Stock market prediction using Twitter Sentiment Analysis

Varun Changala

Dissertation submitted for the Degree of Master of Science in Data Analytics

At Dublin Business School

24th May 2021

Supervisor:

Mr. Shaun Hayden

DECLARATION

I Varun, hereby declare that all the content provided in this study for the Degree of Master of Science in Data Analytics is created by me and contains my sincere efforts as well as work. It is not presented in any Institution for the grant of a Degree.

I hereby declare that all work presented in the study is done ethically and would not harm anyone in any sense. All the ethical concerns are followed, and it cannot and will not lead anyone to the personal details of any individual. As best of my knowledge, this research would not cause discomfort of concern to any individual or institution. This research takes care of all the copyrights, licenses, ethical concerns, and citations. This work follows the academic honesty policy provided by Dublin Business School.

Varun Changala

ACKNOWLEDGEMENT

My supervisor, Mr. Shaun Hayden, was extremely helpful in guiding me through the study process and providing positive feedback. I'm very grateful for his patience and encouragement in the study process. I am grateful to Dublin Business School for assisting me and providing the support I needed to complete this study. I'd also like to express my gratitude to the faculty for supplying us with study materials and for being available whenever we needed them. Finally, I'd like to express my gratitude to my colleagues and friends for their unwavering support.

ABSTRACT

The stock market is the most common place for users to invest. Because of the high profit margins predicted. Forecasting stock market returns has recently gotten a lot of coverage. Stock market forecasting is regarded as a difficult challenge. Data analysis is a method of forecasting thefutureworth. if stock prices will rise or fall in the future.The main goal of this paper is to use the prediction principle to forecast future stock prices. Parse Records calculates the expected value and sends it to the recipient. Using the Automation principle, execute operations such as share purchase and sale automatically. Using machine learning for this. Real-time access is provided by downloading log forms from the Yahoo Finance website and storing them in a dataset.

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF ALGORITHMS

Deep Learning

Machine Learning

Gradient Boosting

Extreme Gradient Boosting (XG Boost)

Long Short Term Memory (LSTM)

Random Forest Classifier

Random Forest Regressor

CHAPTER 1 – INTRODUCTION

In the last few years, the life has become more digital than ever. We rely on the digital information just to survive and progress at the same time. As we go through the day to day motions, it is important to understand the how this change our existence and what can be done with this enormous amount of information. A lot of our questions can be answered just based on trends and connections between multiple sentiments. Our study is focusing on the impact of a social media channel i.e. Twitter on the stock prices of Apple Inc. We are trying to find out how a group of tweets can change the values of stocks and can help the company in gaining profit or can cause loss. We are also trying to find out which Deep learning and Machine learning method is best suited for the prediction of stock prices according to twitter trends.

1.1 Background

Today's word is more depended on the behavior or sentiments of the people which brings behavioral economics in picture. Stocks and its movements are affected by the same in a certain manner, which is difficult to understand as well as predict. The consumer of today is not as rational as before and their decisions can be changed easily, due to this the stock prices can drastically go up or down just because of a tweet. Twitter is one of the leading social media website that can be accessed across the globe, any analysis done to understand hidden twitter trends can help in predicting the future of the stocks and can bring down losses. In this study we are going to be analyzing the stocks of Apple Inc, the first multi trillion dollar company, as it is one of the leading companies is technology and releases new products twice a year. Due to this it can experience a lot of uncertainly during the year and the stock prices can fluctuate.

Twitter started as a communication portal that can be used by friends to stay in touch and let each other know about their lives. But with years, it has converted into a much bigger idea. It experiences more than six thousand tweets in a second and close to five hundred million tweets per day. This provides a surprisingly huge data to be harnessed and studied. Twitter analysis can help in sentimental analysis of many brands, political issues, and products available.

1.2 Business Background

Market Analysis can be of two main types, the first one is technical analysis which can be done on the daily movement of the stocks. It is also affected by any kind of trend indicators, lowest as well as highest values of throughout the day. The second type of analysis is Fundamental Analysis which can be done on the reports and already existing financial data. It is also believed that predicting stock market prices correctly can either be very difficult or just cannot be done.

The motive of this study was to understand trends of the Apple Inc. based on twitter tweets and understand the movement of the market. This could later be used for the prediction of Stock market ups and downs and reduce the overall loss. In opposite sense, it will also be helpful to increase the profit percentages and predict the changes as soon as any new product enters the market. This analysis can also be used to reduce the panic that is induced by a crash of market or due to the bubble burst, as most of the times twitter is the first places where these trends can be seen.

1.3 Research Problem

Our research is trying to find a correlation between the twitter tweets and stock market movement using the sentimental analysis. Research question along with other details are provided in this section below:

***Research* Question:** Which machine or deep learning model can best correlate the stock market data with Twitter?

Aim: To prove the correlation between twitter and sentimental analysis of stock prices.

Objective: To use deep learning and Machine learning methods such as Gradient Descent, XG Boost, Random forest Repressor, Random forest classifier and Long short term memory to analyze the stock relations with Twitter tweets.

Hypothesis: The comparative study will provide the best possible outcomes for the prediction of stock market movement with and will help us in understanding which model works the best.

1.4 Scope

- To pull tweets from the twitter platform using tweepy library and twitter developer API key and save it as a dataset.
- Download the company's dataset from yahoo finance with the same timeline matching with tweets dataset.
- To join the data together using the date timeline as the index.

- To find the best suitable model with accuracy using the comparison between all the models.
- To find the sentiments of the tweets using nltk library and the polarity of tweets.
- To distinguish the sentiments into positive, negative and neutral.
- To find the correlation of the stocks data.
- To find the correlation of the sentiments.
- And finally to show that there is a correlation between the stocks and tweets sentiments.
- To apply multiple models and find the best one out of them.

1.5 Limitations

All studies ever conducted have experienced some limitations, for this particular research we have faced a few limitations. The most prominent issues that we faced was that of the API in which we were only able to pull 5, 000, 00 tweet records per month. Correlation with historical data was failing as the timeline of twitter API's limited data and the Stock's data from yahoo finance were different. Due to this API was discarded and the dataset was picked up from kaggle. The absence of live data was resulted due to the failure of API and historical data from year 2016 to 2019 was taken.

The computation power of the system could also be considered a limitation as having higher power would have helped in achieving a higher accuracy.

1.6 Roadmap

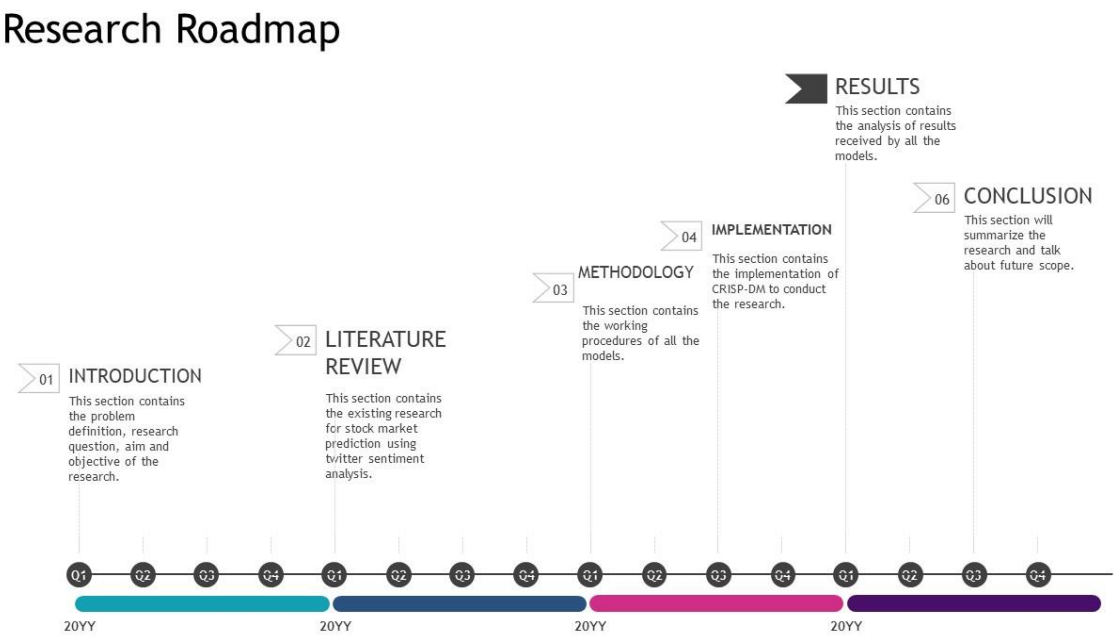

Figure 1 Research Roadmap

We have started this research with Introduction; it is followed by Literature review in the next section. Literature review is followed by Methodology in section 3. Section 3 is followed by Implementation in section 4, by result discussion in section 5 and Conclusion and Future work, references in section 6 and appendix in section 7.

CHAPTER 2- LITERATURE REVIEW

2.1 Background

Research paper (Hao Wang et al., 2012) discusses about a real-time analysis system for public sentiment in the 2012 U.S. election of presidential candidates which is expressed on the Twitter platform. On the IBM's InfoSphere Streams a real time data processing infrastructure was built for accuracy and speed. Gnip Power Track is used to collect all the real time tweets from the Twitter. For creating a baseline sentiment model Amazon Mechanical Turk (AMT) implemented to get varied population of annotators. The training data used in this research contains 17000 tweets. The statistical classifier used for sentiment analysis is Naïve Bayes model on unigram features. Accuracy obtained based on the relevant dataset was around 59% for the Naïve Bayes classifier. For the visualization purpose an Ajax-based HTML dashboard was designed to visualize volume and sentiment by candidate as well as trending words and system statistics.TF-IDF metric implemented for identifying most important words when event occurs. Results and findings declare that the tweet volume is largely driven by campaign events and confirms that the system built can provide real-time analysis during key moments in election cycle.

