<u>**IDS 575: MACHINE LEARNING STATISTICS**</u>

FINAL PROJECT REPORT

SENTIMENT BASED CLASSIFICATION OF TWEETS

Submitted by,
Pratibha Chitta

Table of Contents

ABSTRACT

Sentiment based classification project we are collecting the data of about 14000 tweets from twitter which are from 2012 Presidential elections debate between Obama and Romney posted by various people on twitter .With this project we aim to classify tweets based on sentiment with different classes: positive (1), negative (-1), or neutral (0). We present the results of machine learning algorithms for classifying the sentiment of tweets using distant supervision. We aim to use various machine learning algorithms (Naive Bayes, Decision Tree, SVM, KNN) on Obama and Romney tweets to classify based on opinion expressed in tweets. This paper also describes the preprocessing steps needed such as word2vec and Tf-Idf vectorization process in order to achieve high accuracy. Apart from that, we evaluate different machine learning algorithms and compare using a confusion matrix to evaluate the Precision, Recall, F-score and other relevant metrics. The main contribution of this paper is the idea of using tweets with sentiment based classification for distant supervised learning and unsupervised learning.

INTRODUCTION

Speech is the most widely used form of communication. Lately, the advent of efficient processing technology has helped boost the related research in the field of artificial intelligence and Machine learning. We have decided on conducting a polarity analysis of tweets on twitter. Determining the sentiment polarity of tweets has become a landmark homework exercise in data science classes. Firstly, the problem is based on the works of different pioneers in NLP. It starts off by converting the corpus of the word dictionary found by all the words in the tweets that we have collected into numbers that a machine can understand better. Secondly, the problem dives deeper into classification of speech that arises from the dataset rather than the text processing into numbers. Sentiment analysis often wants to identify trends within the content of tweets, which are then analyzed by machine learning algorithms. Classification, which is employed for sentiment analysis, is an automatic system that must be fed sample text before returning a category, e.g. positive, negative, or neutral.

The main objective is classification of the sentiment or opinion expressed in tweets into one of the three classes: positive (1), negative (-1), or neutral (0). We gathered the data of about 14000 tweets from twitter which are from the 2012 Presidential elections debate between Obama and Romney posted by various people on twitter. After that, We have applied preprocessing steps on those tweets such as word2vec conversion and TF-IDF vectorization process. In the training data, we have collected all these tweets into 3 different classes: positive (1), negative (-1), or neutral (0). We divide the dataset into 2 sets namely train and test sets respectively at random (80:20 ratio) and evaluate different machine learning algorithms and compare using a confusion matrix to evaluate the Precision, Recall, F-score and other relevant metrics. Such as Precision, recall and F1 score of the positive class and negative class.

RELATED WORK

Sentiment analysis in the field of microblogging can be a relatively new research topic, so there is still a lot of room for research in this area. A significant amount of related preparatory work has been done in terms of user reviews, weblog/article sentiment analysis, and phrase-level sentiment analysis. They differ from Twitter primarily because of the 140 character limit per tweet. This forces the user to express a specific opinion condensed into a very short text.

There is a lot of research in the field of sentiment analysis. Some of the first results on sentiment analysis of Twitter data are from Go et al[1]. People who used distance learning to collect sentiment data. They used tweets containing positive emotions such as ":)" and ";)" as positive, and tweets containing negative emoji such as ":(" as negative. They build models using Naive Bayes, KNN and SVM Classifiers and for the functionality we used Preprocessing steps TF-Idf Vectorization and word2vec conversion. They note that the Preprocessing is required before applying the machine learning models. A research report by Pang and Lee on opinion Mining and Sentiment Analysis [2] provides comprehensive research in blogs, reviews, and other fields related to sentiment analysis. Algorithms used in the survey include Decision Tree, SVM, KNN and Naive Bayes.

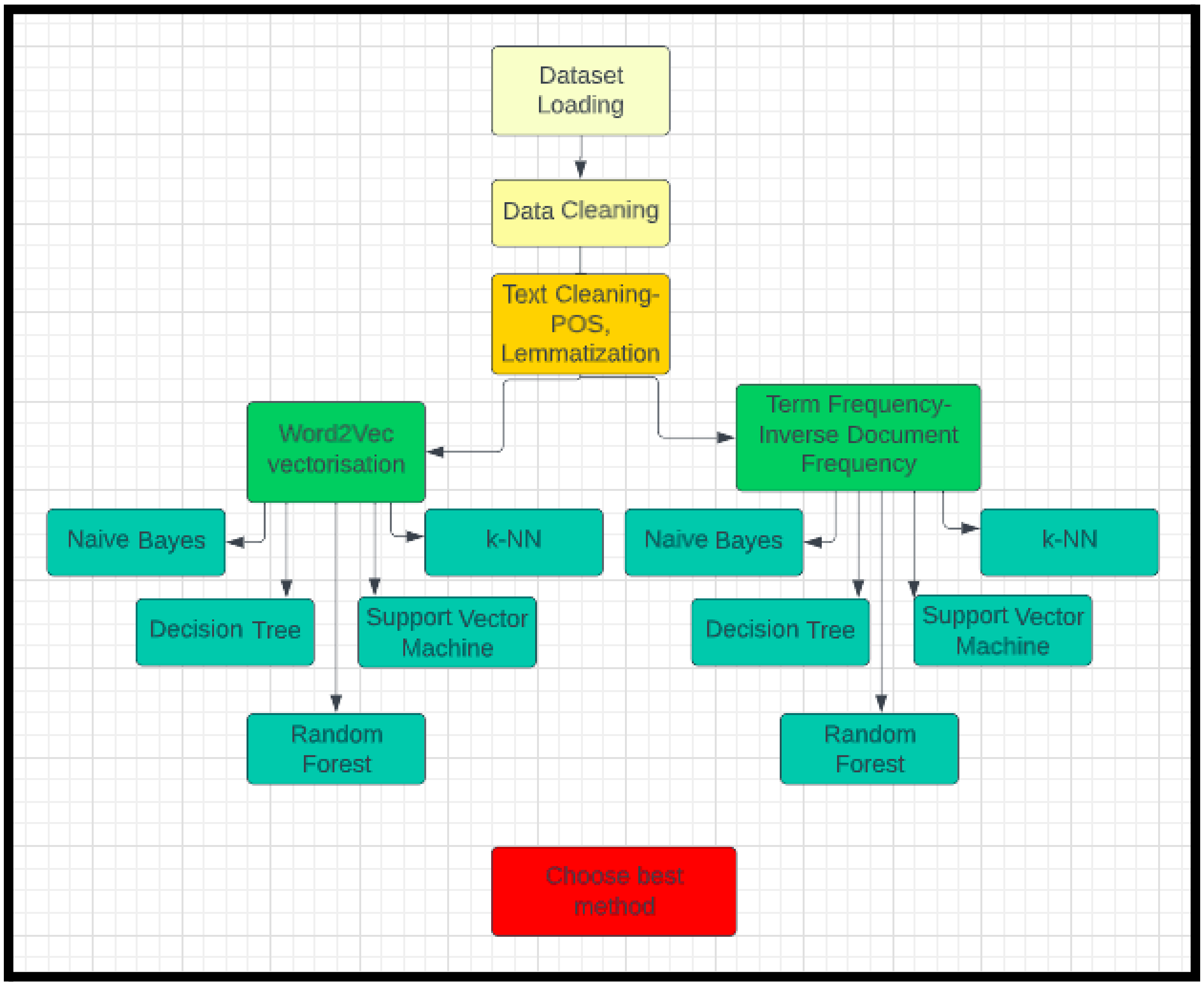
The practical applications of this task are wide, from monitoring popular events (e.g. IMDB movie reviews, Oscars, Offensive language analysis on twitter.) to extracting trading signals by monitoring tweets about public companies. It offers many challenging opportunities to develop new applications, mainly due to the huge growth of available information on online sources like blogs and social networks. For example, recommendations of items proposed by a recommendation system can be predicted by taking into account considerations such as positive or negative opinions about those items by making use of Sentiment analysis[3].

PROBLEM STATEMENT

The main objective of this project is to use Machine Learning Algorithms to predict the sentiment of the tweets trained on the tweets of Obama and Romney which were posted on Twitter during the 2012 Presidential elections. In order to predict the sentiment of a text, it is important that strings are converted into vectors for the machine to actually train the sentiment it represents.

