COMPARISON OF TWITTER SENTIMENTAL ANALYSIS USING VARIOUS CLASSIFIER

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

by

Shashank Shekhar (161287)

Under the supervision
of

Dr. Aman Sharma

(Assistant Professor)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234

Himachal Pradesh

TABLE OF CONTENTS

ABBREVATIONS

SVM	Support vector machine
ML	Machine learning
NLP	Natural language processing
NB.	Naive Bayes
API	application program interface
TDM	Term Document Matrix

TABLES

LIST OF FIGURES

Description Page No.

Candidate's Declaration

I hereby declare that the work presented in this report entitled “Comparison of twitter sentimental analysis using various classifier**”** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2019 to June 2020 under the supervision of Dr. Aman Sharma, Computer Science/ IT.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shashank Shekhar

Shashank Shekhar(161287)

This is to certify that above statement made by the candidate is true to the best of my knowledge.

[illegible]

Dr. Aman Sharma

Computer Science/Information Technology

Dated: 01/06/2020

ACKNOWLEDGMENT

Working for the “Sentiment Analysis” project was interesting. We got to learn about Long Short Term Memory (LSTM) in Machine Learning, and to compare the accuracies and outcomes of each of these.

Special thanks to our supervisor Dr. Aman Sharma for his guidance and advice on this project.

Also, we are very grateful to our college, lectures, and friends where they gave us enough time to complete this report and we would like to thank all other people who supported us in our project.

Thank you.

ABSTRACT

The Social Media Platform has emerged recently as the most powerful platforms for people to give their opinion and influence of almost each and every part of everybody’s day to day life. These opinions of people matters lot to analyze how propagation of the information impacts this world around us in the socials media platforms like twitter and facebook. Sentiment analysis of the tweets and facebook comments help us in understanding the polarities and the inclination of people because of a specific agenda, special topics, a product or any topic concerning them. The areas where this sentiment analysis is widely used are public elections, promotion of a movie, and analysis of a brand new product. In this project we have used the different online platforms to extract the available live tweet and comments and perform sentiment analysis.

The aim basically is to analyze the sentimental score in the noisy twitter rest and facebook comments. This paper gives reports on the sentimental analysis of extracting huge number of the tweet and facebook comments. Final results classify comments and tweets into positive and negative towards that particular topic. Also, we discussed the different possible techniques to carry out sentimental analysis on the extracted datum.

In the implemented system, texts from social media are collected and opinion mining is done on these texts. According to the outcome of the sentiment analysis on these pieces of text various suggestions can be made to the user. Taking an example, opinion mining can be done on dataset present for the patients and consumer’s opinions on the various clinical treatment and medicines. These outcomes gives information for the use of various hospitals, pharmaceutical industry.

Chapter 1

INTRODUCTION

INTRODUCTION

In the fast growing world of the internet social online media platforms like instagram, twitter and facebook, etc. dominate in spreading news or information of any kind throughout the world at a very rapid speed. A topic only becomes the 'hot topic' or the trending one if more and more people are sharing their views on that topic through tweets or comments. These topics generally tend intended to spread awareness or to promote a brand or political leader or party. It is also widely used in the field of entertainment. Big organization uses the people's feedback to improve their products and services. Political parties form their election strategies based on people's feedback.

One such example can be the incident of leaking of pictures of iPhone 11 to create hype in the market and attract the customers. Thus there is this large potential of discovering & using the immense pattern available through various social media platforms and using that data for their benefit in all kind of fields. Sentiment analysis or the opinion mining are basically the process that is used to identify the polarity of a particular comment from social media platforms. It makes uses of the natural language processing, text .analysis', computational .linguistics., and the biometrics to the identify, quantify' and to the study subjectives information.

Through the sentiment anlysis' we predict the emotions of the person writing the particular piece of text. It basically intends to understand the polarity of user towards or against the particular concerning agenda of the user.Most of the people nowadays uses social media' platform so as to stay updated with current affairs and also news. Sites like twitter, facebook; instagram offers a platform to these people to raise their opinions about different agendas.

Design:

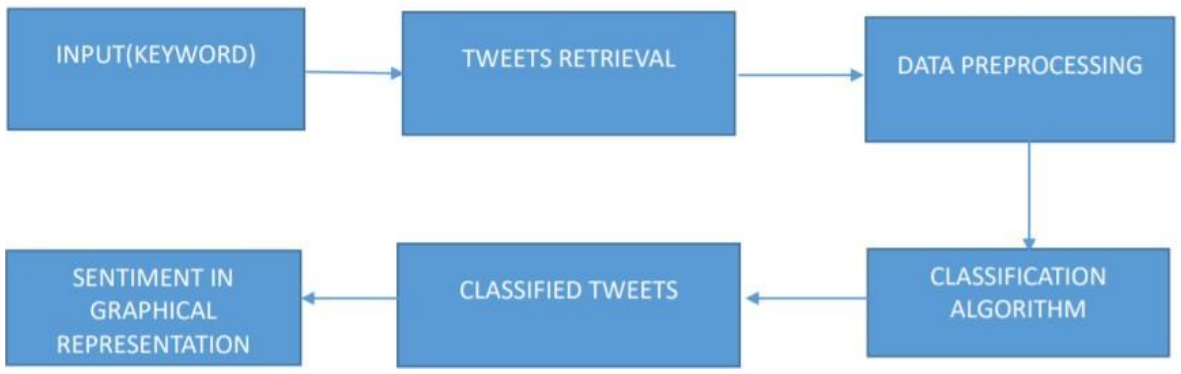


Fig.1 Flow of the twitter behavioural analysis

For example if we watch a TV series, after that we can start a discussion about the acting skills, plot, etc of the series online.

The implmented framework', tweet are too collecteed and the nostalgic examination is the additionally performed on the tweets. Bases on those consequence of the nostalgic examination not many recommendation can be given to the clients. The given executed frameworks can likewise perform wistful examination's on the information accessible for the patients and customer's feeling on different the medications and the medication. These outcomes can likewise give the forward-thinking data' for the emergency clinics, pharmaceutical Industries, and the clinical staff, on the adequacy/incapability of the given treatment. Along these lines, the actualized framework can help in improving the medicinal services of the focused on diabetic patients.

This piece of Information forms a basis for people to evaluate, rate about the performance of Not only any TV series but about various products and to gain information about if it is going to be a success or not. This type of vast knowledge on these sites can used for marketing, advertising and social studies . Therefore, opinion mining has wide range of applications and includes emotion mining, polarity, etc.

Contrarily, analyzing the tweets posted by users is not at all a piece of cake. A number of different challenges are involved in terms of grammar, lexicon and polarity of the tweets posted. They usually are non-grammatical and unstructured. It is quite difficult to interpret and understand their meaning. Also using slang words, not one of the known vocabulary words and acronyms are quite common when posting online . Categorizing these words per polarity usually gets difficult for the natural processors that are involved. *This project of ours makes use of Apache Spark's quick processing competency to analyze sentiments from high velocity real-time tweets.

Next platform that we have used to perform sentiment analysis is facebook. Facebook is a social media platform in which user posts comments, posting pictures and posting and various links to news or all the other content on the world wide' web, video call, chat live, and watch videos. The NLP-based opinion mining has centred on Twitter for product/service reviews, but it is possible to more accurately classify as well as identify the emotion of the user in Facebook status messages due to their nature. Facebook status messages are better than reviews, and also are easier to classify as compared to tweets due to their ability to contain more allowed characters and thereby giving more definite and accurate portrayal of the emotions of the user posting it.

Why should we perform opinion mining on Facebook status updates?

According to an article posted on January 2010 on the site - InsideFacebook.com,

Users usually spent nearly 7 hours a day on Facebook as of December 2009, which is quite on top other than the first 10 companies other than that of facebook on the Internet. From the point of view of marketing, trying to understand the user sentiment as it connects to the topic of interest. If a user is posting something positive or negative about say a health care reform, political party advertisements may appear to be sympathetic to a user's viewpoint. Also, a user may tend to update status might as well show advertisement for a local music club.

PROBLEM STATEMENT

Problem statement in sentimental analysis includes

- classifying the concerned. Polarity of the given extracted line from a particular document, sentence, or the aspect ratio.
- Whether our opinion shown in the given text is positive, negative or neutral.

Online networking site, Twitter is majorly driven by tweets which are having limit of 140 character message. Therefore, the limit on number of characters forces us to use the hashtags for the classification of the text selected. In the past few years nearly 7000 extracted tweets are usually posted in one second that results in approx 561.6 million tweets per day. The obvious streams of these tweets are usually noisy reflecting the multi topics and usually changing attitudes information in a format that is not filtered as well as not structured properly. Twitter sentiment

Analysis uses natural language processing in extracting, identifying in order to characterize the sentiment content. Sentiment Analysis is usually done at the basic two levels:

1) Coarse level
2) Fine level.

The coarse level involves analyzing entire documents, while the fine level only includes analyzing attributes.

There are two types of sensations present in the text:

1. Direct
2. Comparative.

In comparative emotions, objects are compared in the same sentence whereas in the second case, basically objects are not dependent on each other in the same sentence.

OBJECTIVE

The objective of research work are:

1. Data extraction
2. Preprocessing the extracted data
3. Testing and analysis of proposed methodology

METHODOLOGY

For objective 1:

1. Getting twitter API
2. Connecting to Twitter's REST API and downloading data
3. Targeting Relevant Tweets
4. Preprocessing of data
5. Tweet Mining

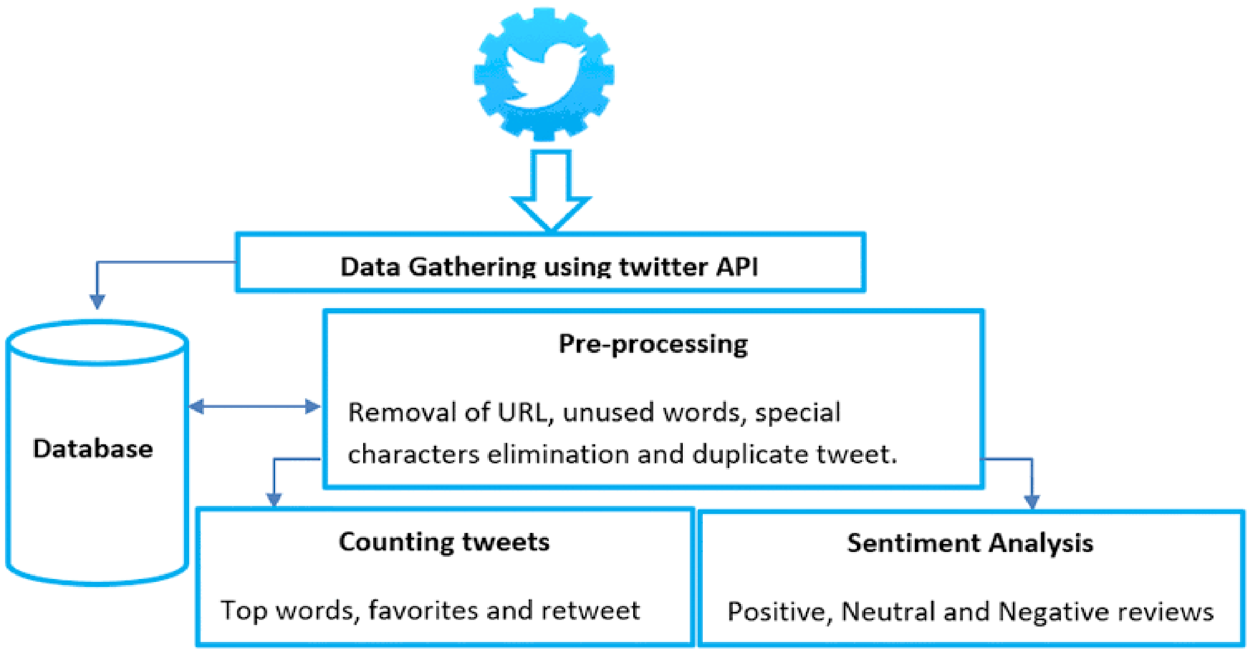

Data Gathering using twitter API
Database
Pre-processing
Removal of URL, unused words, special characters elimination and duplicate tweet.
Counting tweets
Top words, favorites and retweet
Sentiment Analysis
Positive, Neutral and Negative reviews

Fig. 2 Data gathering using twitter API

Research methodology for objective 2:

1. Authorize the twitter API clients
2. Make the get request to the twitter API to the fetch tweets for the particular query
3. Parsing the tweets. Classify each tweets as the positive or negative or neutral.