(MasoudMakrehchi et al., 2013) paper proposes a new approach for labeling social media text using stock market events. Further extraction of useful stock movements, collection of tweets from Twitter and social media sources was done. Labeling of each tweet as positive and negative is done before training the model later predicting the future label of the tweets. Supervised sentiment analysis is discussed having two key components as automatically generating training data from unlabeled user tweets and building a sentiment classifier based on the training data.For the classification of the sentiments two types of sentiment classifiers is in focus one classifier uses heuristic features from the mood where every tweet represented

by feature vector and for the second classifier feature engineering is performed. Introduction of the over sampling technique to deal with class imbalance issue is done. Later, the prediction of future tweets is discussed. Results stated that the trading strategy was able to beat S and P 500 approximately by 20% returns in the time interval of four months.

2.2 Related Works

Research paper (Tahir M. Nisar etal., 2018) uses twitter data to find out a correlation between public sentiments concerning local elections and stock market movements in FTSE on a UK based political event used for the data mining process in this research. The study involved establishing of hypothesis to match out the correlation proposed in the reference papers and the actual experiment carried out. For data collection from Twitter api, software Tweetcatcher and Trendogate was used and further data cleaning is discussed. Sentiment Analysis was performed by extracting the sentiments from every tweet collected with the help of sentiment classifier Umigon. FTSE100 data was also collected and cleaned using a concave function. Taking volume and percentage change as a basis of analysis. Overall the study involved carrying out collection of correlation and regression analysis to compare daily mood with daily change in prices at the market level. Results shows that there is a correlation between general mood of public and their investing behavior also suggests that there is evidence of causation between public sentiment and movement of the stock market including z-scores comparison using Pearson's correlation test and ANOVA testing.

Research paper (RohanPimprikar et al., 2017) uses various Machine learning algorithms while comparing their performance along with Twitter sentiment analysis for the prediction of the stock market prices. In this paper classification of sentiments of twitter data is done. In order to gauge out the market sentimentsvarious Machine learning algorithms and Neural

Networks are used which are, Linear Regression, Support Vector Machine, Multi-Layer Perceptron Neural Network and the Long Short Term Memory. For the sentiment analysis classification the Naïve Bayes classifier is used. The activation function and solver function for the MLP Regressor is taken as “Relu” and “lbfgs” while for the LSTM activation function is “linear” and solver function is “RmsProp” with number of layers in both the techniques is taken as 100. The accuracy achieved for the technique Linear Regression was 82% while that of SVM was 60% approximately and the RMSE values of the Neural Network models were below 0.3. Research concluded that sentiment analysis done on the comments is only influential when more than 80% of the tweets shows positive or negative sentiment or else it would be overshadowed by the neutral sentiment.

(VenkataSasankPagolu et al., 2016) this paper discusses the existence of relationship between the people’s sentiment expressed on the Twitter platform and the stock prices of a company is discussed, leading to develop a sentiment analyzer that analyses the types of sentiments of the tweets which can be positive, negative and neutral. A claim is made that the positive tweets or emotions of the user on twitter regarding the company reflects in its stock prices.In this paper two different textual representations are used for analyzing public sentiments from the tweets, they are: Word2vec and N-gram. Furthermore to analyze the correlation between the sentiments of the tweets and the stock market movements Sentiment Analysis and Supervised Machine Learning technique is applied. For the classification of the sentiments Random Forest classifier is used giving out an accuracy of 70.2% with the features as Word2vec representations of human annotated tweets while the accuracy with N-gram representations having same algorithm was recorded as 70.5%. Stock price and sentiment correlation results observed the same classifier when trained using Logistic Regression gave out the accuracy of 69.01% while the model trained with LibSVM gave a result of 71.82%.

Research paper (John Kordonis et al., 2016) investigates whether the public sentiments collected from the Twitter correlates or at least is predictive of stock values of 16 most valuable and popular IT companies suggested by Yahoo Finance. The Machine learning algorithms used are Naive Bayes Bernoulli classification and Support Vector Machine (SVM), to give positive or negative sentiment on the tweet corpus and same algorithms used to detect the correlation between tweets and stock market price behavior. Feature extraction and Feature filtering was implemented. Classification results showed the accuracy value of Naïve Bayes 80.6% while that of SVM was 79.3%. Stock market prediction using SVM achieved average accuracy of 87% and the prediction errors were under 10%. Concluded, that the effect of changes in public sentiment is visible in the stock market prices.

(TusharRao et al., 2012) paper examines the correlation between the tweet board literature and the financial market instruments for more than 4 million tweets for DJIA, NASDAQ-100 and other 13 big cap technological stocks. API is used for the tweets collection and processing. Sentiment analysis involved with the classification of the tweets was carried out using the online classifier that is Naïve Bayes Classifier also JASON API from the Twittersentiment was used. The Naïve Bayes classifier achieved an accuracy of about 82.7%. Further the tweet feature extraction was implemented along with financial data collection done with the help of Yahoo Finance API. Daily volatility estimation based on intra-day highs and lows was done using Garman and Klass volatility measures. Granger Causality Analysis (GCA) applied to investigate a statistical pattern of lagged correlation. EMMS model having ARIMA and ES methods for forecasting was introduced with OLS regression technique for accuracy prediction which achieved around 91% directional accuracy. Concluded, that negative and positive sentiment of public has strong relationship with price movements of individual stocks.

Research paper (Anshul Mittal et al., 2012) uses Twitter to predict public's mood and combines it with previous day DJIA values for stock market movement prediction. Relation between the public sentiment and stock market sentiment is observed using sentiment analysis and machine learning principles. Yahoo Finance was used to obtain Twitter data. Sentiment analysis for multiclass classification of the sentiments was done using the methods: Tweet Filtering, Word List Generation, Daily Score Computation and Score Mapping. For result testing a newly proposed cross validation method for financial data was introduced thereby obtaining 75.56% accuracy by applying SOFNN (Self Organizing Fuzzy Neural Networks) on Twitter feeds and DJIA values. A naïve portfolio management strategy was implemented based on the predicted values. Results concluded that firstly, the correlation was better between calm and happy mood sentiments with DJIA values, secondly, 75.56% result accuracy was obtained using K-fold sequential cross validation gave strong correlations over the entire range of data.

(RupawariJadhav et al., 2017) paper discusses the stock market prediction based on the social media platform mainly Twitter with the help of sentiment score. To classify the sentiment score classification techniques used are Naïve Bayes, SVM, Maximum Entropy, and Random Forest with SVM scoring the highest accuracy of 89.8%.Further prediction techniques used for the technical analysis of the stock market. Method of Time Series Analysis for the short term prediction of the stock used is Random Walk, Moving Average, Regression method, ARIMA model. Furthermore the evaluation of prediction model carried out using the factors like Mean Absolute Error, MSE, RMSE etc. helped calculate error of different prediction models.

Research paper (ShriBharathi et al., 2017) focuses on the correlation between the stock market values and the RSS feeds along with sentiment of tweets for particular period of time. The aim of this paper is to achieve high precision in stock market prediction with the

combination of Sensex points and the Really Simple Syndication (RSS) feeds and tweets. ARBK is the company from the Amman Stock Exchange for which the RSS feeds, tweets and stock market prices are collected. Two hypothesis is proposed in this paper, one is Null Hypothesis H0 : Stock level indicators predict the trend of stock rates at an allowable rate of minimum 80% above and other is Ha: Stock level indicators along with the sentiment analysis of RSS news feeds and tweets as stock enhances the accuracy of prediction. This research is divided in two parallel tasks: one task is implementing Sentiment mining on the RSS contents and Twitter feeds further performing data cleaning and further passing each sentence through Part-of-Speech (POS) tagger of NLP module to extract sentiments from each word depicting positive, negative and neutral values. After the NLP process an algorithm Sentence Score Sentiment (SSS) is used to predict the overall result of the sentence further an equation is used to calculate Sequence of words. The other parallel task is to use the MMS engine to compute the stock level indicators. First level Indicator used is called as Moving Average, second is Moving Average Convergence/Divergence oscillator (MACD) and the third is Stochastic Relative Strength Index (RSI) resembling an oscillator for calculating value between 0 and 1. Prediction of the stock market is done by combining an analyzing the results of both Sentiment analysis of Twitter, RSS news feeds along with the Sensex movements. Results concluded that the social media sentiments with stock level indicators increase the quality of prediction. The Hypothesis test results show that Ha has noticeable improvement in precision compared to H0.

(AjlaKirlić et al., 2018) uses Twitter API for tweets extraction from the Microsoft Company for the duration of October 2th, 2017 to October 24th, 2017. This paper discusses and investigates the relationship between the Twitter feed content and the stock market movement. Missing stock values were approximated using a calculation function. Vader is used for processing of Twitter data which decides the polarity of the tweets whether is

positive, negative or neutral. Tweets having scores smaller than 0 is termed as negative, for tweets having scored greater than 0 termed as positive and witha score equal to 0 is consideredto haveneutral polarity in this research. Further the data from the stock exchange market was collected from Nasdaq website.Aggregation of Vader scores on tweets each day and related stock price values was performed and discovered that with correlation it is possible to find connection between two variables. Correlation coefficient value termed out to be 0.7815 depicting that there is strong positive relation between stock prices and tweets polarity for same interval of time.