Vectorisation is an essential part of the project. There are various ways to vectorisation (or conversion) to numerical value. The top two methods are calculating the Term Frequency-Inverse Document Frequency or using the deep learning algorithm of Word2Vec / Doc2Vec methodology. Therefore we have divided the project into two categories and it aims to:

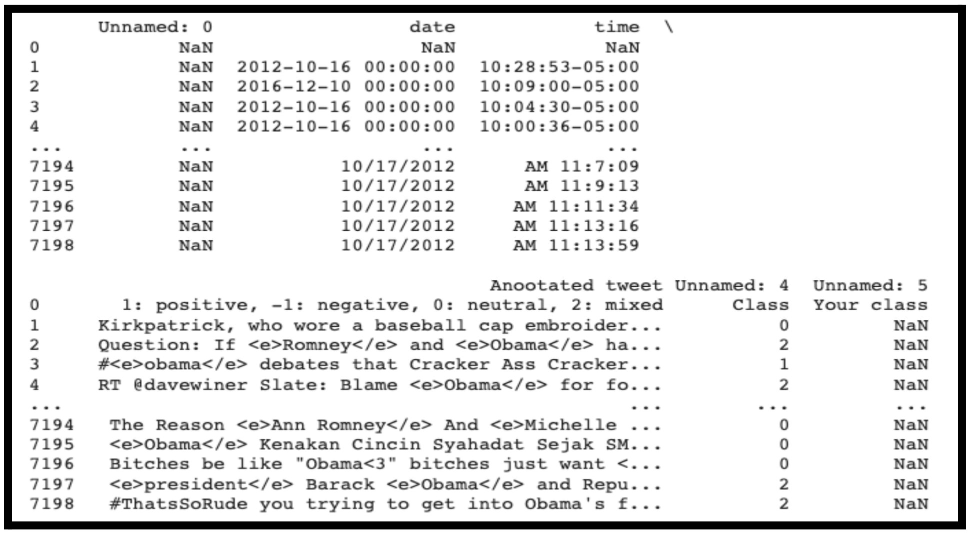
1. Compare the accuracy of machine learning algorithms based on TF-IDF vectorisation and Word2Vec vectorisation to conclude the efficiency of the methodology
2. Find the best model with maximum accuracy to predict the sentiment of tweets



DATA EXPLORATION

The dataset file we found from the Kaggle repository contained an Excel file with two sheets, one containing the tweets about Obama and the second sheet containing the tweets about Romney. The tweets were collected after the debate between 2012 Presidential election candidates, Barack Obama and Mitt Romney. We aim to build a predictive model to classify the tweet as a positive, negative or neutral sentiment. This is a multi-class classification problem with three classes (-1: negative, 0: neutral, 1: positive).

The first step for us was to load the data. We combined the two sheets in Python itself and the dataset we got a dataframe of 14398 rows and 6 columns. The dataset contained the data, time of the tweets as well. The data looked like:



```
      Unnamed: 0                 date            time  \
0            NaN                  NaN             NaN
1            NaN  2012-10-16 00:00:00  10:28:53-05:00
2            NaN  2016-12-10 00:00:00  10:09:00-05:00
3            NaN  2012-10-16 00:00:00  10:04:30-05:00
4            NaN  2012-10-16 00:00:00  10:00:36-05:00
...          ...                  ...             ...
7194         NaN           10/17/2012      AM 11:7:09
7195         NaN           10/17/2012      AM 11:9:13
7196         NaN           10/17/2012     AM 11:11:34
7197         NaN           10/17/2012     AM 11:13:16
7198         NaN           10/17/2012     AM 11:13:59

                                     Anootated tweet Unnamed: 4  Unnamed: 5
0       1: positive, -1: negative, 0: neutral, 2: mixed      Class  Your class
1     Kirkpatrick, who wore a baseball cap embroider...          0         NaN
2     Question: If <e>Romney</e> and <e>Obama</e> ha...          2         NaN
3     #<e>obama</e> debates that Cracker Ass Cracker...          1         NaN
4     RT @davewiner Slate: Blame <e>Obama</e> for fo...          2         NaN
...                                                 ...        ...         ...
7194   The Reason <e>Ann Romney</e> And <e>Michelle ...          0         NaN
7195   <e>Obama</e> Kenakan Cincin Syahadat Sejak SM...          0         NaN
7196   Bitches be like "Obama<3" bitches just want <...          0         NaN
7197   <e>president</e> Barack <e>Obama</e> and Repu...          2         NaN
7198   #ThatsSoRude you trying to get into Obama's f...          2         NaN
```

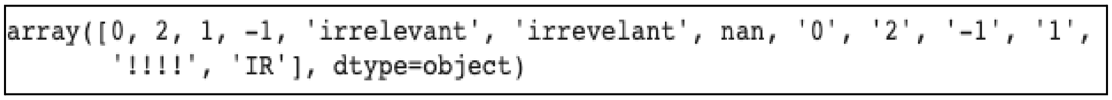
In order to process the data for building machine learning models, we clean our dataset. To clean the dataset and remove irrelevant data we follow the following steps:

I. **<u>Drop Irrelevant Columns</u>**: We remove the 'Unnamed: 0', 'date', 'time', 'Unnamed: 5' columns as these won't contribute as dependent variables for our predictive models.

II. **<u>Rename Columns</u>**: We rename the columns 'Anootated tweet' as 'tweet' and 'Unnamed: 4' as 'class'.

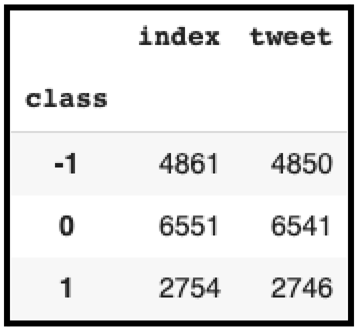
III. **<u>Removing Null values</u>**: We find 2 null values in the 'tweet' column and 34 null values in 'class' which are removed.

IV. **<u>Removing rows with irrelevant</u>**: We check the unique outputs of our categorical target variable.



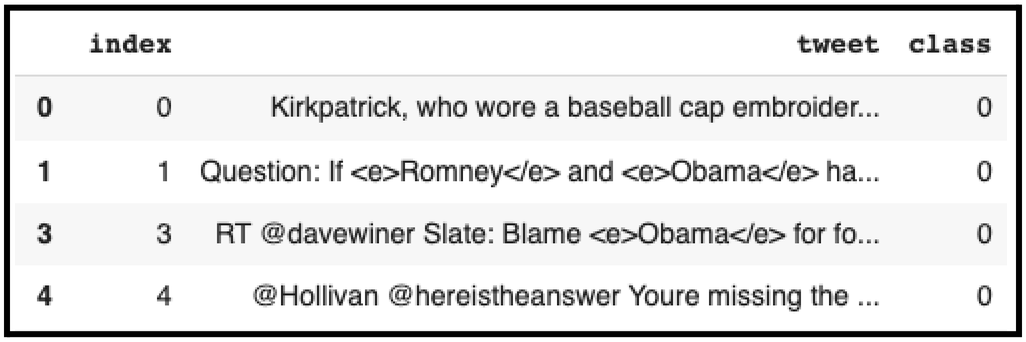
```
array([0, 2, 1, -1, 'irrelevant', 'irrevelant', nan, '0', '2', '-1', '1',
       '!!!!', 'IR'], dtype=object)
```

We remove rows with classes ‘2’, ‘irrelevant’, irrelevant’, ‘nan’, ‘!!!!’, “ ‘2’ “, “ ‘0’ “, “ ‘1’ “, “ ‘-1’ “ , ‘IR’. After removing these classes and null values in tweet or class column, we are left with the following classes and their number of training data:



	index	tweet
class		
-1	4861	4850
0	6551	6541
1	2754	2746

After cleaning the dataset and keeping only the dependent variable (‘tweet’) and independent variable (‘class’) in the dataset, we are ready to work with the dataset to perform sentiment analysis.



	index	tweet	class
0	0	Kirkpatrick, who wore a baseball cap embroider...	0
1	1	Question: If <e>Romney</e> and <e>Obama</e> ha...	0
3	3	RT @davewiner Slate: Blame <e>Obama</e> for fo...	0
4	4	@Hollivan @hereistheanswer Youre missing the ...	0

CLEANING TWEETS FOR TEXT ANALYTICS

For sentiment analysis, we need to process text and for that we need to perform some pre-processing. We need to clean tweets and we remove:

I. url with regular expressions (https?://\S+|www\.\S+)
II. emails (`'\S*@\S*\s?', ''`)
III. new line characters (`'\s+', ' '`)
IV. distracting single quotes (“\ “)
V. special characters
VI. change to lowercase

TEXT NORMALIZATION

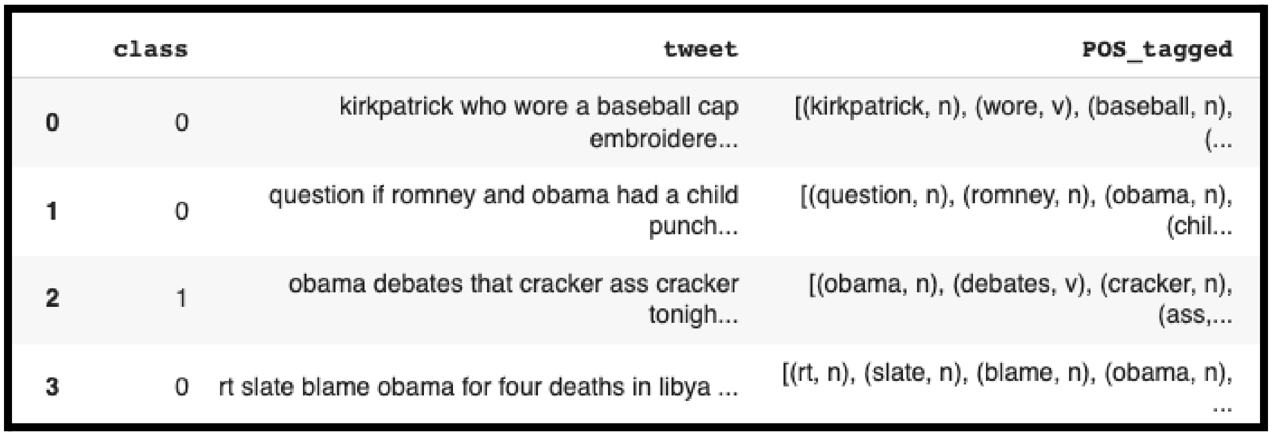
PART OF SPEECH TAGGING

To train our models for sentiment analysis, we first identify each word in the tweet and attach the label of each word, whether it is a noun, adjective, verb, proverb etc. They can often get quite specific, also distinguishing e.g. between types of nouns (**proper nouns** etc). This is important for the machine to understand the context in which word is being used in the sentence. This step is important to perform lemmatization on tweets.