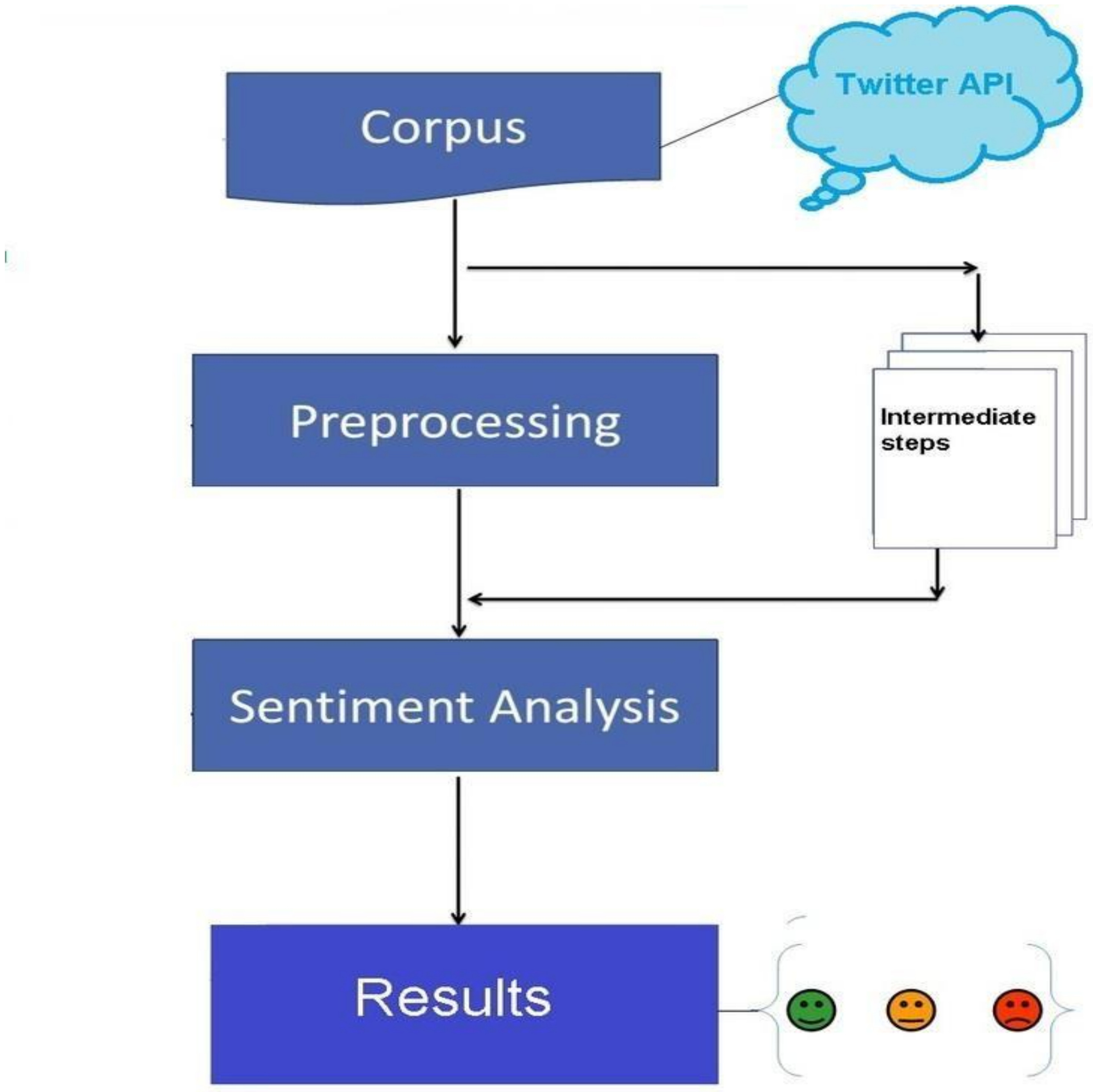


Fig. 3 Extracting twitter data

Research methodology for objective 3:

1. Fitting the Predictive Model such as the naïve Bayes classifiers and the random Forest classifier.

2. Comparing result by both the model on the twitter data set.

Chapter 2

LITERATURE SURVEY

Limitation Of Prior Art

This field of research is relatively new so there is still room for further modification and research. A good amount of previous work on this subject has been done previously by a lot of researchers. The biggest limitation in tweets' cogent analysis is the character limit of 140 which forces one to express very brief textual opinions.

The best result emotion classification is reached using the supervising learning technique –

1. Naive Bayes
2. Random Forest Classifier

But in the case of a supervised learning approach, the required manual labeling is quite expensive.

There is little work on meager techniques and semi-supervised approaches, but there is still room for improvement.

Specialists testing new highlights and grouping procedures analyze their outcomes.

Gauge execution. There ought to be an appropriate and formal approach to analyze between results. To secure through different highlights and grouping methods

The best of highlights and most effective characterization procedures for unique applications.

RELATED WORK

The model –

Bag-of-words is highlight model that id broadly utilized for order of all the content because of its effortlessness and great execution coming connected at the hip. As per this model content is to Can be named a sack or assortment of individual words, in which no words have any reliance with the other, this implies it totally disregards the language structure and request in which the words would have been orchestrated inside the content Are. This model is well known with assumption investigation and has been utilized by different analysts.

This part summarizes various scholarly and research works in the field of ML and data mining to analyze user sentiments on Twitter and Facebook and design predictive models for various applications.

Social media platform that are available to us are increasing everyday in number, the info shared on them has also started to grow and this information can be used to turn into strategies for promotion, social drives, promotional and other business purposes. Social media benefits to understand mass opinions and uproot the emotions are considered and enlightened the fact how the social media platform as twitter provides the advantage at the time of elections. Along with this, hashtag concept is in use for classification of text as few words are used by it to convey emotions. Various researchers have suggested that previous research work lacked better training set and thereby missed few imp features of the target data. They choose a two staged approach as per the framework- first step was to prepare the training data from the site twitter using the method mining and to convey relevant characteristics and after that putting forward the Supervised Learning Model in order to predict outcomes of the elections in USA in the year 2016. Right after gathering and processing the data, training data set was made at first by labeling of the hashtags manually and making clusters, and after that making use of sentimental analyzer – VADER. It gives output of polarity in percentage.

This approach reduced the training data set and they then used the SVM - support vector machine and the NB - nave base algorithm to classify which is used to detect the positive or negative bias of the dataset. An approach called multistage classification was used where

The entity classifier is given a basic set of tweets and classifies it in relation to each candidate to compare them. The meter they used to determine the eventual winner was the "PvT - Ratio", which counts for a positive number of tweets for the candidate.

Opinion mining by research workers Imran et al.[6] learned about the technology 'Apache Spark' and used it for streaming of the tweets fast and studied the approach called Stream Sensing in order to handle the real-time data set noisy form. The approach was conducted on twitter data set to find some important and usable trends that further can be categorized into any real-time written material stream. Unsupervised-learning approach is here used to locate different kinds of patterns and trends from tweets that are worked upon on Apache Spark. Also using the approach by Zhu et al.[1] and Li et al.[2] used for data mining by choosing time window, these authors choose sliding window type for seizing the tweets live. The usual basic approach used in all the worthy research works is collecting data using social networking site twitter.

API-application program interface, preprocessing the data, data filtering and then approaching in extraction of features, categorization and analysis of different patterns makes the difference. Researchers used the sliding window - 5 minutes when collecting data and after that they created Term Document Matrix (TDM) for extraction of the features. The analysis of patterns was carried out using scores of TF-IDF for finding quite imp keywords as elaborated by Wu et al. The buzz topic or the hashtag is given in and data similar to it are processed to make the TDM and calculating the weights of TF-IDF to look for the quite important words is usually the basic idea behind sentiment analysis. Parallel computation of TDM, TF-IDF marks and determining 5 keywords generated from TDM every minute likewise the sliding window moves is one of the major feature of this work. Therefore, this work leverages fast and accurate computation work of the Apache Spark. In one more research work of opinion mining on Twitter, facebook researchers also found out the polarity –

Positive, negative or neutral of tweets by making a classifier. They also made use of multiple algorithms and methods to find out the effect of active entity on the tweet pattern showing particular emotions. Tweets were mined at entity level only i.e. brand, product, celebrity elements in tweets in place of the whole sentence in the updates posted by users. Approach followed by them made use of algorithms to mine features and to keep track of the impact and influence made by their work. Construction of n grams was included in feature extraction process that is done after preprocessing. It also included improving accuracy as well as taking care of negative part in the classification.

These researchers opted for People Rank Algorithm for measuring influence and further analysis in the field. Basic idea lying behind this algorithm is – more people rank's value more central the node is in the graph, further more importance on twitter.

One more algorithm is – Twitter rank algorithm, it is an extension to the previous page rank algorithm in determining influence of users by considering common points between structure of nodes and the users. Shortcomings of the page rank algorithm were addressed by developing this approach. The idea that influential people are following you is used in measuring the influence.

Few mathematical calculations of ratio of followers to following, hashtags, retweets were used to determine few weights and out of it finally deriving mathematical

Following some mathematical computation of ratio of followers/following, retweets, mentions like parameters, they determine the influence of basic few entities.A modified approach to deal with the subjective and the objective sentences was proposed by J.S et el[2] to find opinions from the sentences. Following steps were followed by them in their approach:
1. Classification of sentences as public opinion and non-opinion.
2. Classifying objective sentences as objective or subjective.
3. Classifying subjective sentences as positive, negative or neutral
4. Classifying objective sentences as positive, negative, or neutral

This procedure provided sentiment and context orientation.

Borade A. J. et al.[3] was next to use techniques of review mining for making product ranking system. Ranking to different kinds of products were given according to user review collected.

Issues that are considered while determining scores of the product:

1. Product review
2. Popularity of product
3. Month of product release

Large scale distributed system was outlined by Khuc et al. for sentiment analysis in real time. Components present in the system of Khuc et al.[4]:

1. a sentiment classifier
2. lexicon builder

A distributed different database and map- reduce framework are used to implement above mentioned components.

To connect sentiment lexicon with machine learning algorithm a method was introduced and after that accuracy was observed. Manoj Danthala made suggestion about an idea to analyse tweets. Big data are handled by using Apache Hadoop. For predictive analytics above mentioned analysis can be used, text analytics as well as sentiment analysis. Mrs. Mahia Goyal used dictionary called Sentiwordnet for calculating the user’s sentiments. They used unsupervised approach based on dictionary in their work. Their field of research was limited only to domain of tourism.

The following fragment of the previously mentioned code manages getting the content of tweets dependent on the IDs. We circle through the tweets in corpus, calling the API on each tweet to get the Tweet. Status objects of the specific tweet. A while later, we utilize that equivalent article (status) to get the content related with it and push it into the preparation Data Set at that point rest (for example stop execution) for five minutes (900/180 seconds) so as to comply with as far as possible.

