 0.3  0.7  Maruti -> Tata-Motors
 0.3  1.0  Tata-Motors -> Maruti
 0.3  0.7  Maruti -> BajFinance
 0.3  0.7  BajFinance -> Maruti
 0.3  0.7  Maruti -> BajFinance Reliance
 0.3  1.0  Maruti BajFinance -> Reliance
 0.3  1.0  Maruti Reliance -> BajFinance
 0.3  0.7  BajFinance -> Maruti Reliance
 0.3  0.7  BajFinance Reliance -> Maruti
 0.3  0.5  Reliance -> Maruti BajFinance
 0.3  0.7  Maruti -> Reliance
 0.3  0.5  Reliance -> Maruti
 0.5  1.0  BajFinance -> Reliance
 0.5  0.8  Reliance -> BajFinance
 0.3  0.5  Reliance -> Bajaj-Auto
 0.3  1.0  Bajaj-Auto -> Reliance

User 0: Bajaj-Auto, Reliance, Heromotocorp, BajFinance
 0.3  0.7  BajFinance -> Maruti
 0.3  0.7  BajFinance -> Maruti Reliance
 0.3  0.7  BajFinance Reliance -> Maruti
 0.3  0.5  Reliance -> Maruti BajFinance
 0.3  0.5  Reliance -> Maruti

User 1: Tata-Motors, Heromotocorp, Maruti, Axisbank
 0.3  0.7  Maruti -> BajFinance
 0.3  0.7  Maruti -> BajFinance Reliance
 0.3  0.7  Maruti -> Reliance

User 2: BajFinance, Reliance, Maruti, Cipla
 0.3  0.7  Maruti -> Tata-Motors
 0.3  0.5  Reliance -> Bajaj-Auto

User 3: Bajaj-Auto, ITC, Reliance
 0.3  0.5  Reliance -> Maruti BajFinance
 0.3  0.5  Reliance -> Maruti
 0.5  0.8  Reliance -> BajFinance

User 4: LT, Heromotocorp, Tata, Cipla, Wipro

User 5: Reliance, Maruti, BajFinance, Tata-Motors
 0.3  0.5  Reliance -> Bajaj-Auto
```

CHALLENGES:

During the implementation of this system we came across several challenges and issues, which are stated below.

• When fetching data, same number of tweets can come several times which makes system to produce unexpected results.

• Tweets are less in amount that is tweet corpus is not of large size which makes it more difficult to analyze data.

• To correlate data from twitter and finance department we need data from same timeline which made it difficult to merge file using python code and therefore we used manual way to merge both the data files.

• Processing time can be a big challenge where large dataset can take more than 20 min time to analyze it and therefore can lower the performance of system.

• Twitter does not provide historical data and therefore we are limited to use that dataset.

• Ranking the companies based on the tweets will need some more processing along with recommendation system.

REFERENCES:

[1] http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html

[2]https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[3] https://www.investopedia.com/terms/c/correlation.asp

[4] Machine learning in prediction of stock market indicators based on historical data and data from Twitter sentiment analysis.
http://ieeexplore.ieee.org.ezproxy.gl.iit.edu/stamp/stamp.jsp?tp=&arnumber=6753954&tag=1

[5] Stock Prediction Using Twitter Sentiment Analysis
http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf

[6] Qian, Bo, Rasheed, Khaled, Stock market prediction with multiple classifiers, Applied Intelligence 26 (February (1)) (2007) 2533, http://dx.doi.org/10.1007/s10489-006-0001-7.

[7] E.F. Fama, The behavior of stock-market prices, The Journal of Business 38 (1) (1965) 34105, http://dx.doi.org/10.2307/2350752

[8] J. Leskovec, L. Adamic and B. Huberman. The dynamics of viral marketing. In Proceedings of the 7th ACM Conference on Electronic Commerce. 2006

[9] https://arxiv.org/pdf/1610.09225.pdf

[10] https://github.com/biolab/orange3
[11] https://en.wikipedia.org/wiki/Association_rule_learning
[12]https://github.com/sismetanin/sentiment-analysis-of-tweets-in-russian/blob/master/Sentiment%20Analysis%20of%20Tweets%20in%20Russian%20using%20Multinomial%20Naive%20Bayes.ipynb
[13] https://www.pantechsolutions.net/twitter-sentiment-analysis-using-machine-learning-algorithms-on-python
[14]https://towardsdatascience.com/sentiment-analysis-of-tweets-using-multinomial-naive-bayes-1009ed24276b