Essentially saying that words in different contexts can have different meanings. This is of course a massive gain in information that we can pass to a model!

The word **duck** can be a noun (the bird) or a verb (the motion, to crouch down). If we can tell a model which one of these it is in a given sentence, the model can learn to make a lot more sense out of the sentence.

For this, we have used NTLK corpus. The result we get:



	class	tweet	POS_tagged
0	0	kirkpatrick who wore a baseball cap embroidere...	[(kirkpatrick, n), (wore, v), (baseball, n), (...
1	0	question if romney and obama had a child punch...	[(question, n), (romney, n), (obama, n), (chil...
2	1	obama debates that cracker ass cracker tonigh...	[(obama, n), (debates, v), (cracker, n), (ass,...
3	0	rt slate blame obama for four deaths in libya ...	[(rt, n), (slate, n), (blame, n), (obama, n), ...

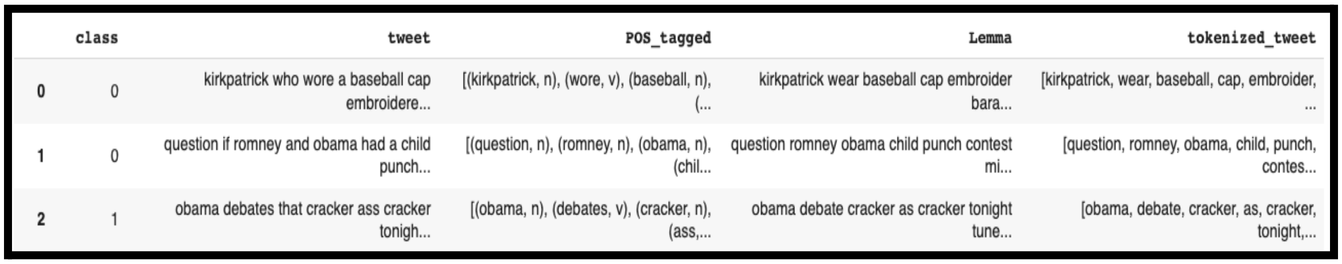
LEMMATIZATION

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meanings to one word.

The purpose of lemmatization is the same as that of stemming but overcomes the drawbacks of stemming. In stemming, for some words, it may not give may not give meaningful representation such as "Histori". For instance, stemming the word 'Caring' would return 'Car'. While lemmatization would return 'Care'. In this case, lemmatization is more successful. Thus, we chose lemmatization to normalize our text.

To Lemmatize our tweets, we use ntlk.WordNetLemmatizer. After lemmatization, we have tokenized the tweets in order to later convert it into numerical value for the machine to understand.

The result we get after lemmatization and tokenization is:



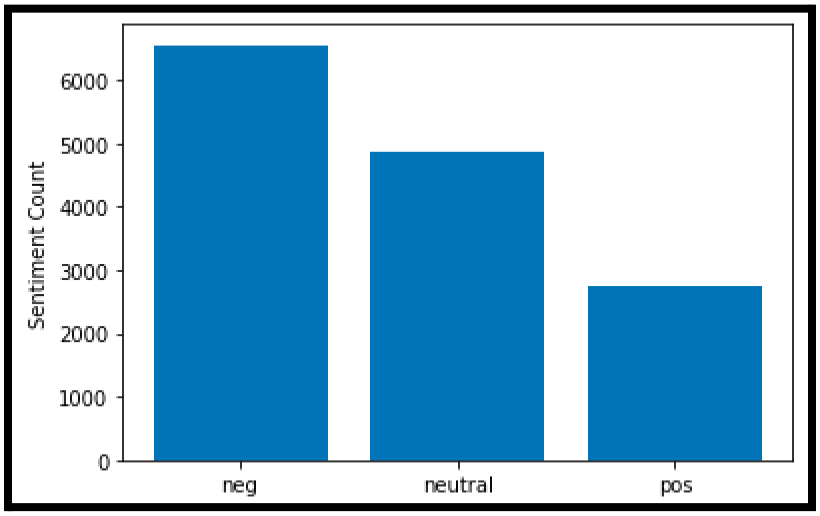
	class	tweet	POS_tagged	Lemma	tokenized_tweet
0	0	kirkpatrick who wore a baseball cap embroidere...	[(kirkpatrick, n), (wore, v), (baseball, n), (...	kirkpatrick wear baseball cap embroider bara...	[kirkpatrick, wear, baseball, cap, embroider, ...
1	0	question if romney and obama had a child punch...	[(question, n), (romney, n), (obama, n), (chil...	question romney obama child punch contest mi...	[question, romney, obama, child, punch, contes...
2	1	obama debates that cracker ass cracker tonigh...	[(obama, n), (debates, v), (cracker, n), (ass,...	obama debate cracker as cracker tonight tune...	[obama, debate, cracker, as, cracker, tonight,...

After lemmatization, it is important to convert words into vectors. For vectorisation, we use and compare two kinds of vectorisation(word2vec and term frequency-inverse document frequency) and compare the accuracies of the model based on those vectorisations.

DATA VISUALISATIONS

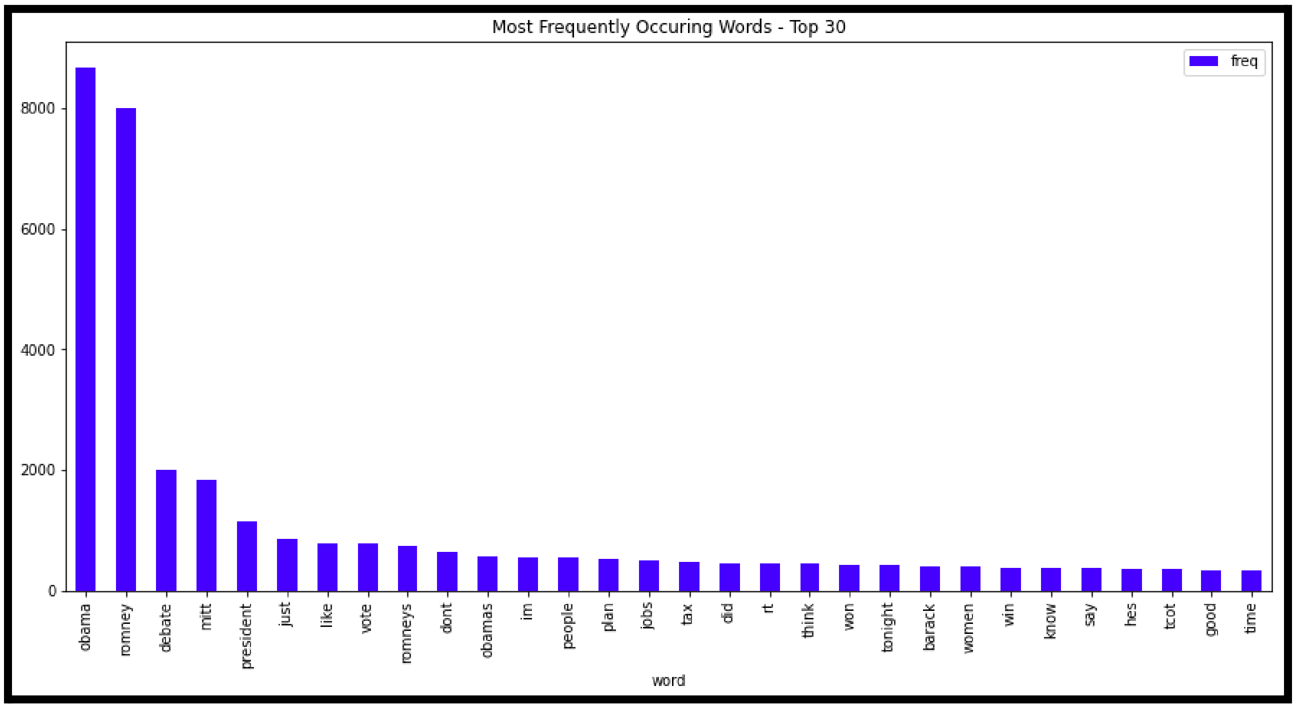
We explore our data using visualizations. We get the following observations.