Chapter 3

SYSTEM DEVELOPMENT

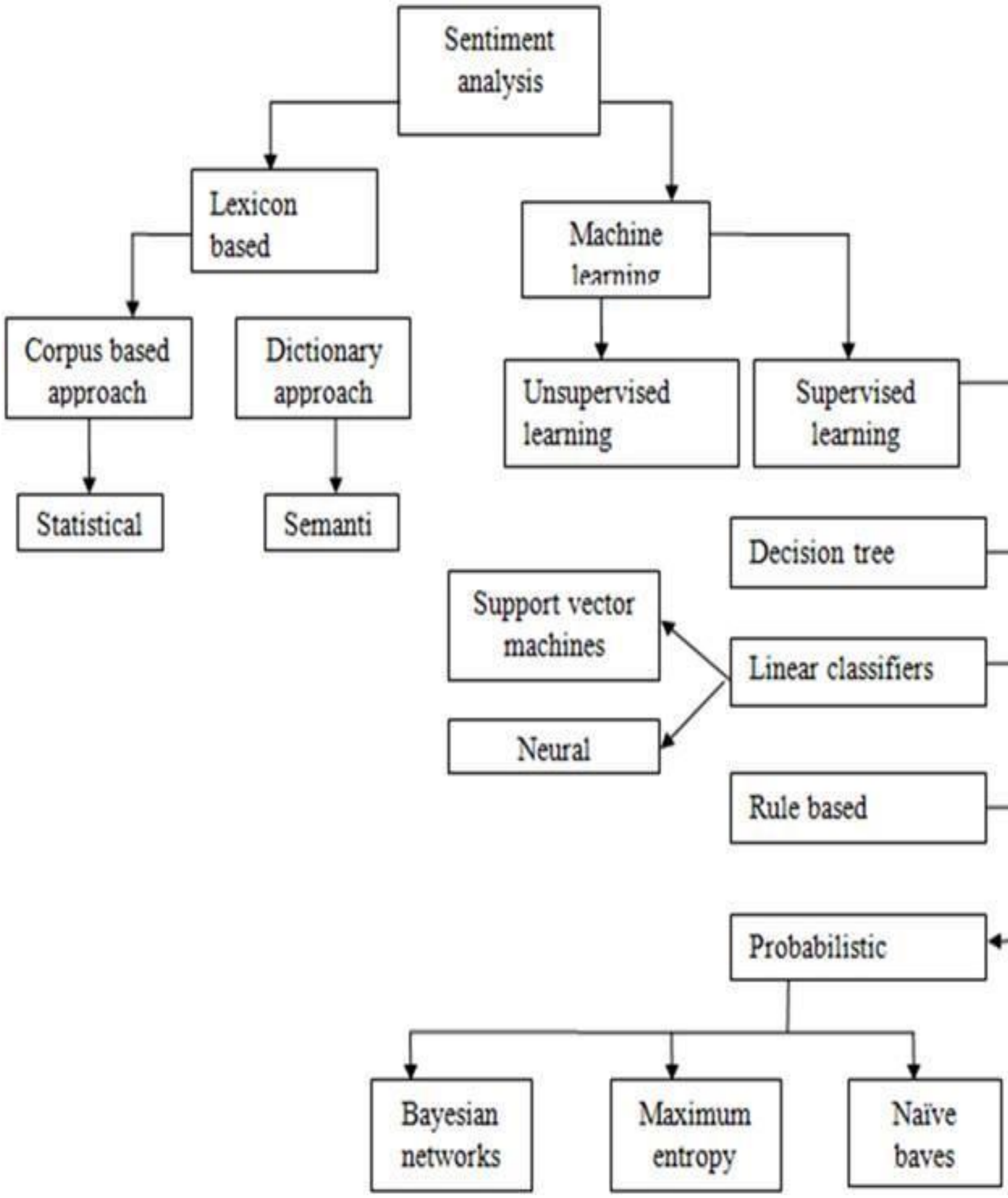


Fig.4 System development

Design a functional classifier for mining according to the following steps:

1. Data acquisition
2. Human labelling
3. Feature extraction
4. Classification
5. TweetMood Web Application

DATA COLLECTION

The data set in the form of raw tweets is obtained using the python library -

"Tweestream"

This library provides a simple package for the Twitter streaming API. This Twitter streaming API allows two ways to get the handle of tweets:

1. Sample Stream

2. Filter the data

The sample stream saves only a small, random sample of tweet streaming in real time. The stream that filters the filter provides tweets that match certain criteria. It delivers tweets according to 3 criteria:

1. Specific keywords to search for in tweets

2. Specific Twitter users (users) according to their user-id

3. Tweets that originate from a specific location (s) (only for geo-tagged tweets).

Any of the above mentioned filtering criteria can be specified by a programmer or various combinations of these. We do not have such restrictions for our purpose and therefore we will stick with this mode - sample stream.

To increase the comprehensiveness of the data collected, instead of collecting the data at once, we acquired it in small portions at different times. If we collect data at once, the prevalence of tweets may be hurt because a large number of tweets will have the same sentiment or mood as they would be referring to a trending topic. This incident was observed while we were going through a sample of acquired tweets. Natural Language Processing (NLP) is very important area of research these days in data science and its important application is sentiment analysis. From Normal Language Processing (NLP) is significant territory of research nowadays in information science and its significant application is opinion examination. From breaking down surveying to making whole promoting methodologies, this has helped a ton in reshaping organizations work; this is a significant territory to be acquainted with. Rather than various individuals collaborating to physically finish the investigation of estimation of focused part of populace this work should be possible in seconds through this method.

Following advances are followed so as to finish feeling examination of twitter information:

1. Understanding the given explanation of the issue
2. Processing and cleaning of the tweets
3. Generation of story and perception from the removed tweets
4. Extraction of the necessary highlights from cleaned tweets
5. Model structure of the assessment investigation

<u>1. Setting up THE TEST SET:</u>

Our assignment of dissecting conclusions is profoundly centered around printed information, normally we expect a ton of content handling there and this is right as well. Test and Training, both will involve message as it were.

Beginning with the Test set so as to prepare for the Training set extraction part, as it depends more on the API. Following are the means we are to pursue:

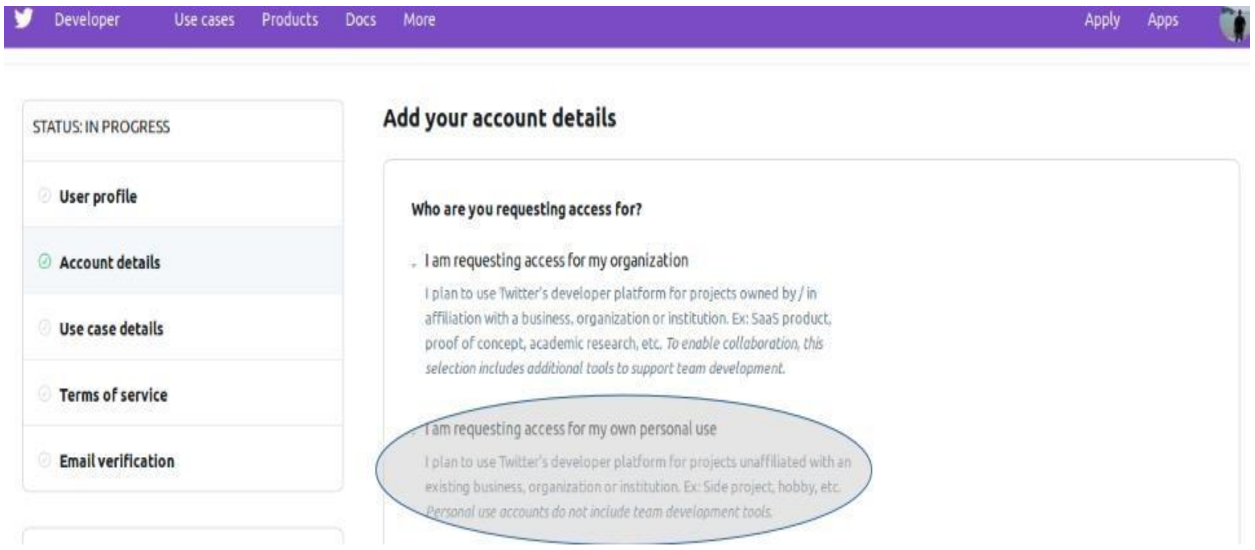
A. Registering Twitter application so as to get our own accreditations.
B. Authenticating the Python content with API utilizing the accreditations.
C. Creating capacity to download the tweets as indicated by a hunt catchphrase.

As registering an application with twitter is the only way to get authentication credentials, it is a critical task. We start with the code as soon as the credentials are done. Test set lies in the above mentioned step 3.

<u>2.Getting the authentication credentials:</u>

Visiting the Twitter Developer website and logging into our own account through the following link:

developer.twitter.com



Add your account details

Who are you requesting access for?

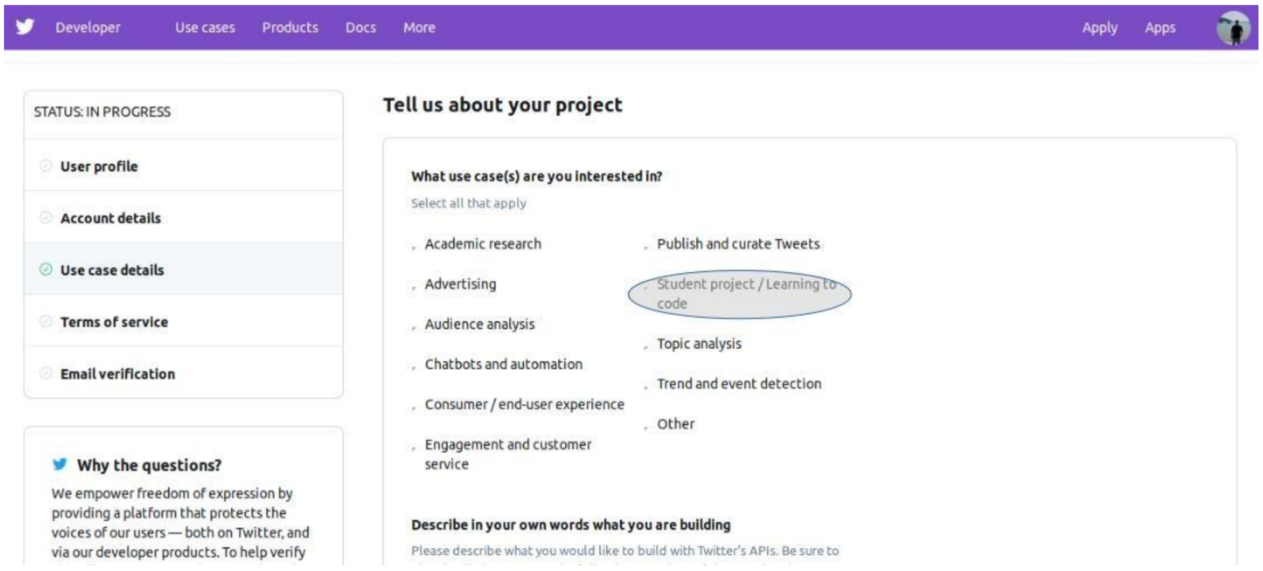
I am requesting access for my organization

I plan to use Twitter's developer platform for projects owned by / in affiliation with a business, organization or institution. Ex: SaaS product, proof of concept, academic research, etc. *To enable collaboration, this selection includes additional tools to support team development.*

I am requesting access for my own personal use

I plan to use Twitter's developer platform for projects unaffiliated with an existing business, organization or institution. Ex: Side project, hobby, etc. *Personal use accounts do not include team development tools.*

Fig 3.1 Getting twitter authentication details



STATUS: IN PROGRESS

- User profile
- Account details
- Use case details
- Terms of service
- Email verification

Why the questions?