Research paper (MehakUsmani et al., 2016) uses different Machine Learning techniques to predict market performance of Karachi Stock Exchange (KSE) on closing day. The prediction model uses different input attributes to predict only two market polarities mainly positive and negative.A comparison of ML techniques is made on the basis of their performance. The technique proposed in this paper uses different factors affecting the market as input attributes for the model. These attributes include Oil rates, Gold and Silver rates, Interest rate, Foreign Exchange rate, ARIMA, SMA, KIBOR, News and social media feed as input for the prediction model. Further Text Mining techniques were used to process the Twitter and News data.Machine Learning techniques like Single Layer Perceptron (SLP), Multi-Layer Perceptron (MLP), Radial Basis Function (RBF) and Support Vector Machine (SVM) were implemented and compared with other techniques. Single Layer Perceptron was trained for 50 epochs and learning rate was 0.3 and tested on the same dataset to give out 83% accuracy. Multi-layer Perceptron trained for 500 epochs and learning rate was 0.05 with 1 hidden layer and 4 neurons when tested on the same dataset it gave 77% correct results whereas RBF when trained with lr = 0.03 and 500 epochs termed out to give 63% accuracy on the test set results. For the Support Vector Machine (SVM) the configuration taken as same kernel function as used in the RBF along with the value of sigma configured as 0.3 when trained on

the training set gave 100% results and test set turned out to be 60% approximately. Covariance and Correlation of the attributes and the Market performance was measured. Results concluded that MLP performed the best when compared to other Machine Learning techniques and that the Machine Learning techniques are very much capable of predicting the stock market performance.

(ApoorvAgarwal et al., 2011)research paper uses Twitter data for sentiment analysis with having two new resources for the pre-processing the data namely Emoticon Dictionary and Acronym Dictionary. POS-specific prior polarity features are introduced in this paper and to avoid the need of tedious feature engineering, uses of tree kernel was investigated. The prior polarity of about 88.9% of English Language words was found. Combination of many categories of features in one overall tree like representation of Tweets was done. Further, to calculate the similarity between two trees Partial Tree kernel was used which is an instance of Convolutional Kernels. A set of features was proposed which was divided into 3 categories which are: bag of words, presence of exclamation marks and capitalized text. Further these were classified into two sub-categories: Polar and Non-polar features. Results were classified for two different classification tasks: Positive versus Negative versus Neutral and Positive versus Negative. For each classification task three models were presented: Unigram model, Tree kernel model, 100 Senti-features model, Kernel plus Senti-features and Unigram plus Senti-features. Support Vector Machine (SVM) and report averaged five fold cross validation test results was used. Combination of Unigrams with Senti-features performs best for the positive versus negative task, with absolute gain of 4.04% was recorded over a hard unigram baseline. Combination of tree kernel with senti-features performs best for 3 way classification task gaining over 4.25% over unigram baseline. Results declare that both tree kernel and feature based models outperform the unigram baseline. Concluded, that sentiment analysis for Twitter data is almost same as sentiment analysis for other genres.

CHAPTER 3- METHODOLOGY

In this study we are using Cross- Industry Standard Process for Data Mining (CRISP-DM) is a methodology that is used to plan a project in an organized manner. It provides a structure by dividing processes in clearly defined stages, which helps in achieving more efficiency and faster results. These phases can be considered as the life cycle of a product with respect to data mining. The use of CRISP-DM can be done in all the domains as it is very versatile and can easily be modified based on the requirement of the projects. Here we are not altering a lot of CRISP DM processes as the project is not requiring it. We are just altering Data pre-processing section as the data obtained is already very clean and is in enough quantity for training, validation and Testing.

All phases of CRISP-DM:-

Business Understanding

Data Understanding

Data Preparation

Modeling

Evaluation

Deployment

Figure 2 CRISP DM Methodology

We will first be starting with Business Understanding and will follow it up with other phases.

3.1 Business Understanding

To start with any project first you need to understand the desired business outcome and how it will be achieved. Business Understanding does exactly this by trying to figure out the purpose of the project according to the needs and preparing a plan to reach that objective. As our study is going to perform a comparative study on some of the deep learning and machine learning models, our business understanding will be of finding the best model out of all applied in this research. One other aspect of this study is to find the correct correlation between the twitter tweets and their impact on the Stock prices of Apple Inc. We would like to understand that does any negative tweet by any individual produces and overall loss to the apple Inc or it remains as it is. We would also like to understand the positive effects of positive words on the stock of

the company, so in later stage, both losses and profits can be analysed and panic can be reduced due to sudden price changes.

The result of this analysis can be noted and used to make an application and further help in other studies.

3.2 Data Understanding

This phase starts with collecting the required data from one or different sources. Once collected, identification of important and prominent characteristics of the data is done while observing its general properties. The main focus is to verify if the available data is fairly associated with the outcome, if not new data needs to be collected. Data exploration is done next, the assumed hypothesis is verified using queries and visualization and the results are taken into consideration. At this point, data needs to be cleared in order to get a solid foundation for the next step.

For our study, the data that we have picked up is present in the format of a .csv file. It is containing the around 1k values and total 9 columns. This .csv contains the details of the stocks, their related date, popularity, twitter volume and as such.

3.3 Data Preparation

Among huge data, good quality relevant data is selected and is cleaned. Cleaning can include removing missing or corrupt values from tables and rows. New derived attributes can also be added for a proper model application. New records can also be made by integrating values from different tables that point out to the same information or by aggregation. Lastly, data can be formatted to make it appropriate

for the model. Formatting can include rearranging values, removing unnecessary characters, trimming strings, etc.

For this study, as the data was fairly cleaned, there was no requirement of pre processing the data. Due to the presence of .csv format, the data was ready to be fed to the models for training and testing. Data pre-processing was required in case the data had any RGB and Greyscale images, which could have been converted to arrays or .csv, but as data was already in that state, it eliminated the requirement of pre-processing.

3.4 Modeling

In this phase a single or multiple models are selected according to the data and assumptions are noted. The test data is divided into two sections; Testing and Training data and the model uses training data to understand the pattern which it later applies on the test dataset. This helps in validating the usefulness of a model with respect to the dataset as instead of predicting new values, the model will predict past values. Based on the result, the model is used to make future predictions.

We have applied following models for the study, as they were the most suitable models for the dataset:-

Gradient Boosting

Extreme Gradient Boosting (XG Boost)

Long Short Term Memory (LSTM)

Random Forest Classifier

Random Forest Regressor

Gradient Boosting

The general use of gradient boosting is a part of Machine learning boosting types. The focus of this boosting technique is the intuition that overall prediction error of the model can be reduced when we combine it with some previous models and it in turn creates the best possible succeeding model.

The result of every single data is dependent on the changes and how as well as how much that change impacts the prediction. For example, if the change is very small and it can cause a large drop in the error, then next outcome is considered of high value. This is having another version where if the small change in prediction causes no changes in the error, then next outcome of this particular case is zero.

Boosting is a technique for turning weak learners into good ones. Each new tree in boosting is based on a changed version of the original data set. The gradient boosting algorithm (gbm) is best demonstrated by first learning about the AdaBoost Algorithm. The AdaBoost Algorithm starts by training a decision tree with equal weights for each observation.

Following the evaluation of the first tree, we increase the weights of the difficult-to-classify observations and decrease the weights of the easy-to-classify observations. As a result, the second tree is based on the weighted results.The goal here is to build on the first tree's predictions. As a result, our new model is Tree 1 + Tree 2. The classification error from this new 2-tree ensemble model is then computed, and a third tree is grown to estimate the revised residuals.

This method is repeated for a set number of iterations. Following trees aid in the classification of findings that were not well classified by previous trees. The weighted total of the predictions made by the previous tree models makes up the final ensemble model's predictions.

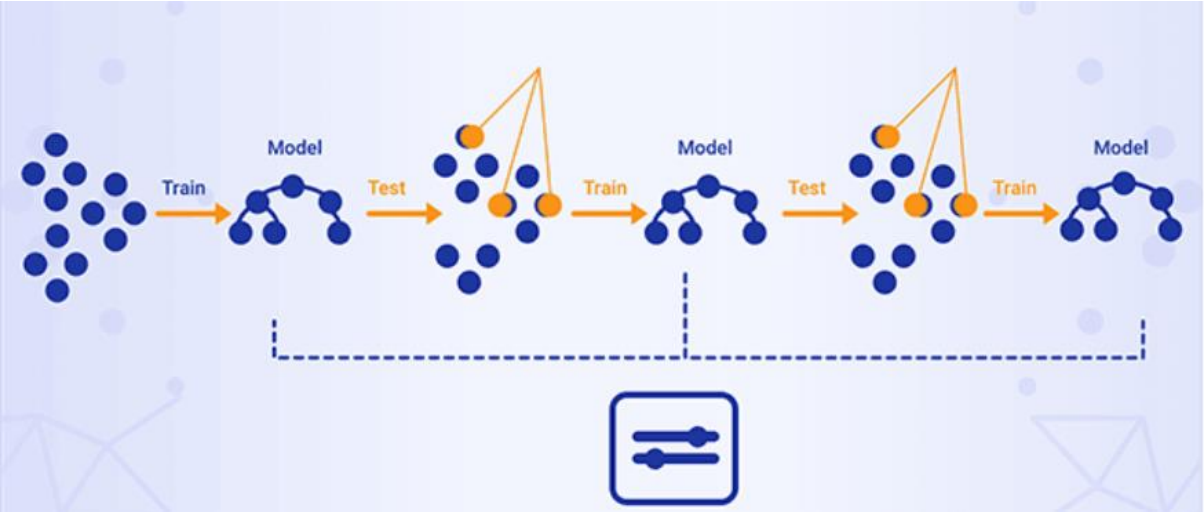

Figure 3 Working of a model using tree algorithm.