Frequency of Tweets



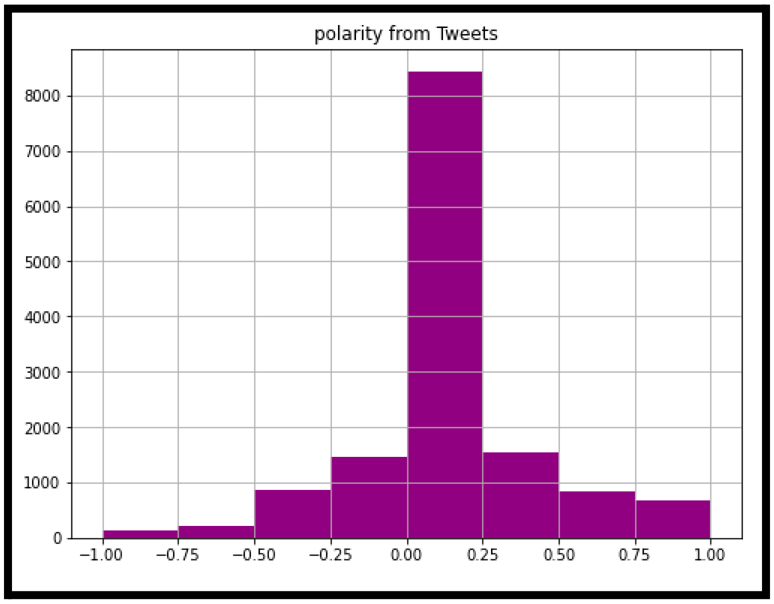
Observation: We have maximum negative tweets in our training dataset, followed by neutral tweets and positive tweets.

Frequency of Most-appeared Words



Observation: Obama is the maximum recurring word in the tweets, followed by Romney, debate, mitt, president etc.

Polarity of Tweets

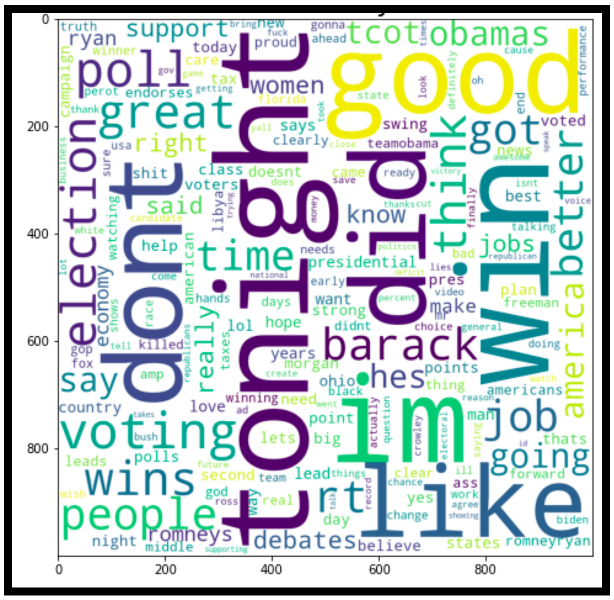


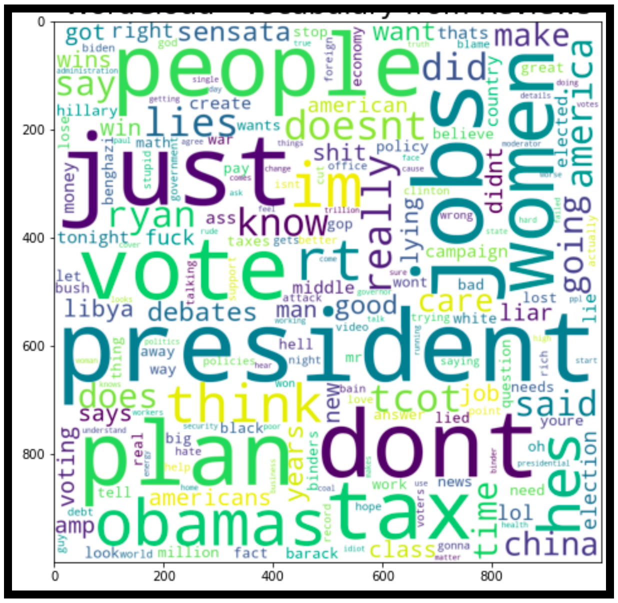
Observation: We find the maximum number of tweets lie between polarity of 0 to 0.25.

Tweets WordCloud

We have removed words that occurred more than 200 times.

```
[('obama', 1859),
 ('romney', 1149),
 ('debate', 439),
 ('president', 320),
 ('mitt', 258),
 ('vote', 252),
 ('won', 234),
 ('just', 150),
 ('tonight', 148),
 ('win', 148),
```





Positive-Sentiment Tweets **Negative-Sentiment Tweets**

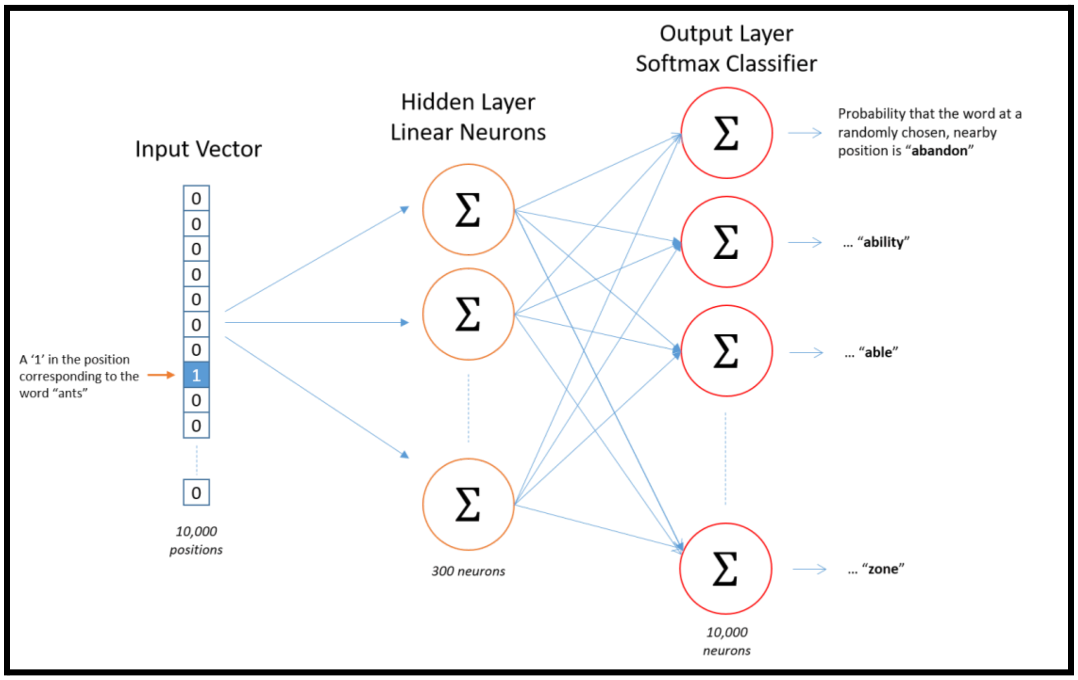
Observation: We find that tonight, good, win, did, like are maximum used in positive tweets. Words like just, vote, president, women, tax are maximum words used in negative tweets.

SUPERVISED LEARNING

We performed supervised learning through vectorization using two methods which are word2vec and term frequency-inverse document frequency. To convert text data into numerical data, we need vectorization, or in the NLP world, word embeddings. Bag of Words converts a corpus document into a numeric vector by mapping each document word to a feature vector for the machine learning model.

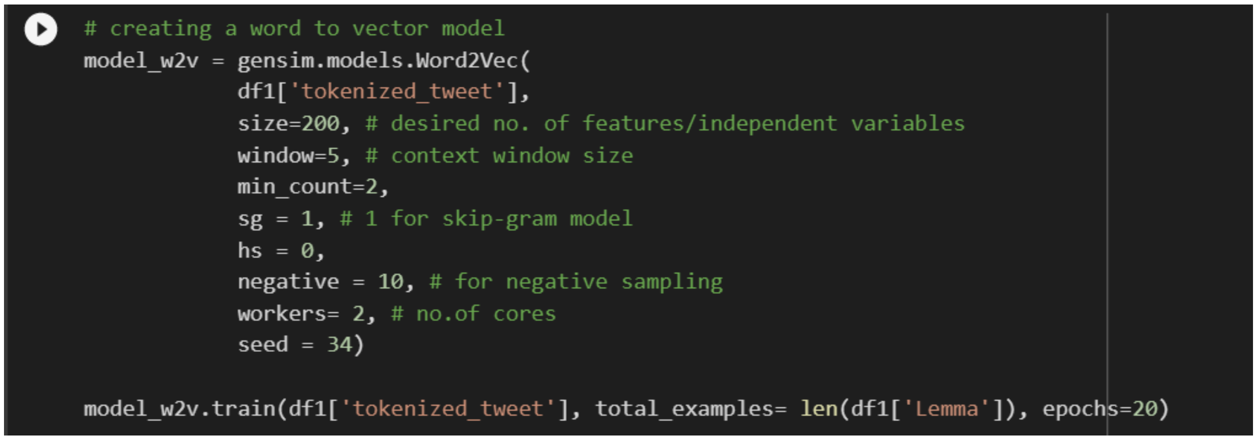
WORD2VEC VECTORIZATION

Word2vec is a collection of related models used to generate word embeddings. These are shallow, two-layer neural networks that have been trained to reconstruct word linguistic contexts. It generates a vector space. Each distinct word in the corpus that shares common contexts with other words is clustered together in the space. Following training, word2vec models can be used to map each word to a vector of hundreds of elements that represents that word's relationship to other words. The hidden layer of the neural network is represented by this vector.