We empower freedom of expression by providing a platform that protects the voices of our users — both on Twitter, and via our developer products. To help verify

Tell us about your project

What use case(s) are you interested in?

Select all that apply

- Academic research
- Advertising
- Audience analysis
- Chatbots and automation
- Consumer / end-user experience
- Engagement and customer service
- Publish and curate Tweets
- Student project / Learning to code
- Topic analysis
- Trend and event detection
- Other

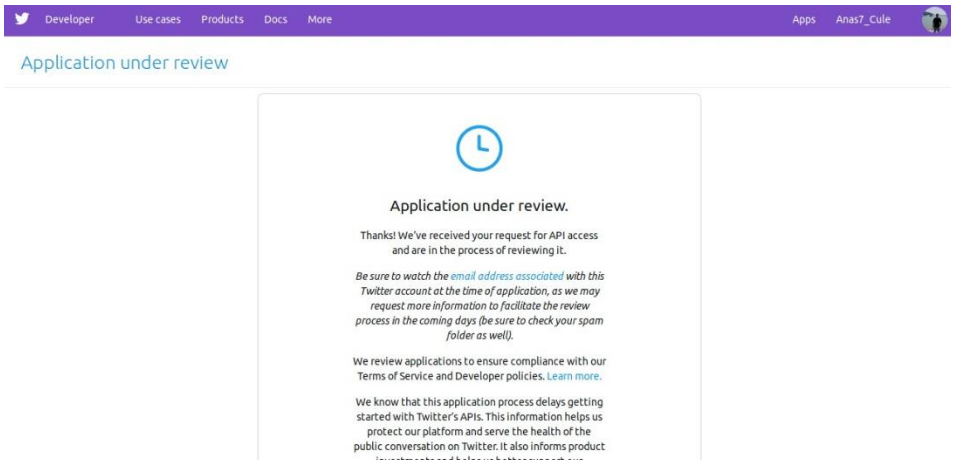
Describe in your own words what you are building

Please describe what you would like to build with Twitter's APIs. Be sure to

Fig.3.2 Step 2 in authentication

After making the choices, scrolling-down and filling the paragraphs of the use cases that is the required. Reading the Term and Conditions list, agreeing to the them and then the Submitting Application.

Finally, we get a message similar to the following:



Application under review

Application under review.

Thanks! We've received your request for API access and are in the process of reviewing it.

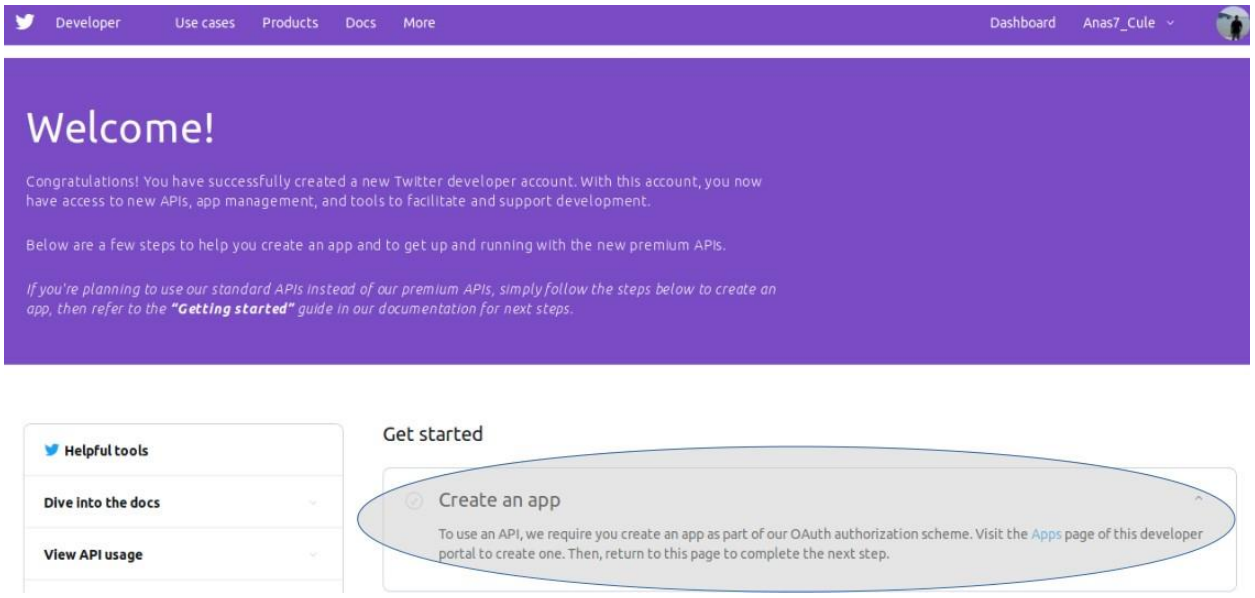
Be sure to watch the email address associated with this Twitter account at the time of application, as we may request more information to facilitate the review process in the coming days (be sure to check your spam folder as well).

We review applications to ensure compliance with our Terms of Service and Developer policies. Learn more.

We know that this application process delays getting started with Twitter's APIs. This information helps us protect our platform and serve the health of the public conversation on Twitter. It also informs product

Fig 3.3 Step three in authentication process

After getting approval from twitter, we are given thes link that redirect us to the page where we select creating the app tab.



Welcome!

Congratulations! You have successfully created a new Twitter developer account. With this account, you now have access to new APIs, app management, and tools to facilitate and support development.

Below are a few steps to help you create an app and to get up and running with the new premium APIs.

*If you're planning to use our standard APIs instead of our premium APIs, simply follow the steps below to create an app, then refer to the **"Getting started"** guide in our documentation for next steps.*

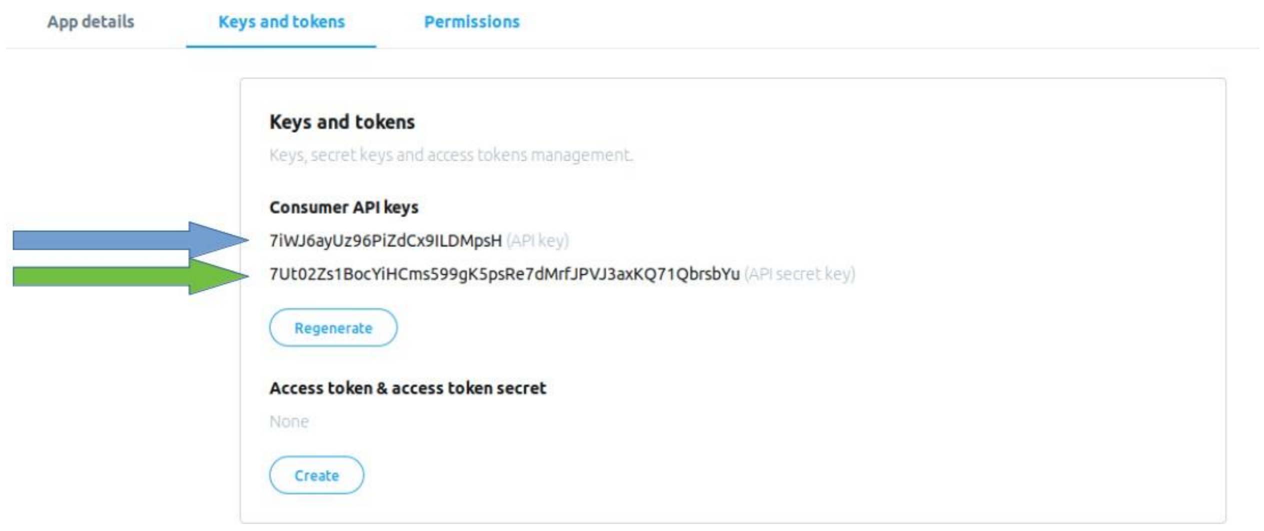
Create an app

To use an API, we require you create an app as part of our OAuth authorization scheme. Visit the Apps page of this developer portal to create one. Then, return to this page to complete the next step.

Fig 3.4 Page after getting approval

The following page that opens up incorporates the application subtleties that we simply input, get to tokens and authorizations. Snap the "Keys and token" tab.

We need to duplicate API key and the API mystery key into a sheltered spot.



Keys and tokens

Keys, secret keys and access tokens management.

Consumer API keys

7iWJ6ayUz96PiZdCx9ILDMpsH (API key)

7Ut02Zs1BocYiHCms599gK5psRe7dMrfJPVJ3axKQ71QbrsbYu (API secret key)

Access token & access token secret

None

Fig 3.5 Duplicate API keys

In order to generate Access tokens credentials click onto creates. Next step is to the copy the Access tokens and the Access token secrets into the safe place as well. Finally we have acquired the credential.

3. Authenticating the python script:

Now that we have the API keys and tokens as our logins credentials, we proceed to the authenticating our programs. We need Twitter librarys, then creates twitter API objects with credentials from the safe place.

```python
import twitter

# initialize api instance
twitter_api = twitter.Api(consumer_key='YOUR_CONSUMER_KEY',
                          consumer_secret='YOUR_CONSUMER_SECRET',
                          access_token_key='YOUR_ACCESS_TOKEN_KEY',
                          access_token_secret='YOUR_ACCESS_TOKEN_SECRET')

# test authentication
print(twitter_api.VerifyCredentials())
```

Running the above codes we get the following JSON responses:

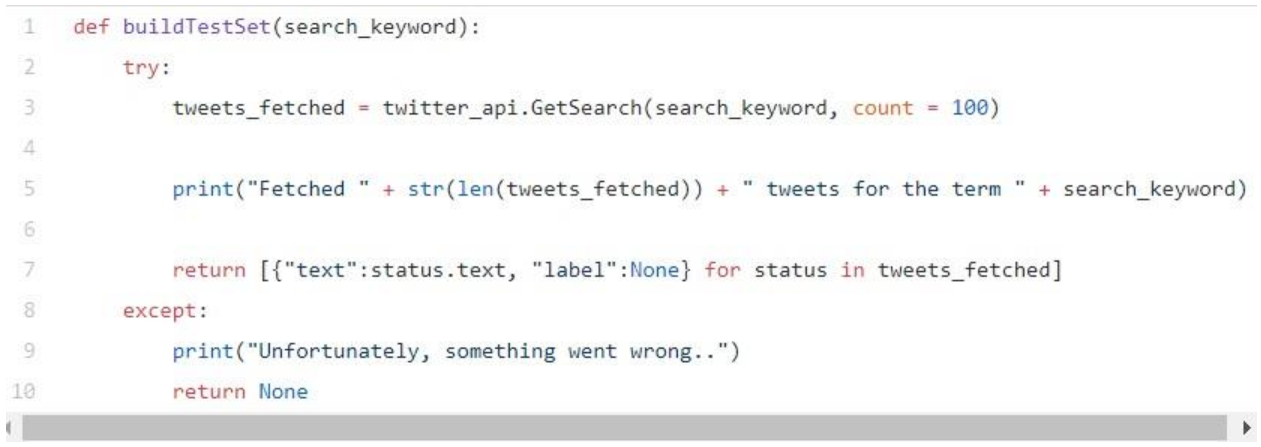
```
{"created_at": "Tue Feb 12 17:48:27 +0800 2019" 'default_profile":
true ............}
```

Fig 3.6 Code snippets 1

4.Making the capacity to manufacture test set:

Presently we make the capacity that downloads the Test we will use in our model. This will be the capacity that will take an inquiry catchphrase (for example string) as info, looks for tweets that incorporate the catchphrase referenced above and returns them as twitter. Status protests that we can circle through.