Extreme Gradient Boosting (XG Boost)

Extreme gradient boosting is a decision tree based algorithm that is using the above explained gradient boosting algorithm. Decision tree based algorithms are one of the best options available for the small and medium sized tabular data.

The image below explains how the tree based algorithms have evolved over the years.

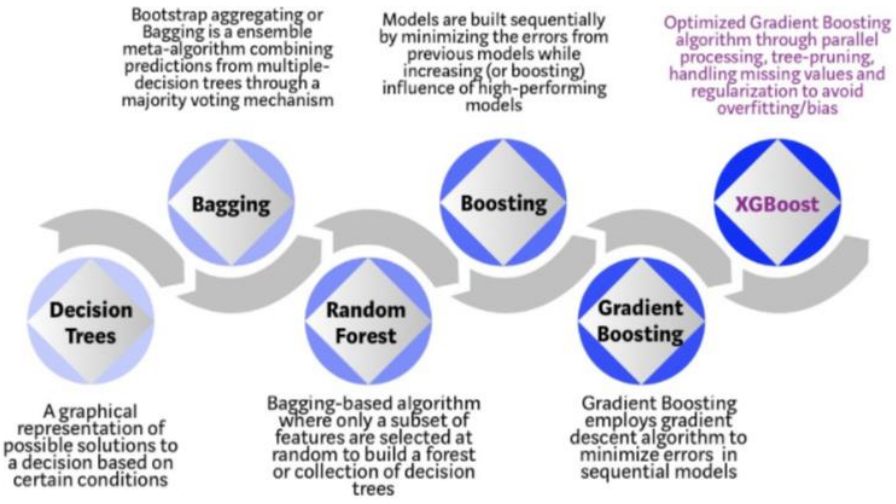

Figure 4 Different types of models used.

XGBoost stands for Extreme Gradient Boosting. Because of its scalability, it has recently gained popularity and is currently leading applied machine learning and Kaggle competitions for structured data. To prevent over-fitting, the regularization term helps to smooth the final learned weights. The regularized goal would favour models that use simple, predictive functions. Traditional optimization methods in Euclidean space cannot be used to optimize the tree ensemble model. Rather, the model is taught in an additive fashion. To avoid overfitting, two additional methods are used in addition to the regularized target.

Friedman invented the first method, shrinkage. After each stage of tree boosting, shrinkage scales newly added weights by a factor. Shrinkage decreases the effect of each tree and leaves room for future trees to develop the model, like a learning rate in stochastic optimization. Column (feature) sub sampling is the second technique. Random Forest makes use of this technique. Over-fitting is prevented much more by column sub sampling than by conventional row sub sampling. The use of column sub-samples often speeds up the parallel algorithm's computations.

The most difficult aspect of tree learning is determining the best break. This algorithm iterates through all the possible function splits. Enumerating all possible splits for continuous features is computationally intensive. The exact greedy algorithm is extremely efficient since it greedily enumerates all possible splitting points. However, when the data does not fit completely in memory, it is difficult to do so effectively. The Approximate Algorithm suggests candidate splitting points based on feature distribution percentiles. The algorithm then divides the continuous features into buckets based on the candidate points, aggregates the statistics, and seeks the best solution among the proposals based on the statistics. Because of its parallel and distributed computation, XGBoost is a faster algorithm than other algorithms.

XGBoost was created with careful consideration of both system optimization and machine learning concepts. The aim of this library is to push machines to their limits in terms of computation in order to create a scalable, compact, and accurate library.

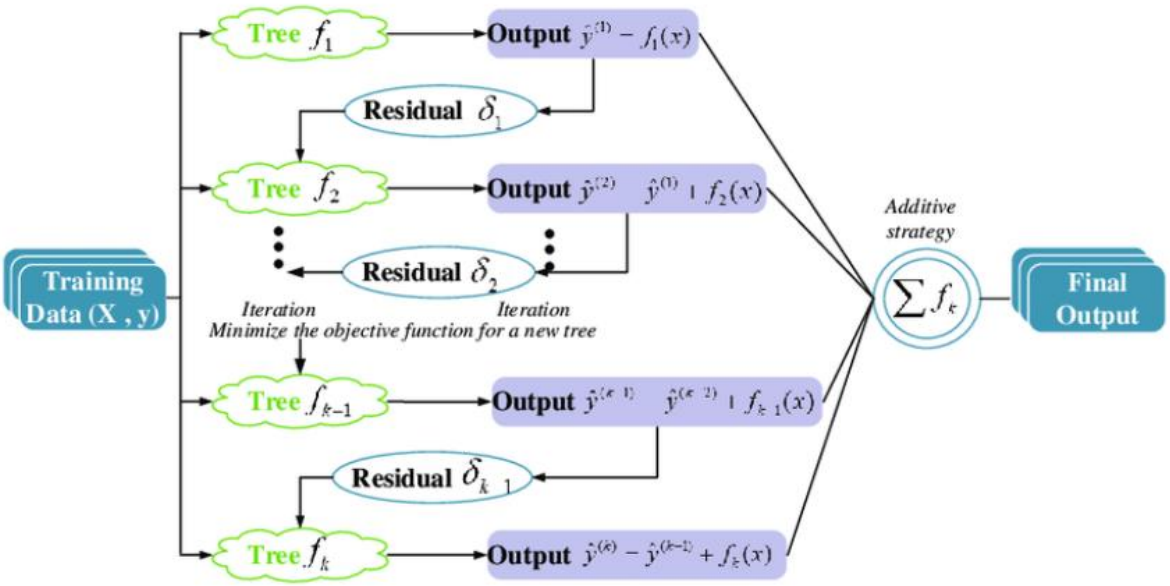

Figure 5 XG Boost algorithm working.

Log Short Term Memory (LSTM)

Long Short Term Memory (LSTM) networks are part of deep learning networks and a trype of recurrent neural networks which are able to learn the order of dependence in problems of sequence predictions. Which is why this is a suitable model to be used in the prediction of stock prices based on twitter tweets.

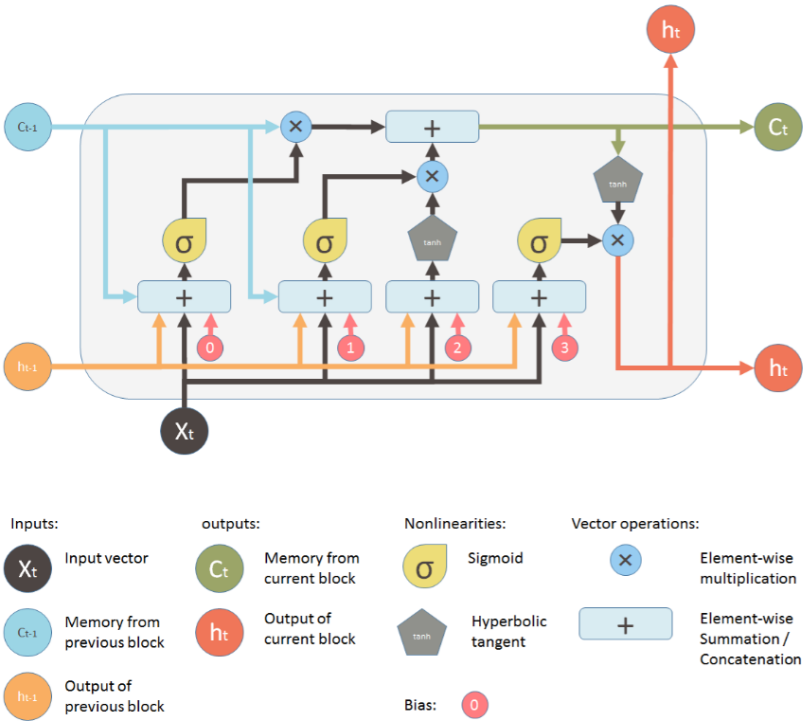

Figure 6 LSTM model working.

Problems with sequence prediction have been around for a long time. They are regarded as one of the most difficult problems in the data science field to solve. These problems vary from forecasting revenue to identifying trends in stock market data, from comprehending movie plots to recognizing your voice, from language translations to predicting your next word on your iPhone's keyboard.

With the recent advances in data science, it has been discovered that Long Short Term Memory networks, also known as LSTMs, are the most powerful solution for almost all of these sequence prediction problems. In several ways, LSTMs outperform traditional feed-forward neural networks and RNNs. This is due to their ability to recall patterns selectively over long periods of time. The aim of this article is to illustrate LSTM and show you how to apply it to real-world problems.

All test cases are considered independent in traditional feed-forward neural networks. That is, when fitting the model for a specific day, the stock prices from the previous days are ignored. RNNs will help us with sequence handling to a degree, but not fully. We prioritize our appointments when we plan our day's schedule, right? We know may meeting might be cancelled to accommodate a potential meeting if we need to make some room for anything significant. An RNN, it turns out, does not. In order to add new details, it fully transforms the existing data by adding a function. As a result, the whole information is altered, i.e., there is no distinction between ‘essential' and ‘not so important' information.