Word2vec generates neural word embeddings using a continuous bag of words (CBOW). Gensim is a Python library that has some of the awesome features needed for text processing. We used the Gensim Library. It enabled us to develop word embeddings by training our own models.

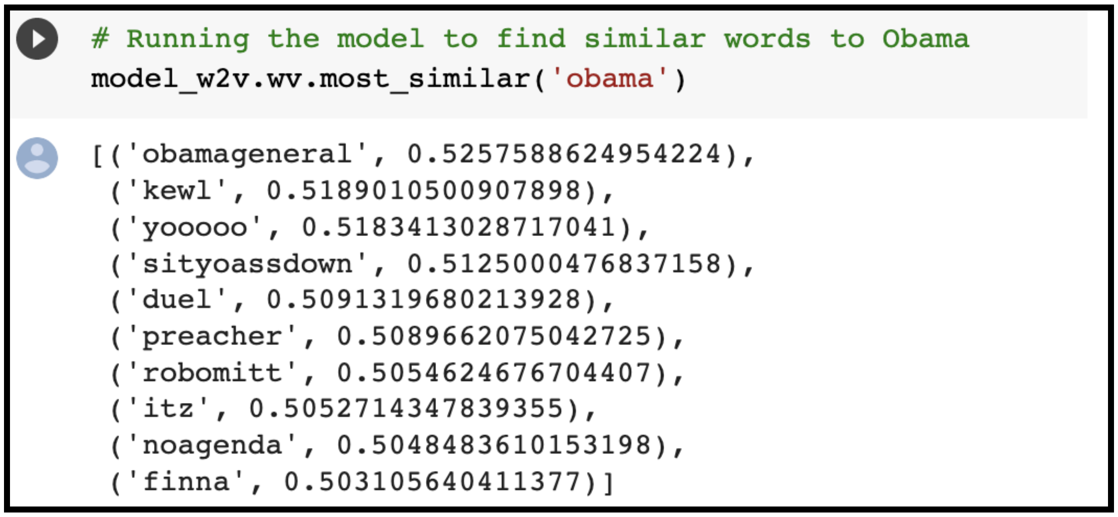
This model's hyperparameters are as follows:
size: The number of dimensions are set to 100 by default.
window: The maximum distance between a target word and the words surrounding it. By default it is 5.
min_count: The minimum number of words to consider when training the model; words with fewer than this number of occurrences will be ignored. The default for min_count is 5.
workers: The number of partitions during training and the default workers is 3.
sg: The training algorithm, either CBOW(0). The default training algorithm is CBOW.



```
# creating a word to vector model
model_w2v = gensim.models.Word2Vec(
            df1['tokenized_tweet'],
            size=200, # desired no. of features/independent variables
            window=5, # context window size
            min_count=2,
            sg = 1, # 1 for skip-gram model
            hs = 0,
            negative = 10, # for negative sampling
            workers= 2, # no.of cores
            seed = 34)

model_w2v.train(df1['tokenized_tweet'], total_examples= len(df1['Lemma']), epochs=20)
```

CHECK SIMILARITY

A similarity measure takes embeddings and returns a number measuring their similarity embeddings are simply vectors of numbers also known as vector dimensions. Similarity between two vectors is nothing but the distances from the two vectors. Using the Gensim library we were able to compute the similar words to ‘Obama’

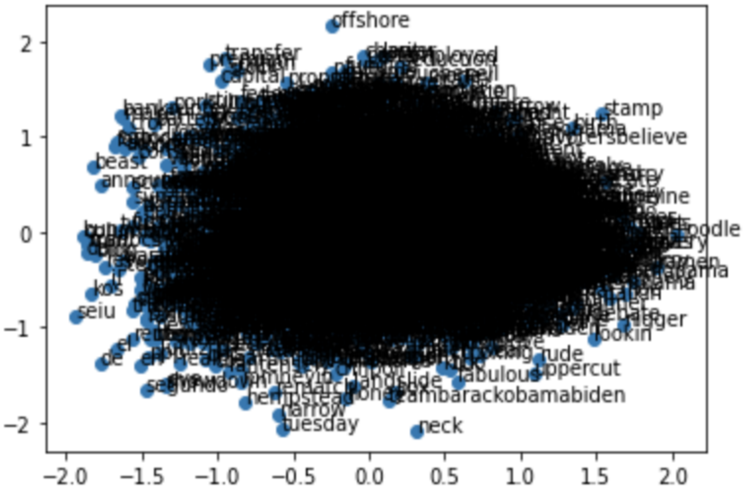


```
# Running the model to find similar words to Obama
model_w2v.wv.most_similar('obama')
```

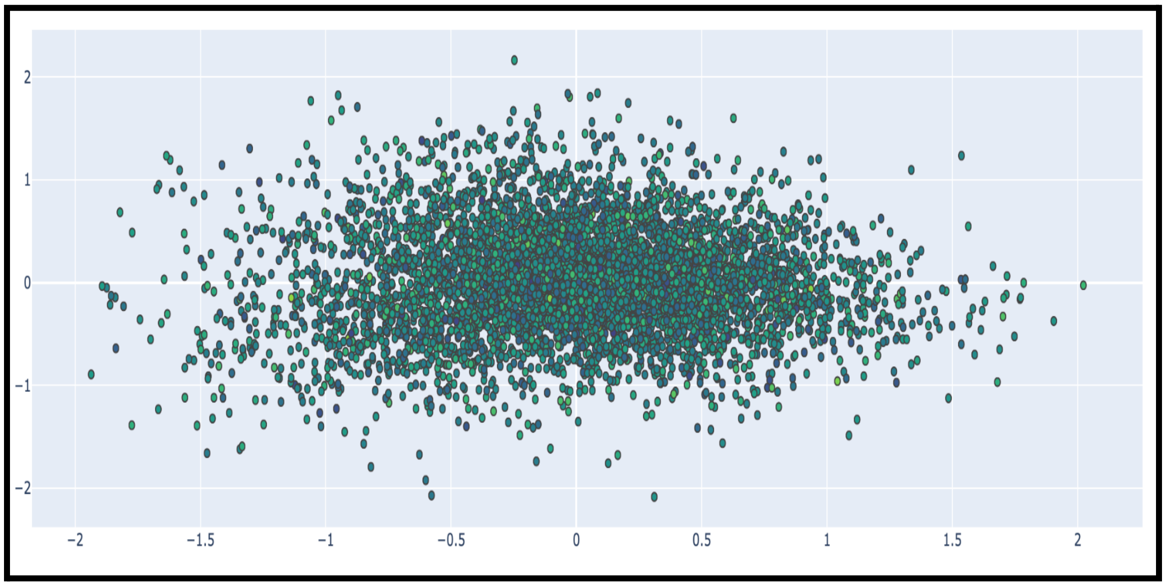
```
[('obamageneral', 0.5257588624954224),
 ('kewl', 0.5189010500907898),
 ('yooooo', 0.5183413028717041),
 ('sityoassdown', 0.5125000476837158),
 ('duel', 0.5091319680213928),
 ('preacher', 0.5089662075042725),
 ('robomitt', 0.5054624676704407),
 ('itz', 0.5052714347839355),
 ('noagenda', 0.5048483610153198),
 ('finna', 0.503105640411377)]
```

VISUALIZATION OF WORD2VEC

Here are the results of visualization by running Principal component analysis to the vector cluster words. PCA enables the visualization of multidimensional data.



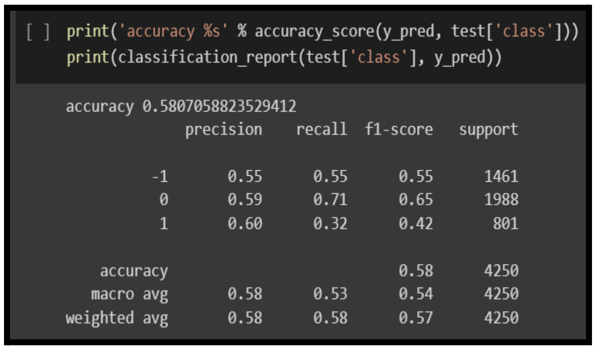
This is way too clumsy. So we use plot py instead of matplotlib for better interactive visualization. We used plot py instead of matplotlib for better interactive visualization.we get the following output.