We just get up to 180 tweets utilizing our pursuit work that is referenced above on at regular intervals. Our Training set won't be enormous. Our capacity for the looking for the tweets (for example Test set will be:

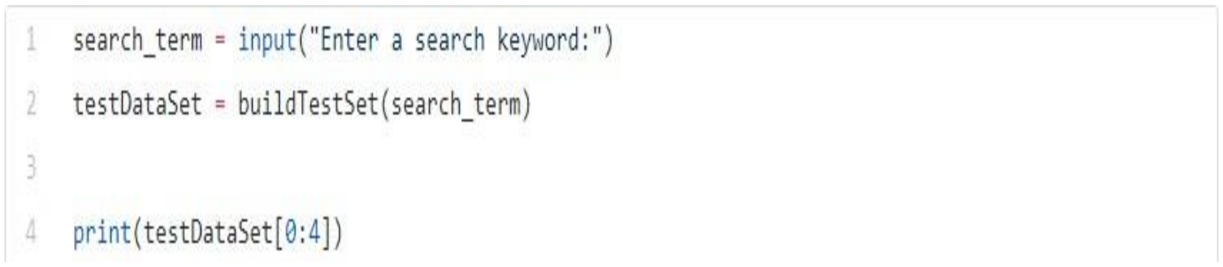


```
def buildTestSet(search_keyword):
    try:
        tweets_fetched = twitter_api.GetSearch(search_keyword, count = 100)

        print("Fetched " + str(len(tweets_fetched)) + " tweets for the term " + search_keyword)

        return [{"text":status.text, "label":None} for status in tweets_fetched]
    except:
        print("Unfortunately, something went wrong..")
        return None
```

Fig 3.10 Code snippets

This function is used to return the sets of tweets that contain the searched keywords.

We add the following codes to function's body before doing testing:



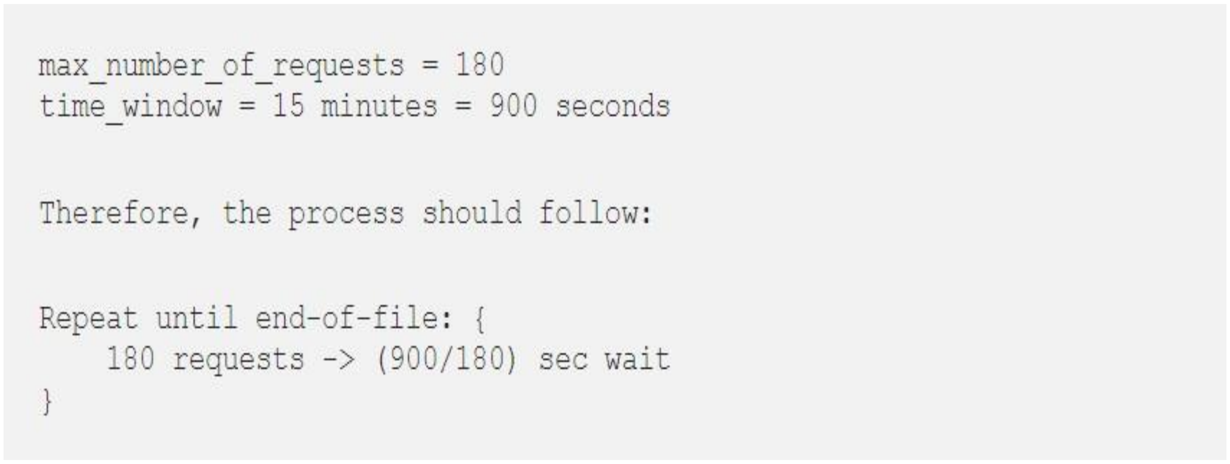
```
search_term = input("Enter a search keyword:")
testDataSet = buildTestSet(search_term)

print(testDataSet[0:4])
```

Fig 3.7 Code snippet 2

This should have print out of the five tweet that contains our search keyword on the Terminal of your IDE's. Now we have everything thing we required for the time being. We have our own Test set and we could move on to the building our Training set.

5. Preparing the data sets:

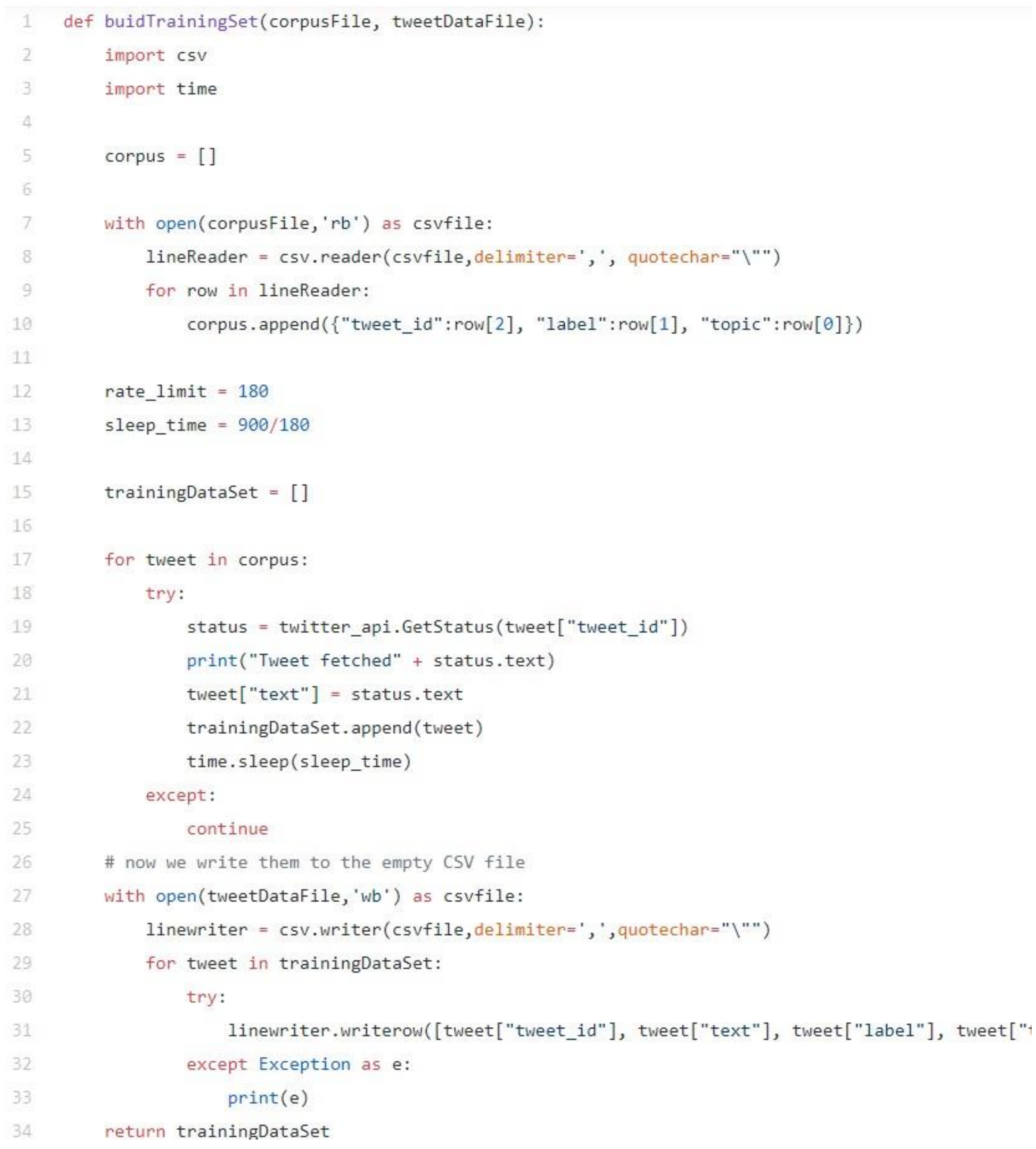
All together for our model to be a triumph we have to have a solid preparing set. We have to mark the information without any irregularities or deficiency. This is all on the grounds that our preparation depends vigorously on exactness of the information. To finish this errand, we will utilize the Niek Sanders' Corpus of more than 5000 hand-characterized tweets, making it very solid. We utilized API to get the genuine tweets message through each tweet's ID number that is remembered for Corpus. So as to download 5000 tweets, we have to pursue:



```
max_number_of_requests = 180
time_window = 15 minutes = 900 seconds


Therefore, the process should follow:


Repeat until end-of-file: {
    180 requests -> (900/180) sec wait
}
```

Fig 3.8 No f requests and time window

After this we have wrote the code that does have exactly what we required. First of all we saved the tweets into the CSV files that we have retrieved through the API’s. This is the function we have used:



```python
def buidTrainingSet(corpusFile, tweetDataFile):
    import csv
    import time

    corpus = []

    with open(corpusFile,'rb') as csvfile:
        lineReader = csv.reader(csvfile,delimiter=',', quotechar="\"")
        for row in lineReader:
            corpus.append({"tweet_id":row[2], "label":row[1], "topic":row[0]})

    rate_limit = 180
    sleep_time = 900/180

    trainingDataSet = []

    for tweet in corpus:
        try:
            status = twitter_api.GetStatus(tweet["tweet_id"])
            print("Tweet fetched" + status.text)
            tweet["text"] = status.text
            trainingDataSet.append(tweet)
            time.sleep(sleep_time)
        except:
            continue
    # now we write them to the empty CSV file
    with open(tweetDataFile,'wb') as csvfile:
        linewriter = csv.writer(csvfile,delimiter=',',quotechar="\"")
        for tweet in trainingDataSet:
            try:
                linewriter.writerow([tweet["tweet_id"], tweet["text"], tweet["label"], tweet["
            except Exception as e:
                print(e)
    return trainingDataSet
```

What we did here was:

First we characterize the capacity to take two sources of info the two of which are record ways:

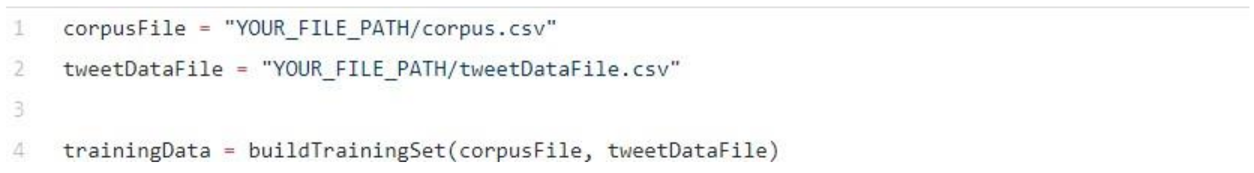
•CorpusFile is the string way to the Niek Sanders' CSV corpus record we downloaded. This document, as referenced prior, incorporates the tweet's theme, mark and id.

•TweetDataFile is the string way to the document we might want to spare the full tweets in. As opposed to corpusFile, this record will incorporate each tweet's content just as theme, mark and id.

Subsequent stage was beginning with an unfilled rundown called corpus.