LSTMs, on the other hand, use multiplications and additions to make minor changes to the data. Knowledge flows through a system known as cell states in LSTMs. LSTMs may selectively recall or forget things in this way. There are three different dependencies on the information at a specific cell state.

Random forest classifier

A supervised classification algorithm, the Random Forest algorithm. We can tell from the name that the aim is to create a forest in some way that is random. The number of trees in the forest has a direct relationship with the accuracy of the results: the more trees, the more accurate the result. However, it is important to remember that developing the forest is not the same as building the decision using the knowledge gain or gain index method.

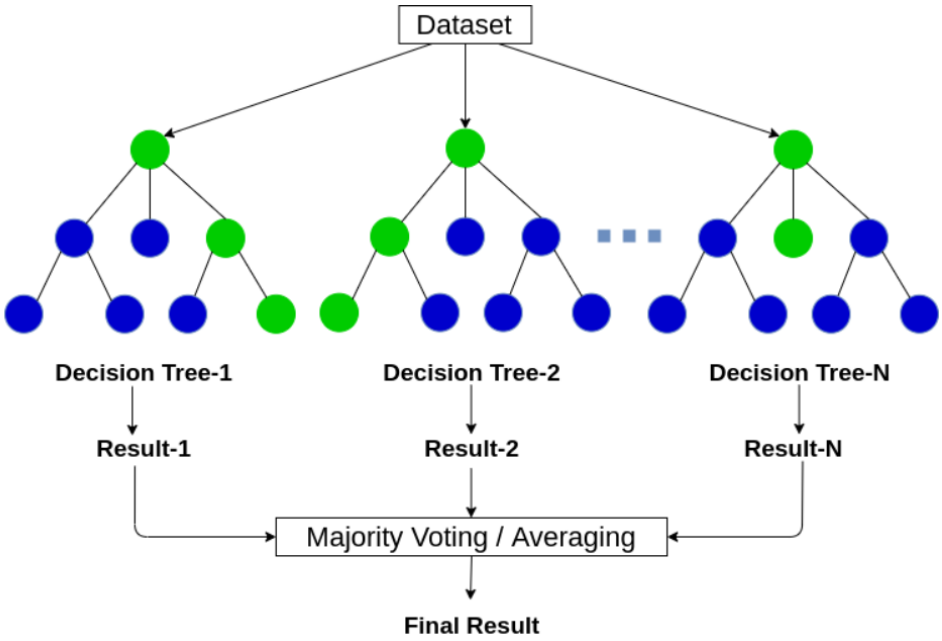

Figure 7 Random Forest working.

The decision tree is a method for aiding in decision-making. It depicts the potential outcomes using a tree-like graph. When you give the decision tree a training dataset with goals and features, it will generate a set of rules. Predictions can be made using these laws. If you want to predict whether your daughter will enjoy an animated film, gather the previous animated films she has enjoyed and use some features as feedback. The rules can then be generated using the decision tree algorithm. You may then enter the details of this film to see if your daughter can enjoy it.

The difference between the Random Forest algorithm and the decision tree algorithm is that the method of finding the root node and separating the feature nodes in Random Forest is done at random. Overfitting is a serious issue that can sabotage results, but with the Random Forest algorithm, the classifier will not overfit the model if there are enough trees in the forest. The Random Forest classifier can handle missing values, and the Random Forest classifier can be modelled for categorical values, which is the third advantage.

Random Forest Regressor

Random forest is an algorithm that can easily be used for both Regression and Classification, as we have used it for the classification. In this section we are using it for regression as well. Even in regression same kind of processes are followed as they are done in classification so it can be considered reliable for both the segregations.

As this study has both kinds of data, we will also be applying random forest regression for the predictions. Random forest regression is a group of decision trees that work together and provide the outcome. These trees are assembled in a particular way to form a Random Forest. Every single tree is made from multiple samples of rows and on each node , a feature is selected for splitting the features. As there are multiple trees, a single tree makes its own predictions and produces the results, these values are then averaged and a single result is provided. Due to having the average of multiple random forest trees, this provides the best results for regression.

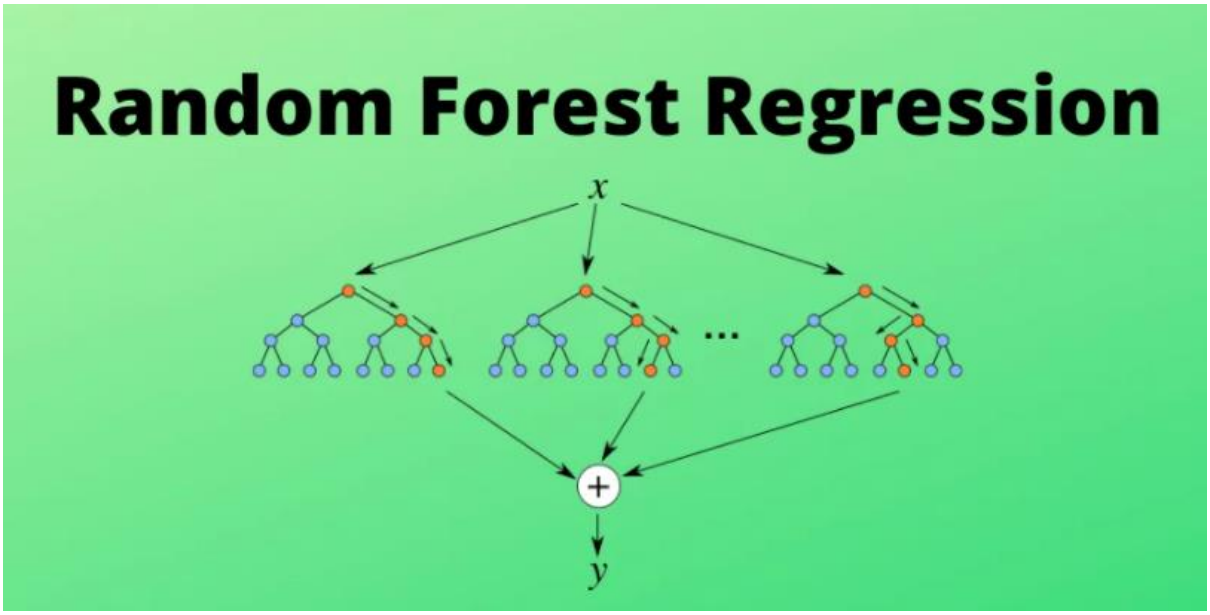

Figure 8 Random Forest Regression.

3.5 Evaluation

Before deploying or displaying the changes it is made sure that model is doing exactly what it is supposed to do and is not deviating from the business objective. This is done by observing the results and verifying if all impactful factors are taken into consideration and the correct objective is achieved. Quality Assurance can also be included at this stage for an error free end result.

3.6 Deployment

Once all the above mentioned steps are completed the team decides if it is okay to make the product live for the customer use. Different strategies are made for the deployment based on results from all the steps. Once deployed every project shifts into maintenance phase where issues are highlighted and dealt with daily in order to keep the product up and running and also to provide updates when necessary. Based

on the behaviour of the product a final report can be made to highlight improvements and issues that can be sorted at later stages.

Here, as this is a study, there is less possibility to make a server and deploy the changes. But this segment can be taken up as Data presentation. We are presenting all our results in the format of this report and an additional ppt. This report will provide all the required information that will be needed to recreate the study or understand it at any stage.

CHAPTER 4 - IMPLIMENTATION

4.1 System Specifications

Stock price forecasting is a difficult and time-consuming task since there are so many factors in play, such as political events, economic conditions, and other environmental elements that can influence the stock price. Due to these considerations, determining the dependability of a single factor on future prices and patterns is challenging. So, to resolve this task various Machine Learning algorithms are applied in this research to predict stock prices.

This research was conducted on Jupyter (Anaconda) platform which is mainly used for executing and writing Python code. The system configuration comprises of GPU as NVIDIA with 32GB RAM. Furthermore the Python programming language is used for the implementation of this research because python language being easy to understand and consists of various libraries that ease the model implementation process.

4.2 Dataset Description

Dataset used for this research consists of Twitter data for Apple stock collected from Kaggle (https://www.kaggle.com/nadun94/twitter-sentiments-aapl-stock) for the application of Sentiment Analysis process for the time period of January 01, 2016 to August 31, 2019 and for the same time frame the Apple Stock data was also collected for stock price prediction and historical data analysis from Yahoo Finance using (https://finance.yahoo.com/quote/AAPL/history?p=AAPL)website).

For the analysis purposes both the datasets were merged to form a final csv dataset which was considered and used as Base data.

The dataset contains 922 rows and 9 columns. Information gathered from the Twitter was treated to be used for the specific dates on which the stock market was open making up to 256 days a year for trading. The dataset comprises of 9 columns which are: Date, Open, High, Low, Close, Adj Close, Volume, ts_polarity, twitter_volume. Data types included: float64, int and object.

```
In [33]: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 922 entries, 0 to 921
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            922 non-null    object
 1   Open            922 non-null    float64
 2   High            922 non-null    float64
 3   Low             922 non-null    float64
 4   Close           922 non-null    float64
 5   Adj Close       922 non-null    float64
 6   Volume          922 non-null    int64
 7   ts_polarity     922 non-null    float64
 8   twitter_volume  922 non-null    float64
dtypes: float64(7), int64(1), object(1)
memory usage: 72.0+ KB
```

Figure 9 Data Frame information.