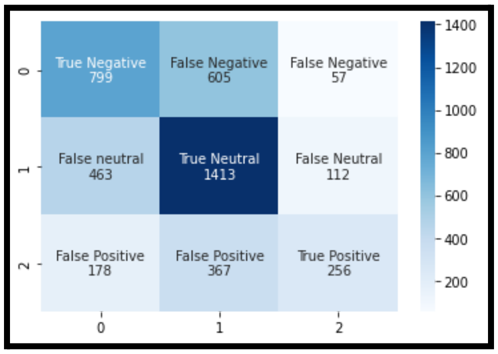
MODELS BASED ON WORD2VEC

1. Logistic Regression

Classification report:

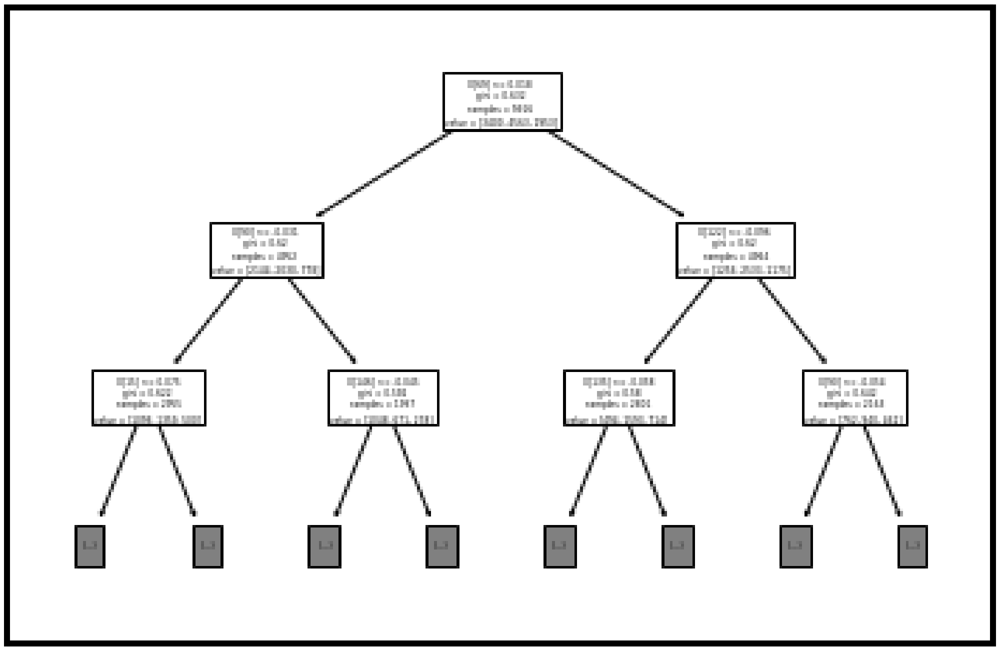


```
[ ] print('accuracy %s' % accuracy_score(y_pred, test['class']))
    print(classification_report(test['class'], y_pred))

    accuracy 0.5807058823529412
                  precision    recall  f1-score   support

              -1       0.55      0.55      0.55      1461
               0       0.59      0.71      0.65      1988
               1       0.60      0.32      0.42       801

        accuracy                           0.58      4250
       macro avg       0.58      0.53      0.54      4250
    weighted avg       0.58      0.58      0.57      4250
```

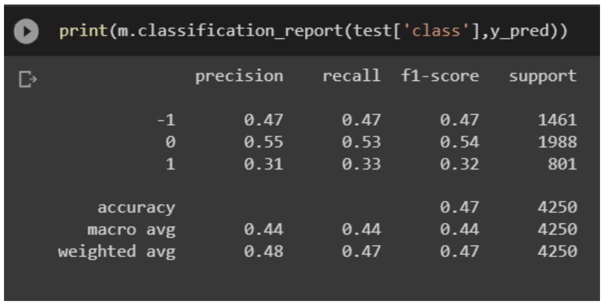


2. Decision Tree

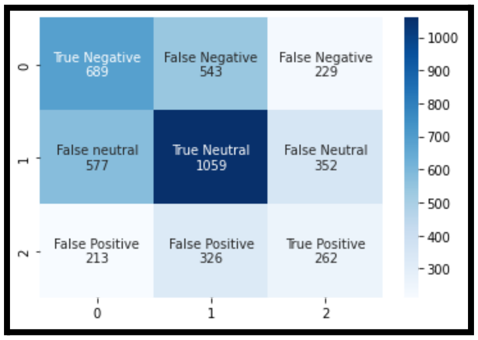
Plotting the decision tree



Classification report:

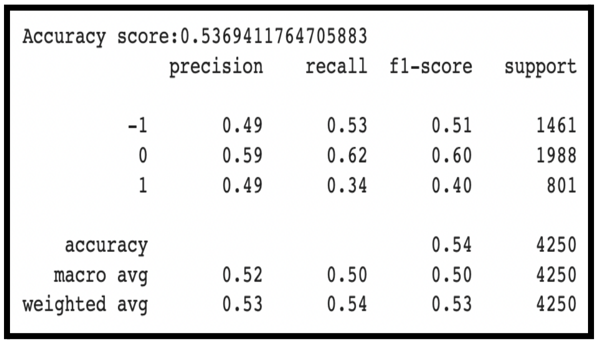


```
print(m.classification_report(test['class'],y_pred))

              precision    recall  f1-score   support

          -1       0.47      0.47      0.47      1461
           0       0.55      0.53      0.54      1988
           1       0.31      0.33      0.32       801

    accuracy                           0.47      4250
   macro avg       0.44      0.44      0.44      4250
weighted avg       0.48      0.47      0.47      4250
```

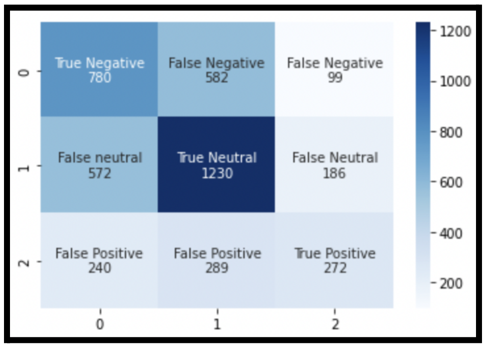


3. K-Nearest Neighbour

Classification report:

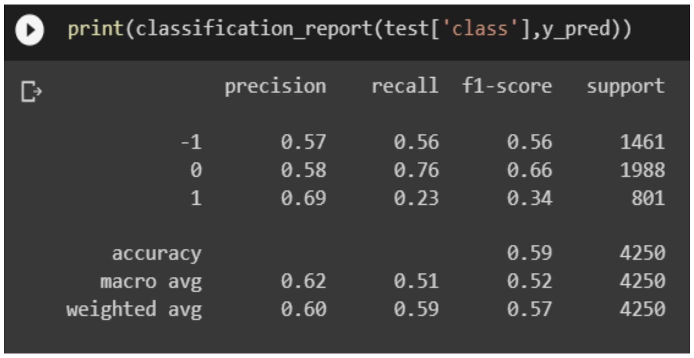


```
Accuracy score:0.5369411764705883
              precision    recall  f1-score   support

          -1       0.49      0.53      0.51      1461
           0       0.59      0.62      0.60      1988
           1       0.49      0.34      0.40       801

    accuracy                           0.54      4250
   macro avg       0.52      0.50      0.50      4250
weighted avg       0.53      0.54      0.53      4250
```

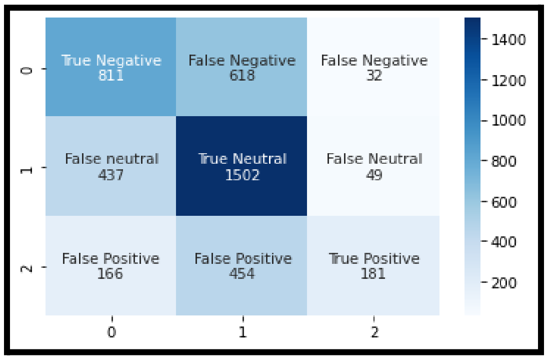


4. Random Forest

Classification report:

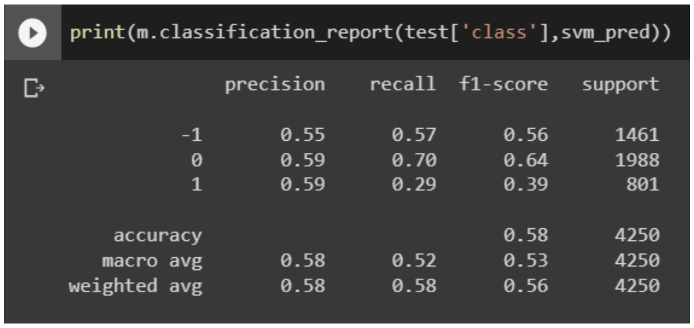


```
print(classification_report(test['class'],y_pred))

              precision    recall  f1-score   support

          -1       0.57      0.56      0.56      1461
           0       0.58      0.76      0.66      1988
           1       0.69      0.23      0.34       801

    accuracy                           0.59      4250
   macro avg       0.62      0.51      0.52      4250
weighted avg       0.60      0.59      0.57      4250
```

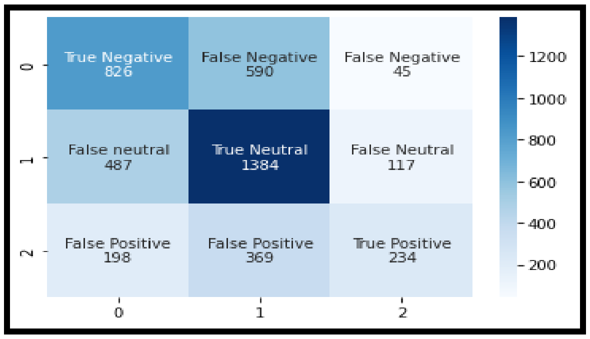


5. Support vector machine

Classification Report:



```
print(m.classification_report(test['class'],svm_pred))

              precision    recall  f1-score   support

          -1       0.55      0.57      0.56      1461
           0       0.59      0.70      0.64      1988
           1       0.59      0.29      0.39       801

    accuracy                           0.58      4250
   macro avg       0.58      0.52      0.53      4250
weighted avg       0.58      0.58      0.56      4250
```



TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF) VECTORIZATION

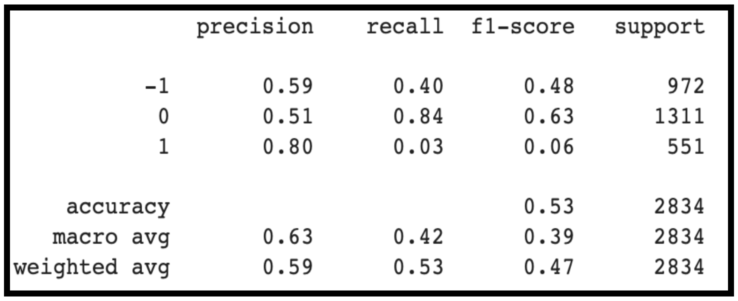
TF-IDF is used to weigh a keyword and based on the number of times a word appears, it assigns importance to that particular word. The TF-IDF value increases proportionally to the number of times a word appears in a document, which refers to each text data point, and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. A score between 0 and 1 is assigned to each word. If the assigned weight is smaller then, it means that it is a more common word. On the other hand, a word having a higher numerical weight value simply means that this term is rare.