The following fragment of the previously mentioned code manages getting the content of tweets dependent on the IDs. We circle through the tweets in corpus, calling the API on each tweet to get the Tweet. Status objects of the specific tweet. A while later, we utilize that equivalent article (status) to get the content related with it and push it into the preparation Data Set at that point rest (for example stop execution) for five minutes (900/180 seconds) so as to comply with as far as possible

We utilize the accompanying bit to download the tweets:



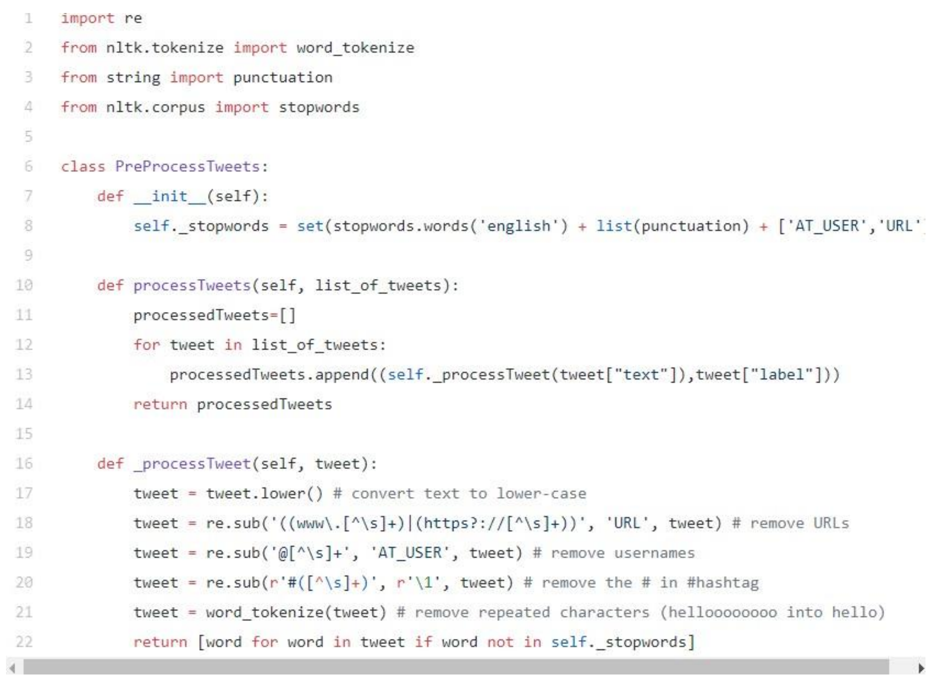
```
corpusFile = "YOUR_FILE_PATH/corpus.csv"
tweetDataFile = "YOUR_FILE_PATH/tweetDataFile.csv"

trainingData = buildTrainingSet(corpusFile, tweetDataFile)
```

6. Preprocessing tweets in the dataset:

This progression is basic and for the most part takes quite a while when building Machine Learning models.

One of the significant pieces of this errand is the words. We most likely can't get the assumption from accentuation. In this way, accentuations doesn't make a difference to Sentimental Analysis. In addition, tweet segments like pictures, recordings, URLs, usernames, emoticons, and so forth don't add to the extremity (regardless of whether it is sure or negative) of the tweet. In any case, this is valid for this application. For example, in another application, you could have a Deep Learning picture classifier that learns and predicts whether this picture the tweet contains represents something positive (for example a rainbow) or negative (for example a tank). With regards to the detail, both Sentiment Analysis and Deep Learning fall under Machine Learning. Indeed, you can perform Sentiment Analysis through Deep Learning, however that is a story for one more day.

The pre-processor class that we have utilized here:



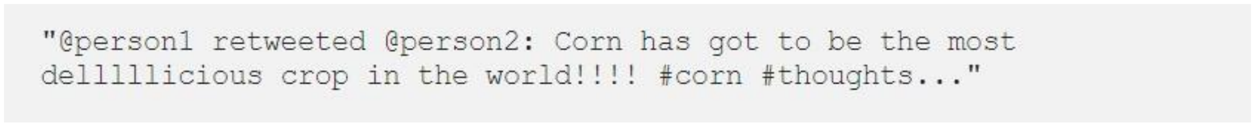
```
import re
from nltk.tokenize import word_tokenize
from string import punctuation
from nltk.corpus import stopwords

class PreProcessTweets:
    def __init__(self):
        self._stopwords = set(stopwords.words('english') + list(punctuation) + ['AT_USER','URL'

    def processTweets(self, list_of_tweets):
        processedTweets=[]
        for tweet in list_of_tweets:
            processedTweets.append((self._processTweet(tweet["text"]),tweet["label"]))
        return processedTweets

    def _processTweet(self, tweet):
        tweet = tweet.lower() # convert text to lower-case
        tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'URL', tweet) # remove URLs
        tweet = re.sub('@[^\s]+', 'AT_USER', tweet) # remove usernames
        tweet = re.sub(r'#([^\s]+)', r'\1', tweet) # remove the # in #hashtag
        tweet = word_tokenize(tweet) # remove repeated characters (helloooooooo into hello)
        return [word for word in tweet if word not in self._stopwords]
```

Fig 3.8 Code snippet

Our first imported library is re, which is a python normal articulation library. This library parses the strings and changes them without having to expressly emphasize through the characters involving the specific string.

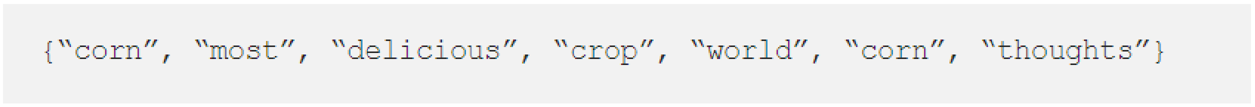
The procedure Tweets work just circles through every one of the tweets contribution to it, calling its neighboring capacity process Tweet on each tweet in the rundown. The last does the real pre-handling by first making all the content in lower-case letters. This is just in light of the fact that, in practically all programming dialects, "Vehicle" isn't deciphered a similar route as "vehicle". In this way, it is smarter to standardize all characters to be lower-case over the entirety of our information. Also, URLs and usernames are expelled from the tweet. This is for the reasons we uncovered before in the article. A short time later, the number sign (for example #) is expelled from each hashtag, so as to maintain a strategic distance from hashtags being prepared in an unexpected way. To wrap things up, copy characters are freed off of, so as to guarantee that no significant word goes natural regardless of whether it is explained in a bizarre manner (for example "caaaaar" becomes "vehicle"). At long last, tweet's content is broken into word (tokenized) so as to the facilitate its handling in up and coming stage.

Preprocessors will give us the tweet like this:



```
"@person1 retweeted @person2: Corn has got to be the most
dellllicious crop in the world!!!! #corn #thoughts..."
```

And after the tokenization:



```
{"corn", "most", "delicious", "crop", "world", "corn", "thoughts"}
```

Fig. 3.9 code snippet

Presently code has evacuated the copy characters. In any case, the copy words are not expelled from the content as they assume a significant job in deciding extremity of the content.

<u>Naive bayes grouping:</u>

Naive Bayes Classifier is a grouping calculation that depends on Bayes' Theorem. This hypothesis gives a method for figuring a sort or likelihood called back likelihood, in which the likelihood of an occasion A happening is dependent on probabilistic known foundation (for example occasion B proof). For instance, on the off chance that Person_X possibly plays tennis when it isn't pouring outside, at that point, as per Bayesian measurements, the probability of Person_X playing tennis when it isn't coming down can be given as

```
P(X plays | no rain) = P(no rain | X plays)*P(x plays)/P(no rain)
```

The baye's theorem:

```
P(A|B) = P(B|A)*P(A)/P(B)
```

Fig 3.9 Bayes theorem

Steps taken further:

1-Build a jargon (rundown of expressions) of the considerable number of words inhabitant in our preparation informational index.

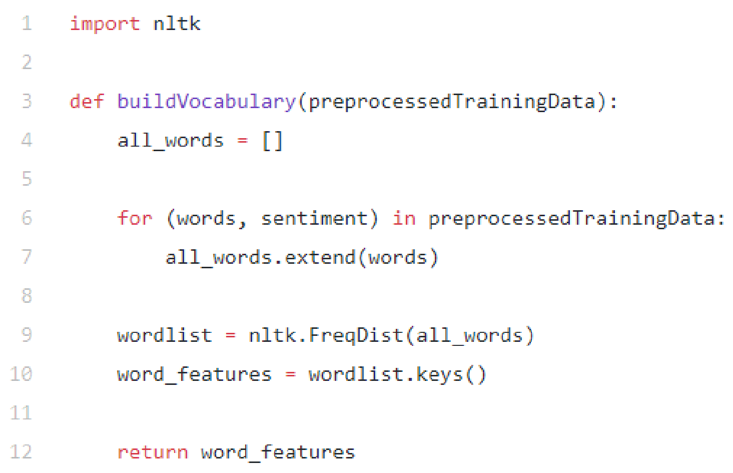
2-Match tweet content against our jargon — word-by-word.

3-Build our pledge include vector.

4-Plug our component vector into the Naive Bayes Classifier

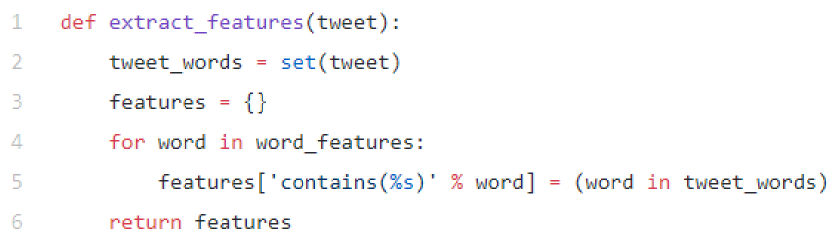
Code snippets of the above mentioned steps are as follows:

Step 1:



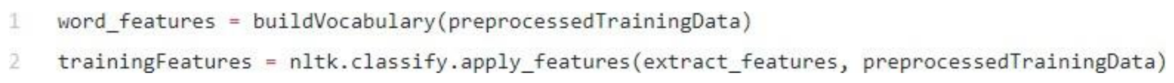
```
import nltk

def buildVocabulary(preprocessedTrainingData):
    all_words = []

    for (words, sentiment) in preprocessedTrainingData:
        all_words.extend(words)

    wordlist = nltk.FreqDist(all_words)
    word_features = wordlist.keys()

    return word_features
```

Step 2:



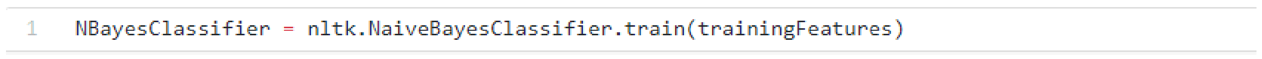
```
def extract_features(tweet):
    tweet_words = set(tweet)
    features = {}
    for word in word_features:
        features['contains(%s)' % word] = (word in tweet_words)
    return features
```

Step 3:



```
word_features = buildVocabulary(preprocessedTrainingData)
trainingFeatures = nltk.classify.apply_features(extract_features, preprocessedTrainingData)
```

Step 4:

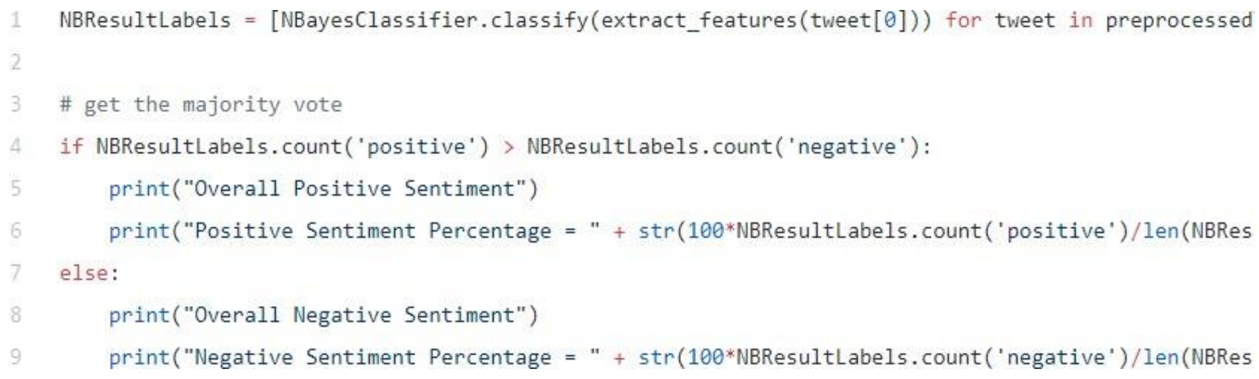


```
NBayesClassifier = nltk.NaiveBayesClassifier.train(trainingFeatures)
```

Fig 3.9 Code snippet

7. Testing the Models:

As per our pursuit term, getting the greater part vote of the marks returned by the classifier, at that point yielding the complete positive or negative rate (for example score) of the tweets.