Workflow

A brief outline is presented in this section, of the implementation steps of various Machine Learning Algorithms used for the analysis of correlation between Twitter sentiments and Apple stock price movement. The performance of the models

deployed depends greatly on the system configuration as the training time and execution are subject to change with different system configurations.

This research was conducted in two phases:

Classification

Regression

Classification

Machine Learning algorithms deployed for the classification task are:

Gradient Boosting Classifier

Implementation:

Prediction of apple stock trend with the help of Gradient Booster classifier, this classifier is used to identify the trend in Apple Stock using APPL.csv file. Various python libraries have been imported in the start of the modeling process as the libraries grants the access already implemented logic data or functions in the form of variables like numpy, pandas, plotly etc. The data APPL.csv is then loaded and preprocessedin the jupyter platform using import os function further providing the path in a pandas Dataframe known as df.

Loading and Preprocessing Data

Load the `APPL.csv` in a pandas DataFrame called `df`

In [25]:
```
import os
path = 'Z:\\DBS\\Thesis\\Sentiment-Analysis-of-Twitter-Data-for-predicting-Apple-stock-price-master'
os.chdir(path)
print(os.getcwd())
```

```
Z:\DBS\Thesis\Sentiment-Analysis-of-Twitter-Data-for-predicting-Apple-stock-price-master
```

In [26]:
```
df = pd.read_csv('AAPL.csv')
df.dropna(inplace=True)
df.tail()
```

Out[26]:

	Date	Open	High	Low	Close	Adj Close	Volume	ts_polarity	twitter_volume
917	2019-08-26	51.47	51.80	51.26	51.62	51.12	104174400	0.072340	888.0
918	2019-08-27	51.97	52.14	50.88	51.04	50.54	103493200	0.117541	962.0
919	2019-08-28	51.03	51.43	50.83	51.38	50.88	63755200	0.061477	895.0
920	2019-08-29	52.13	52.33	51.67	52.25	51.74	83962000	0.056460	1083.0
921	2019-08-30	52.54	52.61	51.80	52.19	51.67	84573600	0.106096	1005.0

Figure 10 Dataset Description.

Further the Index is set as date in order to correlate the stock prices and Twitter data with date. The ts_polarity is sorted according to the sentiments classified as Positive, Negative and Neutral.Binary Encoding is implemented on the sentiment column using the pd.get_dummies function.

In [14]:
```
# Binary encoding Sentiment column
appl_trend = appl_df[["Adj Close", "Volume", 'twitter_volume', "Sentiment", "Trend"]]
appl_trend = pd.get_dummies(appl_trend, columns=["Sentiment"])
appl_trend.head()
```

Out[14]:

Date	Adj Close	Volume	twitter_volume	Trend	Sentiment_Negative	Sentiment_Neutral	Sentiment_Positive
2016-01-05	23.83	223164000	1430.0	0	0	0	1
2016-01-06	23.36	273829600	1949.0	0	0	0	1
2016-01-07	22.38	324377600	2289.0	0	0	0	1
2016-01-08	22.50	283192000	2235.0	1	0	0	1
2016-01-11	22.86	198957600	1222.0	1	0	1	0

Figure 11 Sentiment column.

After defining the features set and target vector the data is then split into train and test sets with the ratio of 70:30 and the features are scaled using StandardScaler function

then Gradient Booster model is created with n_estimators set to 20 and trained on various learning rates for different accuracy comparisons on different rates. The model is created and fitted using the classifier.fit function and the model score is calculated and predicted using model.predict function and further the model is evaluated using precision, recall and f1score evaluation metrics.

Create a Gradient Booster Model

In [19]:
```
# Choosing learning rate
learning_rates = [0.05, 0.1, 0.25, 0.5, 0.75, 1]
for learning_rate in learning_rates:
    model = GradientBoostingClassifier(n_estimators=20,
                                       learning_rate=learning_rate,
                                       max_features=2,
                                       max_depth=3,
                                       random_state=0)
    model.fit(X_train_scaled,y_train.ravel())
    print("Learning rate: ", learning_rate)

    # Scoring the model
    print("Accuracy score (training): {0:.3f}".format(
        model.score(
            X_train_scaled,
            y_train.ravel())))
    print("Accuracy score (validation): {0:.3f}".format(
        model.score(
            X_test_scaled,
            y_test.ravel())))
    print()
```

Figure 12 Gradient Booster Model.

RandomForest Classifier

Implementation:

Prediction of apple stock trend with the help of Random Forest Classifier, this classifier is used to identify the trend in Apple Stock using APPL.csv file. Firstly, the important libraries are imported like numpy, pandas, standardscaler etc. then the dataset APPL.csv is loaded in a pandas dataframe called as df using the import os function. Then the data is read and the tail of the dataset is visualized using pd.read and df.tail functions. The columns are renamed in the dataframe and the head is displayed. Therefore the same process as mentioned in the Gradient Booster section

is repeated as setting of the index as Date, Sorting of the ts_polarity into Positive, Negative and Neutral sentiment. The count of the sentiments is then calculated using value_counts() function.

```
In [11]: # Sorting ts_polarity into Positive, Negative and Neutral sentiment

sentiment = []
for score in appl_df['ts_polarity']:
    if score >= 0.05 :
          sentiment.append("Positive")
    elif score <= - 0.05 :
          sentiment.append("Negative")
    else :
        sentiment.append("Neutral")

appl_df["Sentiment"] = sentiment
appl_df.head()
```

Figure 13 Sentiment Analysis.

```
In [12]: # Sentiment Count
         appl_df['Sentiment'].value_counts()

Out[12]: Positive    785
         Neutral     134
         Negative      3
         Name: Sentiment, dtype: int64
```

Figure 14 Value counts of sentiment column

The Binary Encoding is implemented on the sentiments column and further defining the feature set and target vector. Data is then split into train and test ratio of 70:30 and the features were scaled using standardscaler. The Random Forest Classifier model is then created with n_estimators=500 and the random_state=78, thereby fitting the model using model.fit function. Model.predict function is used to make the predictions and the accuracy score is calculated using the y_test data. In the Evaluation part the model performance is evaluated using the f1 score, precision, recall and support thereby presenting the actual and predicted values.

Regression

Machine Learning algorithms deployed for the regression task are:

Random Forest Regressor

Implementation:

Random Forest Regressor is used for the APPL stock prediction, for the implementation of the model numpy and pandas libraries have been imported and with the help of sklearn the algorithm Random Forest Regressor is imported. The dataset is then imported using the import os function in the Jupyterplatform and read using the pd.read_csv function. A dataframe is created using the columns Adj close, ts_polarity, twitter_volume. Checking the missing values and then dropping those values using the function dropna,

```
In [5]: # pct change based on Adj close value
        df["Pct_change"] = df["Adj Close"].pct_change()

        # Drop null values
        df.dropna(inplace = True)
        df.head()
```

Out[5]:

	Adj Close	ts_polarity	twitter_volume	Pct_change
1	23.83	0.133635	1430.0	-0.024959
2	23.36	0.072042	1949.0	-0.019723
3	22.38	0.074369	2289.0	-0.041952
4	22.50	0.051595	2235.0	0.005362
5	22.86	0.019443	1222.0	0.016000

Figure 15 Pct change column

Window_data() function was used to create the input features X and the target vector Y. This windows_data () function consists of following parameters df, window, feature_col_number and target_col_number. Prediction of the closing price using a 3 day window of previous closing price is made with the window_size is set to 3. Further, use of 70% of the data for training and rest for testing is done. MinMaxScaler is imported using the sklearn library for rescaling the values between 0 and 1. After

the rescaling the Random Forest Regressor model is created with the n_estimators value set to 1000, then the model is fitted and the predictions are made using model.predict function. Root Mean Squared Error and Squared Error calculated for the evaluation of the model deployed.

```
In [14]: # Evaluating the model
         print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, predicted)))
         print('R-squared :', metrics.r2_score(y_test, predicted))

         Root Mean Squared Error: 0.09929035103598029
         R-squared : 0.8227285845558746
```

Figure 16 Evaluation of Random Forest Regressor

LSTM RNN

Implementation:

APPL stock prediction is made using the LSTM RNN algorithm. This method was implemented on the Jupyter platform. Libraries imported for the implementation of this algorithm was numpy, pandas, matplotlib and from sklearn metrics. In order to set the random seed for reproducibility seed was imported from the numpy.random and also random was imported from tensorflow. The dataset APPL.csv contains open, high, low, close, Adj close, Volume of Apple stock with twitter polarity scores and twitter volume and to import as well as read this data import os function along with pd.read_csv function was used. Dataframe was created by dropping the columns using df.dropna function.

In [6]:
```
# Read APPL.csv contains open, high, low, close, Adj close, Volume of Apple stock with twitter polari
import os
path = 'Z:\\DBS\\Thesis\\Sentiment-Analysis-of-Twitter-Data-for-predicting-Apple-stock-price-master'
os.chdir(path)
print(os.getcwd())
df = pd.read_csv('AAPL.csv')
df.dropna(inplace=True)
df.tail()
```

```
Z:\DBS\Thesis\Sentiment-Analysis-of-Twitter-Data-for-predicting-Apple-stock-price-master
```

Out[6]:

	Date	Open	High	Low	Close	Adj Close	Volume	ts_polarity	twitter_volume
917	2019-08-26	51.47	51.80	51.26	51.62	51.12	104174400	0.072340	888.0
918	2019-08-27	51.97	52.14	50.88	51.04	50.54	103493200	0.117541	962.0
919	2019-08-28	51.03	51.43	50.83	51.38	50.88	63755200	0.061477	895.0
920	2019-08-29	52.13	52.33	51.67	52.25	51.74	83962000	0.056460	1083.0
921	2019-08-30	52.54	52.61	51.80	52.19	51.67	84573600	0.106096	1005.0

In [7]:
```
# Dataframe with Adj close, ts_polarity, twitter_volume of APPL
df = df[["Adj Close", "ts_polarity", "twitter_volume"]]
df.head()
```

Out[7]:

	Adj Close	ts_polarity	twitter_volume
0	24.44	0.070389	1133.0
1	23.83	0.133635	1430.0
2	23.36	0.072042	1949.0
3	22.38	0.074369	2289.0
4	22.50	0.051595	2235.0

Figure 17 Creating main dataframe.