Therefore, TF-IDF vectorization gives high importance to words which are frequent in a document and rare in the corpus. The reason for using TF-IDF vectorization in our research is to find out which word appears in each tweet and its significance in each tweet.

MODELS BASED ON TF-IDF

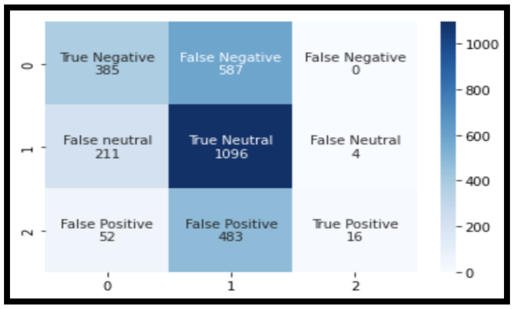
1. **Naïve Bayes:**

Classification report:



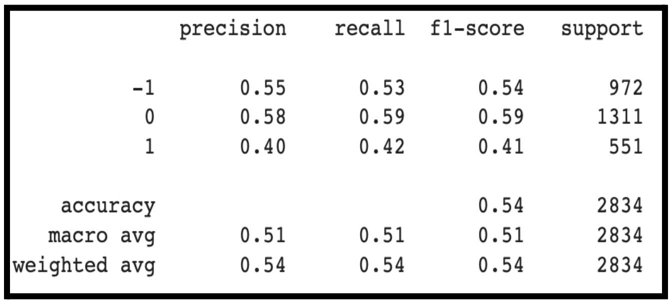
```
              precision    recall  f1-score   support

          -1       0.59      0.40      0.48       972
           0       0.51      0.84      0.63      1311
           1       0.80      0.03      0.06       551

    accuracy                           0.53      2834
   macro avg       0.63      0.42      0.39      2834
weighted avg       0.59      0.53      0.47      2834
```

Confusion Matrix:



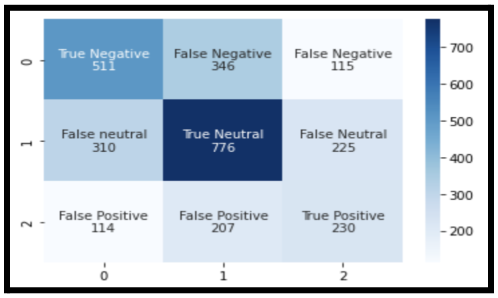
2. **Logistic Regression:**

Classification report:



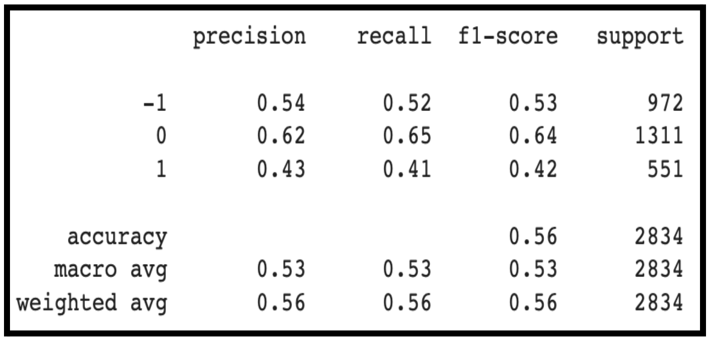
```
              precision    recall  f1-score   support

          -1       0.55      0.53      0.54       972
           0       0.58      0.59      0.59      1311
           1       0.40      0.42      0.41       551

    accuracy                           0.54      2834
   macro avg       0.51      0.51      0.51      2834
weighted avg       0.54      0.54      0.54      2834
```

Confusion matrix:



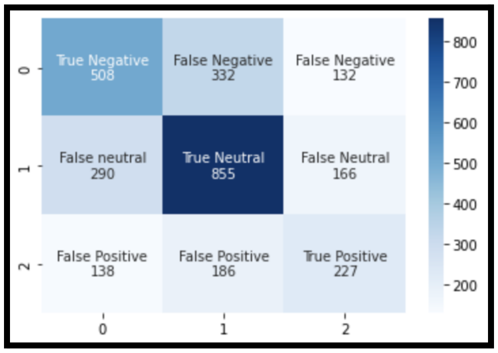
3. **Decision Tree:**

Classification report:



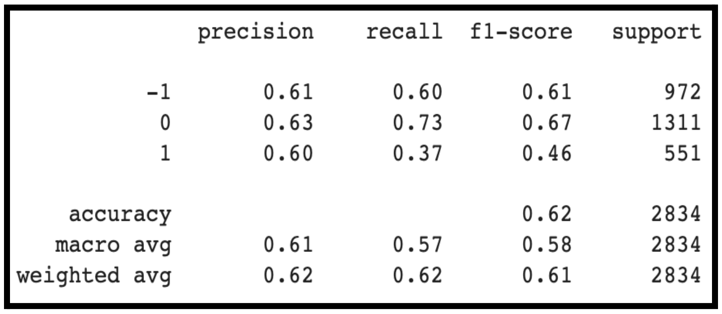
```
              precision    recall  f1-score   support

          -1       0.54      0.52      0.53       972
           0       0.62      0.65      0.64      1311
           1       0.43      0.41      0.42       551

    accuracy                           0.56      2834
   macro avg       0.53      0.53      0.53      2834
weighted avg       0.56      0.56      0.56      2834
```

Confusion Matrix:



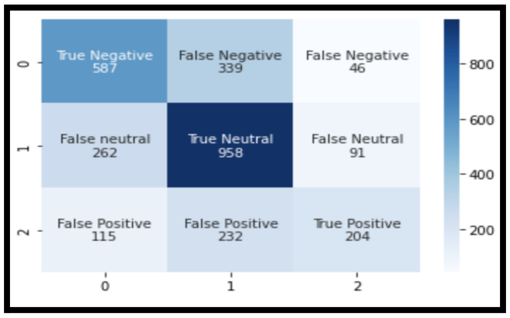
4. **Random Forest:**

Classification report:



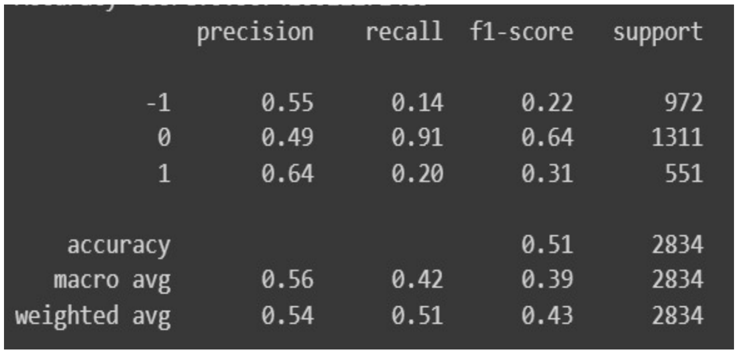
```
              precision    recall  f1-score   support

          -1       0.61      0.60      0.61       972
           0       0.63      0.73      0.67      1311
           1       0.60      0.37      0.46       551

    accuracy                           0.62      2834
   macro avg       0.61      0.57      0.58      2834
weighted avg       0.62      0.62      0.61      2834
```

Confusion Matrix:



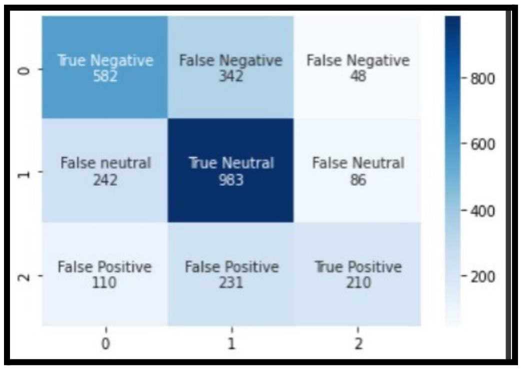
5. **K-Nearest Neighbor (kNN):**

Classification report:



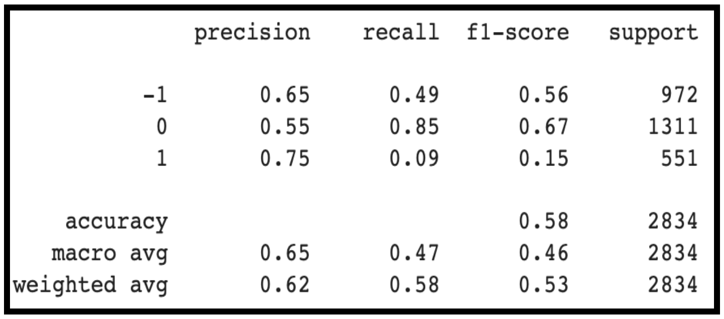
```
              precision    recall  f1-score   support

          -1       0.55      0.14      0.22       972
           0       0.49      0.91      0.64      1311
           1       0.64      0.20      0.31       551

    accuracy                           0.51      2834
   macro avg       0.56      0.42      0.39      2834
weighted avg       0.54      0.51      0.43      2834
```

Confusion Matrix:



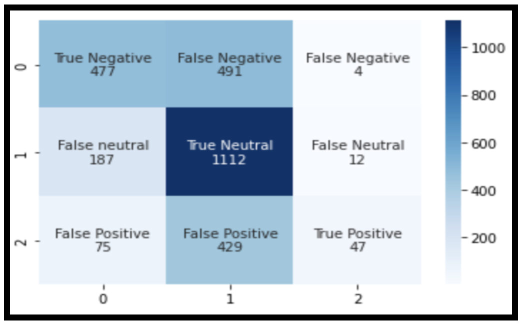
6. **Support Vector Machine (SVM):**

Classification report:



```
              precision    recall  f1-score   support

          -1       0.65      0.49      0.56       972
           0       0.55      0.85      0.67      1311
           1       0.75      0.09      0.15       551

    accuracy                           0.58      2834
   macro avg       0.65      0.47      0.46      2834
weighted avg       0.62      0.58      0.53      2834
```

Confusion Matrix:



UNSUPERVISED LEARNING

We decided to perform a cluster analysis on the dataframe to see if unsupervised learning produced better results and if it was useful for our dataset. Because our dataset contained three different types of classes, the value of k = 3. We used Principal Component Analysis to implement k-means. Because the dataset is too large to visualize clusters, we can reduce the dimensions of our dataset to any number less than the current number of features with the help of PCA. Another application of PCA is to compress data and thus save computational time. The goal is to find interesting patterns in the data, such as whether there are any subgroups or 'clusters' among the tweets.

Certain factors can impact the efficacy of the final clusters formed when using k-means clustering

SILHOUETTE ANALYSIS:

The silhouette coefficient, also known as the silhouette score k-means, measures how similar a data point is within a cluster (cohesion) when compared to other clusters (separation). The equation for calculating the silhouette coefficient for a particular data point is given by the following equation:

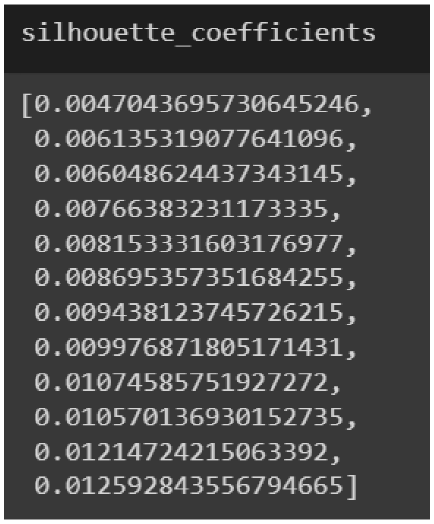
$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where,

- S(i) is the silhouette coefficient of the data point i.
- a(i) is the average distance between i and all the other data points in the cluster to which i belongs.
- b(i) is the average distance from i to all clusters to which i does not belong.

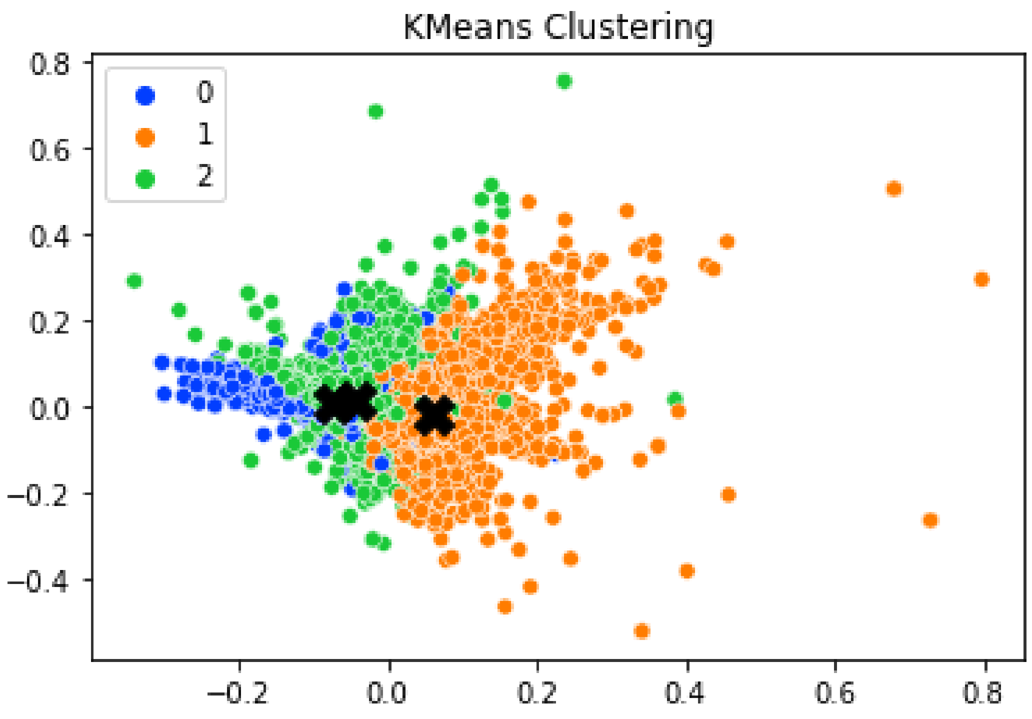
The silhouette coefficient has a value between [-1, 1]. A score of 1 indicates that the data point i is very compact within the cluster to which it belongs and is located far away from the other clusters. The worst possible value is -1. Near-zero values indicate overlapping clusters.

Silhouette coefficients are:



```
silhouette_coefficients

[0.004704369573064524б,
 0.006135319077641096,
 0.006048624437343145,
 0.00766383231173335,
 0.008153331603176977,
 0.008695357351684255,
 0.009438123745726215,
 0.009976871805171431,
 0.01074585751927272,
 0.010570136930152735,
 0.01214724215063392,
 0.012592843556794665]
```

Results after computing cluster analysis for k=3, we get the following scatterplot:



RESULTS

As mentioned above, we used two methods for vectorization, Word2Vec and TF-IDF. The following table summarizes the accuracy score of each model present in both vectorization methods.

Models	**Word2Vec**	**TF-IDF**
Logistic Regression	57%	54%
Decision Tree	46%	56%
Random Forest	57%	62%
SVM	58%	58%
Naive Bayes	52%	53%
KNN	54%	51%

In order to decide which method and particular model to choose, we used the models' precision, recall and F1 scores apart from just relying on their respective accuracy scores.

Word2Vec:

	Precision Score	**Recall Score**	**F-score**
Logistic Regression	0.57	0.57	0.57
Decision Tree	0.46	0.46	0.46
Random Forest	0.59	0.57	0.55
SVM	0.58	0.58	0.57

TF-IDF:

	Precision Score	**Recall Score**	**F-score**
Logistic Regression	0.54	0.54	0.54
Decision Tree	0.56	0.56	0.56
Random Forest	0.62	0.62	0.61
SVM	0.62	0.58	0.53
Naive Bayes	0.59	0.53	0.47
KNN	0.54	0.51	0.43

CONCLUSION

Although there is not much difference between the two methods, we choose TF-IDF as our final vectorisation method as being slightly better than Word2Vec based on accuracy scores. Among all the models, Random Forest has the highest accuracy of 62%. Not only it has the highest accuracy but also the highest f-score along with the precision and recall scores. F1 score is a harmonic mean of precision and recall scores. The f-score of Random Forest is 0.61 which makes it better than all other models. Therefore, as per these results and the accuracy score, we can conclude that the Random Forest based on TF-IDF is a better and more accurate model as compared to all other models used in this project.

The accuracy of all models is not high. In the future, we can improve the models by tuning hyperparameters to find the best condition, and incorporating more classification machine learning models, such as bagging, boosting in random forest and so on.

Based on the sentiments expressed in tweets, analyzing the demographics would be helpful. As our research focused on 2012 US Presidential Elections, candidates could analyze the demographics of the people who are showing negative opinion and then start campaigning in those areas from where the majority of negative tweets come from. This is also helpful for any other company or brand in the world. As people openly express their opinions on Twitter, by understanding the demographics of unhappy customers a company can improve their service in that area. Moreover, it could also help in measuring change in people’s sentiments or attitudes towards a certain brand before and after a new marketing campaign.

REFERENCES

[1] Alec Go, Richa Bhayani and Lei Huang, Twitter Sentiment Classification using Distant Supervision
[2] B. Pang and L. Lee. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2):1–135, 2008.
[3] Vishal A. Kharde and S.S. Sonawane. Sentiment Analysis of Twitter Data: A Survey of Techniques, International Journal of Computer Applications (0975 – 8887), 2008.