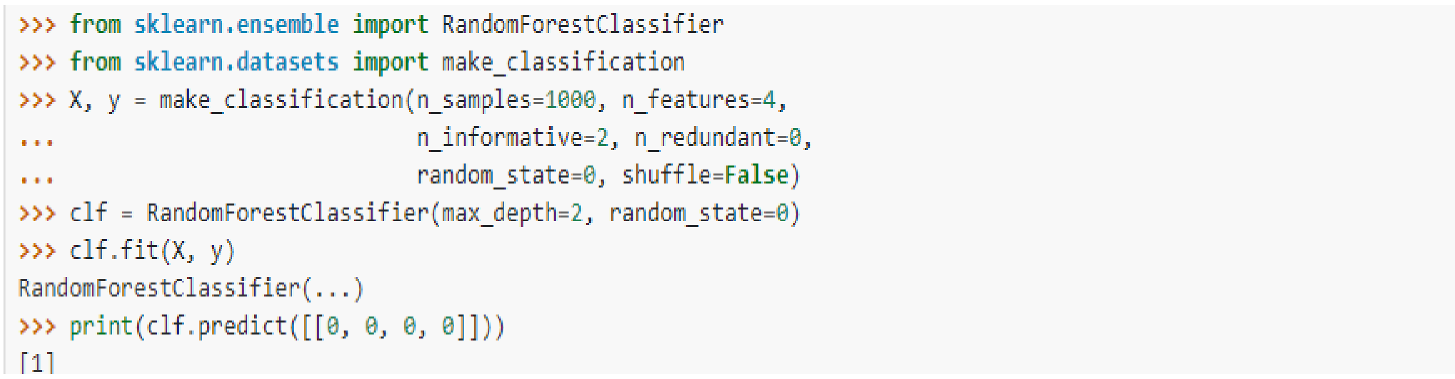


```
NBResultLabels = [NBayesClassifier.classify(extract_features(tweet[0])) for tweet in preprocessed

# get the majority vote
if NBResultLabels.count('positive') > NBResultLabels.count('negative'):
    print("Overall Positive Sentiment")
    print("Positive Sentiment Percentage = " + str(100*NBResultLabels.count('positive')/len(NBRes
else:
    print("Overall Negative Sentiment")
    print("Negative Sentiment Percentage = " + str(100*NBResultLabels.count('negative')/len(NBRes
```

Summing up all, the code snippets, this is the whole python program for twitter sentiment analysis.After doing the sentiment analysis on twitter data, our next step is to extract data from facebook for the same targeted people and perform sentiment anlysis on them and further check for the contrasts in the behaviour of the targeted group of the people.

Random Forest Classifier:

A random forest is a meta estimator that fits the various choice tree classifiers on the different subtests of the extracted datasets and utilizations averaging to improve the prescient exactness and the authority of the over-fitting. The subtest size or length is controlled with the max samples parameter if bootstrap=True(default) , in any case the entire dataset is utilized to the manufacture of the each tree. It is an ensemble algorithms which combine more than of the same type algorithms or the distinct algorithms.



```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.datasets import make_classification
>>> X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(...)
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

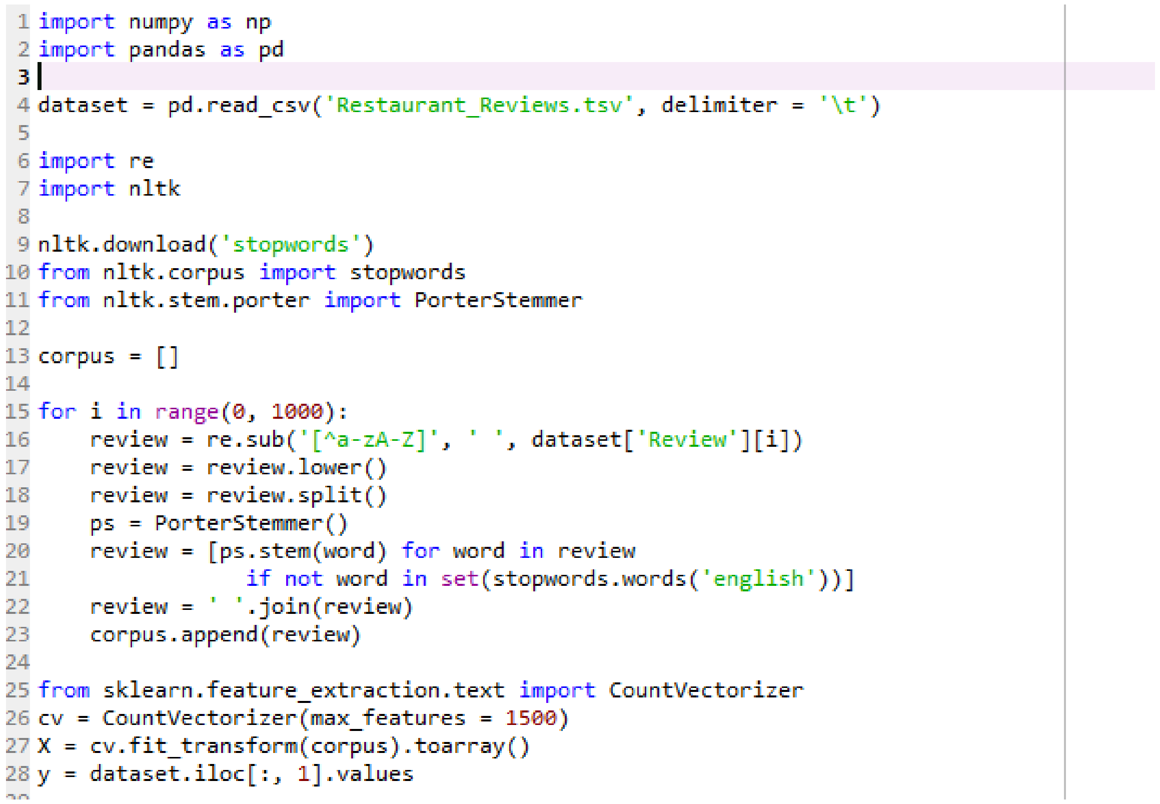
Fig 3.10 Code snippet

Parameters of the random forest classifier are:

n _estimators: This the number of the trees in the forest whose default value is set to 10.

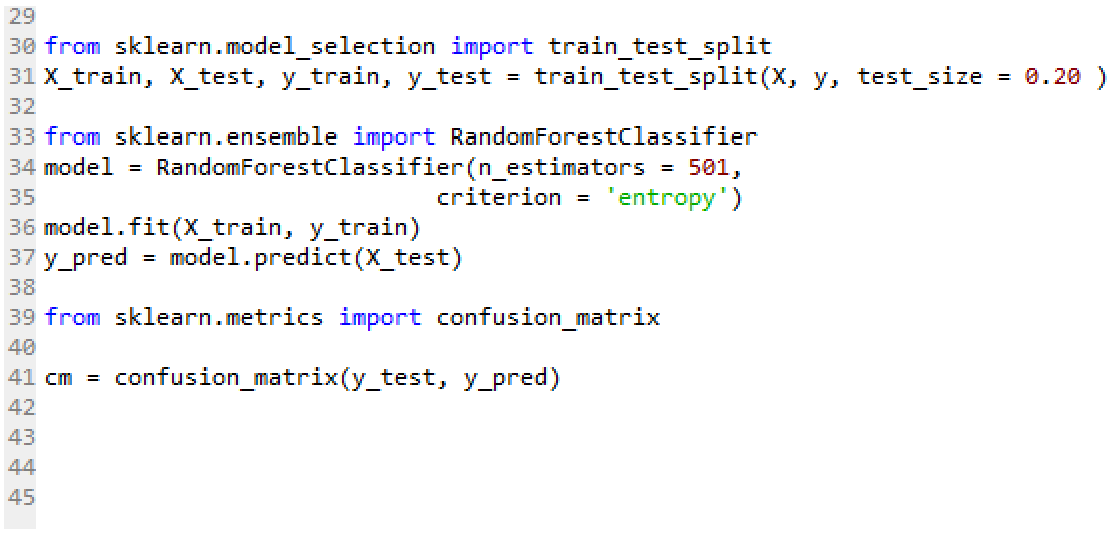
Criterion: entropy or the gini which are same as the decision tree classifiers.

min_samples split**:** It is the minimum number of the working set size at the node required to split whose default value is registered as 2.



```
import numpy as np
import pandas as pd

dataset = pd.read_csv('Restaurant_Reviews.tsv', delimiter = '\t')

import re
import nltk

nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

corpus = []

for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review
                if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values
```

Fig 3.11 Code snippet



```python

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20 )

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 501,
                               criterion = 'entropy')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

Fig 3.12 Code snippet

Model’s:

In the message dependent on the nostalgic investigation , we assemble the benchmark model and the element based model. We likewise attempt to play out the orders utilizing a mix of both of these models. Our methodology could be partitioned into different advances. Every one of these given advances are free of the other however are significant at same time. Figure 1 and 2 speaks to the methodology for the preparation and the testing the model.

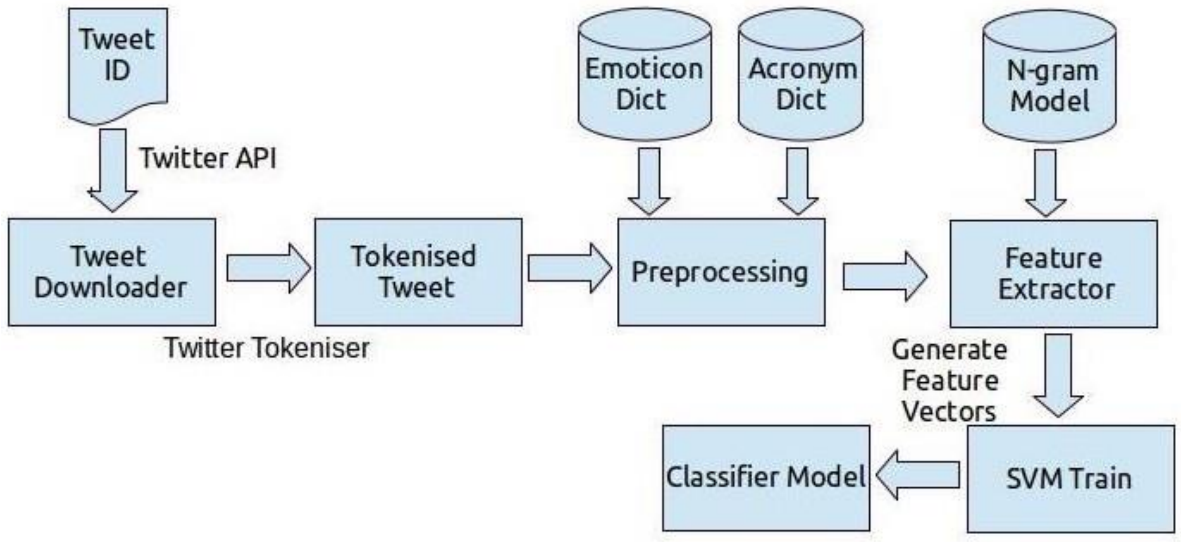


Fig. Flow diagram of training

In the given benchmarks approach, we first clean a given tweet. We play out a preprocessing step recorded and the become familiar with the positives, negatives and impartial frequencie of the unigram, bigram and trigram in the preparing. Each token has three likelihood score: Positive (Pp), Negatives (Np) and Neutral Probabilitys (NEp).