Pct change based on the Adj close value was implemented and again the detection of missing or null values is done and dropped using the dropna function.

In [8]:
```
# pct change based on Adj close value
df["Pct_change"] = df["Adj Close"].pct_change()

# Drop null values
df.dropna(inplace = True)
df.head()
```

Out[8]:

	Adj Close	ts_polarity	twitter_volume	Pct_change
1	23.83	0.133635	1430.0	-0.024959
2	23.36	0.072042	1949.0	-0.019723
3	22.38	0.074369	2289.0	-0.041952
4	22.50	0.051595	2235.0	0.005362
5	22.86	0.019443	1222.0	0.016000

Figure 18 Pct change column.

Initial step towards the data preparation was to create the target vector y and input feature as , for this the window_data() function was used. Furthermore this function consists of parameters like df, window, feature_col_number and target_col_number and chunks the data up with rolling window Xt-n to predict Xt and returns a numpy array of X and Y. For prediction the window size is set to 3. The 70% of the data was used for training and the other 30% was used for testing. MinMaxScaler was used to rescale the values between 0 and 1 further the scaler is fitted to training and testing data. The LSTM api needs to receive the features data as a vertical vector, so that we need to reshape the x data. The keras modules were imported for building and training of LSTM RNN model. The optimizer used in building the model was Adam and the loss was set to mean squared error. The model was then trained for 10 epochs with the batch_size = 5. Later the model was predicted and evaluated using model.predict and model.evaluate function. Lastly a dataframe was created of Real and Predicted values and also visualized using the plots.

```
In [19]: # Train the model
         model.fit(X_train, y_train, epochs=10, shuffle=False, batch_size=5, verbose=1)

         Epoch 1/10
         129/129 [==============================] - 5s 6ms/step - loss: 0.0070
         Epoch 2/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.1126
         Epoch 3/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0844
         Epoch 4/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0554
         Epoch 5/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0340
         Epoch 6/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0159
         Epoch 7/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0064
         Epoch 8/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0062
         Epoch 9/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0044
         Epoch 10/10
         129/129 [==============================] - 1s 5ms/step - loss: 0.0052
```

Figure 19 LSTM training.

XGBoostRegressor

Implementation:

Prediction of the APPL stock using the XGBoostRegressor model. To implement the following XGBoost was installed using pip install function on the Jupyter platform. Libraries imported like Numpy, Pandas, XGBRegressor and from sklearn metrics. The APPL data was read and loaded using pd.read_csv and import os function. Again the windows_data() function was used to create input features x and target vector y as numpy arrays. MinMaxScaler used to scale all values between 0 and 1 imported from sklearn.

```
In [24]: from sklearn.preprocessing import MinMaxScaler

In [25]: # Use the MinMaxScaler to scale data between 0 and 1.
         x_train_scaler = MinMaxScaler()
         x_test_scaler = MinMaxScaler()
         y_train_scaler = MinMaxScaler()
         y_test_scaler = MinMaxScaler()

         # Fit the scaler for the Training Data
         x_train_scaler.fit(X_train)
         y_train_scaler.fit(y_train)

         # Scale the training data
         X_train = x_train_scaler.transform(X_train)
         y_train = y_train_scaler.transform(y_train)

         # Fit the scaler for the Testing Data
         x_test_scaler.fit(X_test)
         y_test_scaler.fit(y_test)

         # Scale the y_test data
         X_test = x_test_scaler.transform(X_test)
         y_test = y_test_scaler.transform(y_test)
```

Figure 20 XG Boost model training.

Further the XGBoostRegressor instance was created with the n_estimators value set to 1000 and the model was fitted using model.fit function. Root Mean Squared and R-Squared was used to evaluate the model and a dataframe of Real and Predicted values was created.

In [31]:
```
# Create a DataFrame of Real and Predicted values
stocks = pd.DataFrame({
    "Real": real_prices.ravel(),
    "Predicted": predicted_prices.ravel()
}, index = df.index[-len(real_prices): ])
stocks.head()
```

Out[31]:

	Real	Predicted
646	46.39	47.278915
647	46.13	45.850224
648	46.22	46.335148
649	48.95	45.902985
650	50.38	48.946964

Figure 21 Real vs Predicted values.

CHAPTER 5 – RESULTS

Here we are going to discuss about a comprehensive analysis of the results obtained and their implications on using the model for future purpose. We also discuss about how our model is capable of finding better relation between the stock price and twitter sentiment than the other models. It is very important to analyse the results obtained after our model predicts for the stock price. We also check about any bias or variance in the model present. We will plot the correlation between the real stock price and predicted stock price to see how accurate we are in prediction while we also build a confusion matrix with both predicted values and real values. We will look at all our models with their respective confusion matrix and correlation plot for a better understanding of our model. Here we have acquired the results using two different approaches namely regression and classification. We built the models XG Boost, Random Forest Regressor and Long-Term Short Memory models which are regression models and we also built Random Forest Classifier and Gradient Booster models which come under the classification models. We will also compare between the regression and classification models to see which performs better.

XG Boost

XG Boost is a regression model. While implementing XG Boost we provided the dataset with complete stock market prices and the twitter volume. Then we created a dataframe with the adjacent closing price, tweets polarity and twitter volume. Later we also added a column by calculating the difference between the adjacent closing price of the stock using pct change which gives us the result by filling each row with percentage change from the immediately previous row. Model training was done by

splitting the data into 70% train data, 30% test data and by using a 3-day window of previous closing price. This gives our model enough time to see the changes in the stock price using the tweet sentiment.

By doing so we achieved Root Mean Squared error of 0.04766297161947317 and R-squared of 0.95915049000093728. We also built a plot between the real and predicted values as the following.

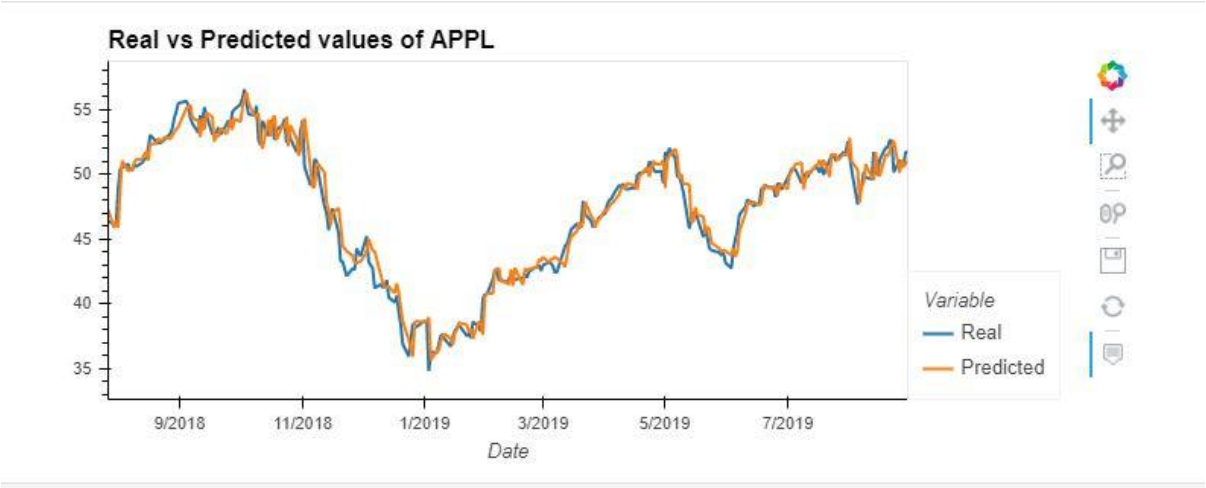

Figure 22 Correlation plot using XG Boost.

To gain further insight into our model we also built a table with the real and predicted values as such.

	Real	Predicted
646	46.39	47.278915
647	46.13	45.850224
648	46.22	46.335148
649	48.95	45.902985
650	50.38	48.946964

Figure 23 Confusion matrix of XG Boost.

Here we can see that the difference between the real and predicted values is not so huge.

Random Forest Regressor

Random Forest Regressor is a regression model. While implementing Random Forest Regressor we provided the dataset with complete stock market prices and the twitter volume. Then we created a dataframe with the adjacent closing price, tweets polarity and twitter volume. Later we also added a column by calculating the difference between the adjacent closing price of the stock using pct change which gives us the result by filling each row with percentage change from the immediately previous row. Model training was done by splitting the data into 70% train data, 30% test data and by using a 3-day window of previous closing price. This gives our model enough time to see the changes in the stock price using the tweet sentiment.

By doing so we achieved Root Mean Squared error of 0.09929035103598029 and R-squared of 0.8227285845558746. We also built a plot between the real and predicted values as the following.

Figure 24 Correlation plot of Random Forest Regressor.

Here we can see that the difference between the real and predicted values is a bit higher when compared to XG Boost model. To gain further insight into our model we also built a table with the real and predicted values as such.

	Real	Predicted
646	46.39	47.215169
647	46.13	47.215169
648	46.22	47.215169
649	48.95	47.215169
650	50.38	47.215169

Figure 25 Confusion matrix of Random Forest Regressor.

As we can see that although there is not much difference between the real and predicted values, but when compared to XG Boost model our Random Forest Regressor gives us a bit different values even though the difference is not much.

Long-Term Short Memory

LSTM is a regression model. While implementing LSTM we provided the dataset with complete stock market prices and the twitter volume. Then we created a dataframe with the adjacent closing price, tweets polarity and twitter volume. Later we also added a column by calculating the difference between the adjacent closing price of the stock using pct change which gives us the result by filling each row with percentage change from the immediately previous row. Model training was done by splitting the data into 70% train data, 30% test data and by using a 3-day window of previous closing price. This gives our model enough time to see the changes in the stock price using the tweet sentiment.

By doing so we achieved Root Mean Squared error of 0.1333873059194587 and R-squared of 0.68007098996925. We also built a plot between the real and predicted values as the following.

Figure 26 Correlation plot of LSTM.

Here we can see that the difference between the real and predicted values is a bit higher when compared to XG Boost model and a bit lower when compared to

Random Forest Regressor. To gain further insight into our model we also built a table with the real and predicted values as such.

	Real	Predicted
646	46.39	47.215169
647	46.13	47.215169
648	46.22	47.215169
649	48.95	47.215169
650	50.38	47.215169

Figure 27 Confusion matrix of LSTM.

As we can see that although there is not much difference between the real and predicted values, but when compared to XG Boost and Random Forest Regressor model our Long-Term Short Model gives us a bit different values even though the difference is not much.

Now we will be looking at the results of the classifier models which are namely Random Forest Classifier and Gradient descent model.

Random Forest Classifier

Random Forest Classifier is a classification model. While implementing Random Forest Classifier we provided the dataset with complete stock market prices and the twitter volume. Then we created a dataframe with the adjacent closing price, tweets polarity and twitter volume. Later we also added a column by calculating the sentiment of the column ts_polarity which is sentiment score of the tweets. After

calculating the new column sentiment is filled with values positive, negative and neutral based upon the score. Now we build a new column Trend with the values showing change in the adjacent closing price of the stocks. After doing so we binary encode the sentiment column splitting the data into train data and test data where the test data contains only the target variable which is our Trend column. After this we split the data into 70% train data and 30% test data. We also use standard scaler to scale both the data train and test. Model training was done by giving n_estimators = 500 and random_state = 78. This gives our model enough time to see the changes in the stock price using the tweet sentiment. Finally, we get built a confusion matrix with the real and predicted values.

	Predicted 0	Predicted 1
Actual 0	37	90
Actual 1	40	110

Figure 28 Confusion matrix of Random Forest Classifier.

We can observe that the real and predicted values have vast difference between them. For better understanding we also built a classification report showing us the accuracy, macro average, weighted average, f1 score, precision, recall and support.

Classification Report

	precision	recall	f1-score	support
0	0.48	0.29	0.36	127
1	0.55	0.73	0.63	150
accuracy			0.53	277
macro avg	0.52	0.51	0.50	277
weighted avg	0.52	0.53	0.51	277

Figure 29 Classification report of Random Forest Classifier.

From the above fig we can see that we have acquired an accuracy of 53% which is rather less compared to the regression models.

Gradient Booster Classifier

Gradient Booster is a classification model. While implementing Gradient Booster Classifier model we provided the dataset with complete stock market prices and the twitter volume. Then we created a dataframe with the adjacent closing price, tweets polarity and twitter volume. Later we also added a column by calculating the sentiment of the column ts_polarity which is sentiment score of the tweets. After calculating the new column sentiment is filled with values positive, negative and neutral based upon the score. Now we build a new column Trend with the values showing change in the adjacent closing price of the stocks. After doing so we binary encode the sentiment column splitting the data into train data and test data where the test data contains only the target variable which is our Trend column. After this we split the data into 70% train data and 30% test data. We also use standard scaler to scale the data both train data and test data. Model training was done by training the model by giving the different learning rates to the models where the models is trained in different rates, this is done by giving n_estimators = 20, learning_rate = (0.05,0.1,0.25,0.5,0.75,1), max_features = 2, max_depth = 3, random state = 0. After this the model learns the accuracy score for both training and validation. Later, when the model has been trained and we have seen that the model gives better results at 0.75 learning_rate so we create the final model with learning_rate = 0.75, n_estimators = 20, max_features = 5, max_depth = 3, random_state = 0. This gives our model enough time to see the changes in the stock price using the tweet sentiment. Finally, we get built a confusion matrix with the real and predicted values.

	Predicted 0	Predicted 1
Actual 0	42	85
Actual 1	37	113

Figure 30 Confusion matrix of Gradient Booster.

We can observe that the real and predicted values have vast difference between them. For better understanding we also built a classification report showing us the accuracy, macro average, weighted average, f1 score, precision, recall and support.

Classification Report

	precision	recall	f1-score	support
0	0.53	0.33	0.41	127
1	0.57	0.75	0.65	150
accuracy			0.56	277
macro avg	0.55	0.54	0.53	277
weighted avg	0.55	0.56	0.54	277

Figure 31 Classification report of Gradient booster model.

From the above fig we can see that we have acquired an accuracy of 56% although it is higher than the Random Forest classifier model, it is less when compared to regression models.

CHAPTER 6 – CONCLUSION

We have shown in this paper that there is a clear connection between a company's stock price rise/fall and public perceptions or emotions towards that company shared on Twitter by means of tweets. Our work's key contribution is the creation of a sentiment analyser capable of determining the form of sentiment. In the tweet, there is a lot of emotion. The tweets have been categorized as Positive, negative, and neutral are the three categories. At the moment, we believed at the outset that optimistic emotions or sentiments of the stock price of a business would be affected if information about it was made public on Twitter. The cost Our hypothesis is well supported by the outcomes and seems to have a bright future in science. It is also worth noting that we did not take into account a lot of variables in our study. First and foremost, our dataset does not reflect true public sentiment; it only considers twitter users who speak English. Second, the larger the dataset, the better the forecast, but the issue becomes more complex at the same time.

We used only twitter data to analyse people's sentiment in this study, which could be skewed since not everybody who trades stocks shares their thoughts on Twitter. Stocktwits is a financial networking forum dedicated solely to the exchange of investor, entrepreneur, and trader ideas and perspectives. Stocktwits data may be added to the current study to make it more comprehensive. In addition, data from the press can be used to compile a comprehensive public opinion survey. 922 tweets were used to train the sentiment analyser, which is a relatively small number for training a sentiment analyser. In the future, we intend to manually annotate over 10,000 tweets and train the classifiers. The models begin to do better as the size of the training datasets grows.

References

Nisar, T.M. and Yeung, M., 2018. Twitter as a tool for forecasting stock market movements: A short-window event study. The journal of finance and data science, 4(2), pp.101-119.

Pimprikar, R., Ramachadran, S. and Senthilkumar, K., 2017.Use of machine learning algorithms and twitter sentiment analysis for stock market prediction. Int J Pure Appl Math, 115(6), pp.521-526.

Pagolu, V.S., Reddy, K.N., Panda, G. and Majhi, B., 2016, October.Sentiment analysis of Twitter data for predicting stock market movements.In 2016 international conference on signal processing, communication, power and embedded system (SCOPES) (pp. 1345-150).IEEE.

Kordonis, J., Symeonidis, S. and Arampatzis, A., 2016, November. Stock price forecasting via sentiment analysis on twitter.In Proceedings of the 20th Pan-Hellenic Conference on Informatics (pp. 1-6).

Rao, T. and Srivastava, S., 2012. Analyzing stock market movements using twitter sentiment analysis.

Mittal, A. and Goel, A., 2012.Stock prediction using twitter sentiment analysis. Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf), 15.

Jadhav, R. and Wakode, M.S., 2017. Survey: Sentiment Analysis of Twitter Data for Stock Market Prediction. International Journal of Advanced Research in Computer and Communication Engineering, 6(3), pp.507-509.

Bharathi, S., Geetha, A. and Sathiynarayanan, R., 2017. Sentiment analysis of twitter and RSS news feeds and its impact on stock market prediction. International Journal of Intelligent Engineering and Systems, 10(6), pp.68-77.

Kirlić, A., Orhan, Z., Hasovic, A. and Kevser-Gokgol, M., 2018.Stock market prediction using Twitter sentiment analysis. Invention Journal of Research Technology in Engineering & Management (IJRTEM), 2(1), pp.01-04.

Usmani, M., Adil, S.H., Raza, K. and Ali, S.S.A., 2016, August.Stock market prediction using machine learning techniques.In 2016 3rd international conference on computer and information sciences (ICCOINS) (pp. 322-327).IEEE.

Agarwal, A., Xie, B., Vovsha, I., Rambow, O. and Passonneau, R.J., 2011, June.Sentiment analysis of twitter data.In Proceedings of the workshop on language in social media (LSM 2011) (pp. 30-38).

Makrehchi, M., Shah, S. and Liao, W., 2013, November. Stock prediction using event-based sentiment analysis.In 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) (Vol. 1, pp. 337-342).IEEE.

Wang, H., Can, D., Kazemzadeh, A., Bar, F. and Narayanan, S., 2012, July. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle.In Proceedings of the ACL 2012 system demonstrations (pp. 115-120).