P_f = *Positive Training Sets frequency*

N_f = *Negative Training Sets frequency*

NE_f = Neutral Training Sets frequency

P_p = Positive Probabilitys = $P_f / (P_f + N_f + NE_f)$

N= Negative Probability's = $N_F / (P_f + N_f + NE)$

NE= Neutral Probability's = $NE / (P_f + N_f + NE_f)$

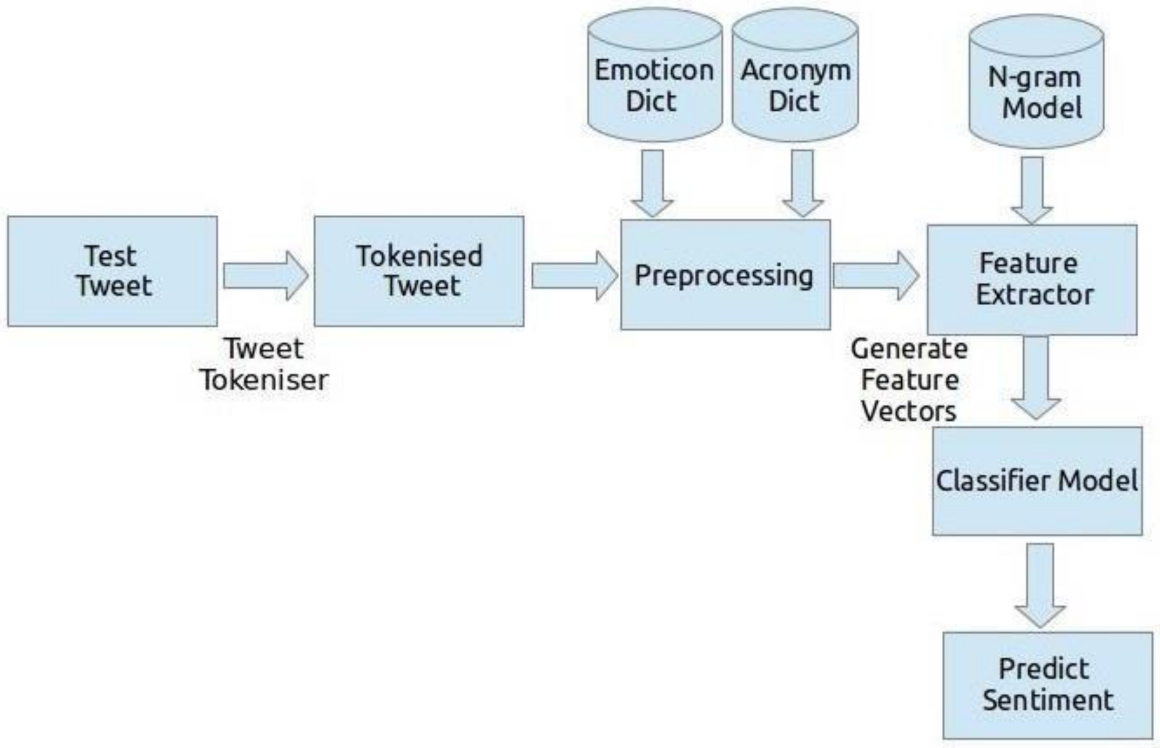


Fig. flow diagram of testing

Next we make a component vector of the tokens which can recognize the opinion of the tweets with high certainty. For instance, the nearness of token like am cheerful!, love, bullsh*t ! encourages us in the confirming that the tweets convey positive, negative or impartial supposition with high certainty. We call such words as Emotion Determiner. A token is considered as Emotion Determiner utilizing something like the hypothesis of the Triangular Inequality. The likelihood of feeling for any one supposition must be more prominent than or equivalent to the likelihood of the other two conclusions by a specific limit ted. It is discovered that we have various edges for unigrams, bigrams and trigrams. The parameter's for three tokens is tuned and the ideal edge esteems are being found. Note it before ascertaining the likelihood esteem, we sift through those tokens which are rare (which show up in under 10 tweets). Table 4 shows a rundown of unigrams, bigrams and trigrams which comply with the base ideal edge criteria. It tends to be seen that the nearness of such tokens guarentees the conclusion of the tweet with a high certainty.

Highlight based model:

A. Earlier Polarity Scoring some of our highlights depend on earlier extremity of words. For acquiring the earlier extremity of words, we use AFINN lexicon and expand it utilizing senti-Wordnet. We first look into the tokens in the tweets in the AFINN vocabulary. This is the lexicon of around 2490 English language word relegates each word an agreeableness score between - 5 (Negative) and +5 (Positive). We initially standardize the scores by plunging each score by the scale (which is equivalent to 5). In the event that a word isn't straightforwardly found in the lexicon we recover all the equivalent word from Wordnet. We at that point search for every one of the equivalent words in AFINN. On the off chance that any equivalent word is found in AFINN, we allocate the first word a similar enjoyableness score as its equivalent word. In the event that none of the equivalent words is available in AFINN, we play out a subsequent level turn upward in the senti-Wordnet lexicon. On the off chance that the word is available in senti-Wordnet, we dole out the score recovered from senti-Wordnet.

B. Highlights We propose the arrangement of the highlights recorded in Table 1 for our trial. These are a sum of 22 highlights. We ascertain every one of these highlights for the entire tweets on account of message based the wistful examination and for the all-inclusive expressions (got by taking those 2 tokens on either sides of the separated expressions) if there should arise an occurrence of the expression based nostalgic investigation. We allude to these highlights as the Emotion-includes all through paper. Our highlights can be isolated into three general classes: ones that are principally tallies of different highlights and in this manner the estimation of the element is a characteristic number N. The Second highlights whose worth is the genuine number R. These are essentially includes that catch the score recovered from AFINN. Thirdly, highlights whose qualities are boolean B. These are the pack of words, nearness of the shout marks and the promoted content. Table 1 outlines the highlights utilized in our test.

Chapter 4

PERFORMANCE ANALYSIS

CONFUSION MATRIX

Confusion Matrix describes the complete performance of the model.

For example,

In case of a binary classification problem. We can relate to two classes: YES or NO. Also, the Naïve Bayes classifier that is used in this project predicts a class for a given input sample. On testing our model, we get a confusion matrix that looks like:

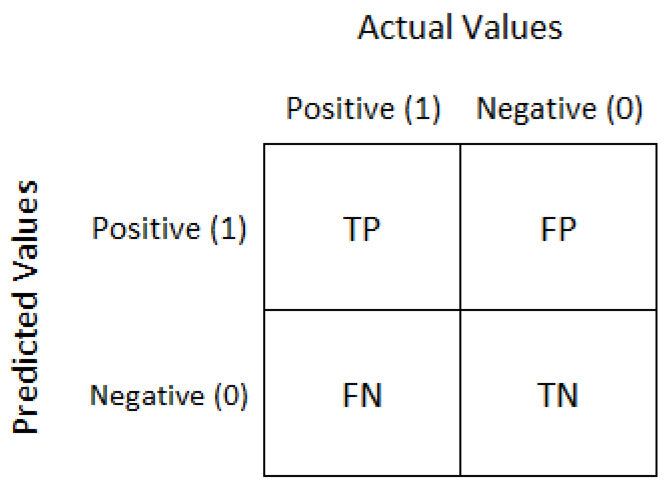


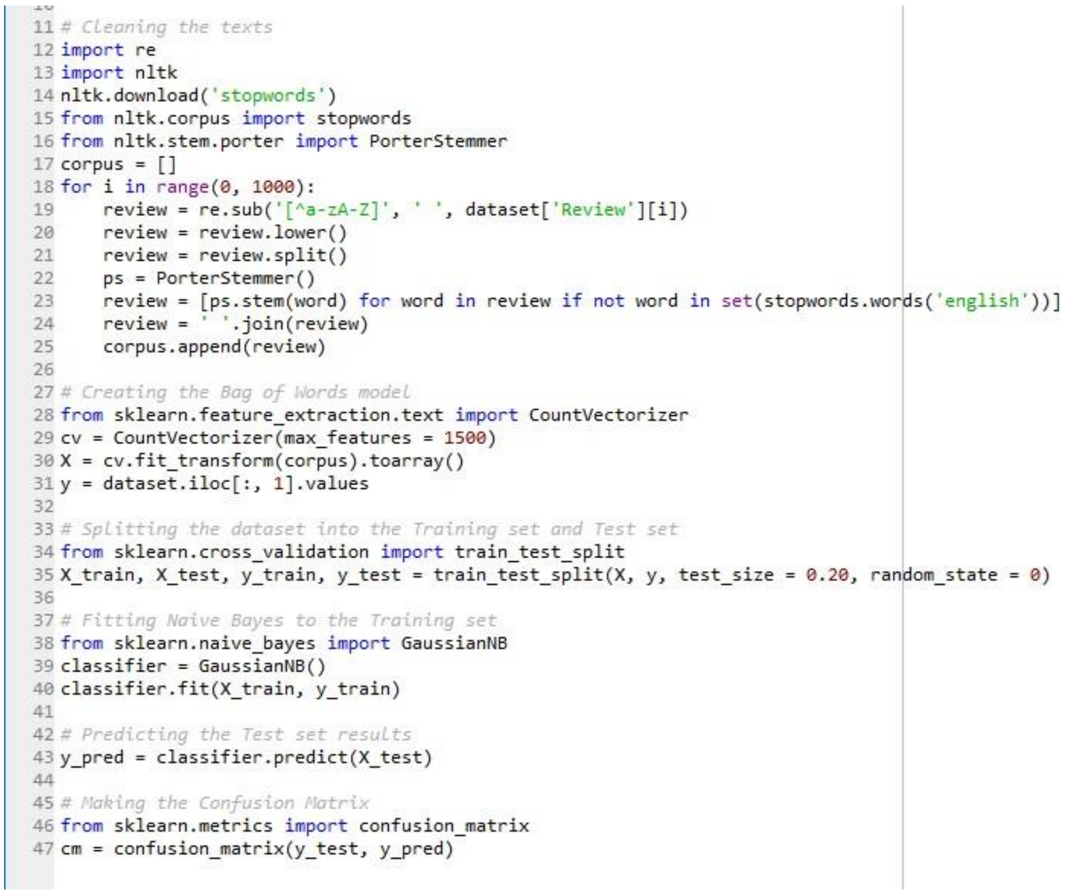
Fig. Confusion Matrix

There are 4 important terms:

- **True Positives**: When our prediction was "1" which was same as actual output.
- **True Negatives**: When our prediction was "0" which was same as actual output.
- **False Positives**: When our prediction was "1" and actual output was "0".
- **False Negatives**: When our prediction was "0" and actual output was "1".
- Matrix's accuracy can be computed by taking the average of the values lying **diagonally** i.e.

$$Accuracy = \frac{TruePositives + FalseNegatives}{TotalNumberofSamples}$$

Confusion Matrix forms the basis for the other types of metrics.



```python
# Cleaning the texts
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
corpus = []
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)

# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Fig. Code Snippet #1

F1 SCORE

F1 Score is utilized to gauge a test's exactness

F1 Score is the Harmonic Mean among the precision and recall. The range for F1 Score is [0, 1]. It reveals to you how exact your classifier is (what number of cases it orders accurately), just as how powerful it will be (it doesn't miss a critical number of cases). High precision however lower recall, gives you an incredibly precise, yet it at that point misses' countless occurrences that are hard to order. The more noteworthy the F1 Score, the better is the performance of our model. Scientifically, it tends to be communicated as:

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

F1 Score

F1 Score attempts to discover the harmony among precision and recall.

• Precision: It is the quantity of right positive outcomes separated by the quantity of positive outcomes anticipated by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Precision

- **Recall:** It is the number of true positives divided by the number of *all applicable* samples (all samples that should have been identified as positive).

CLASSIFICATION ACCURACY

Classification Accuracy is the ratio of number of correct predictions to the total number of predictions made.

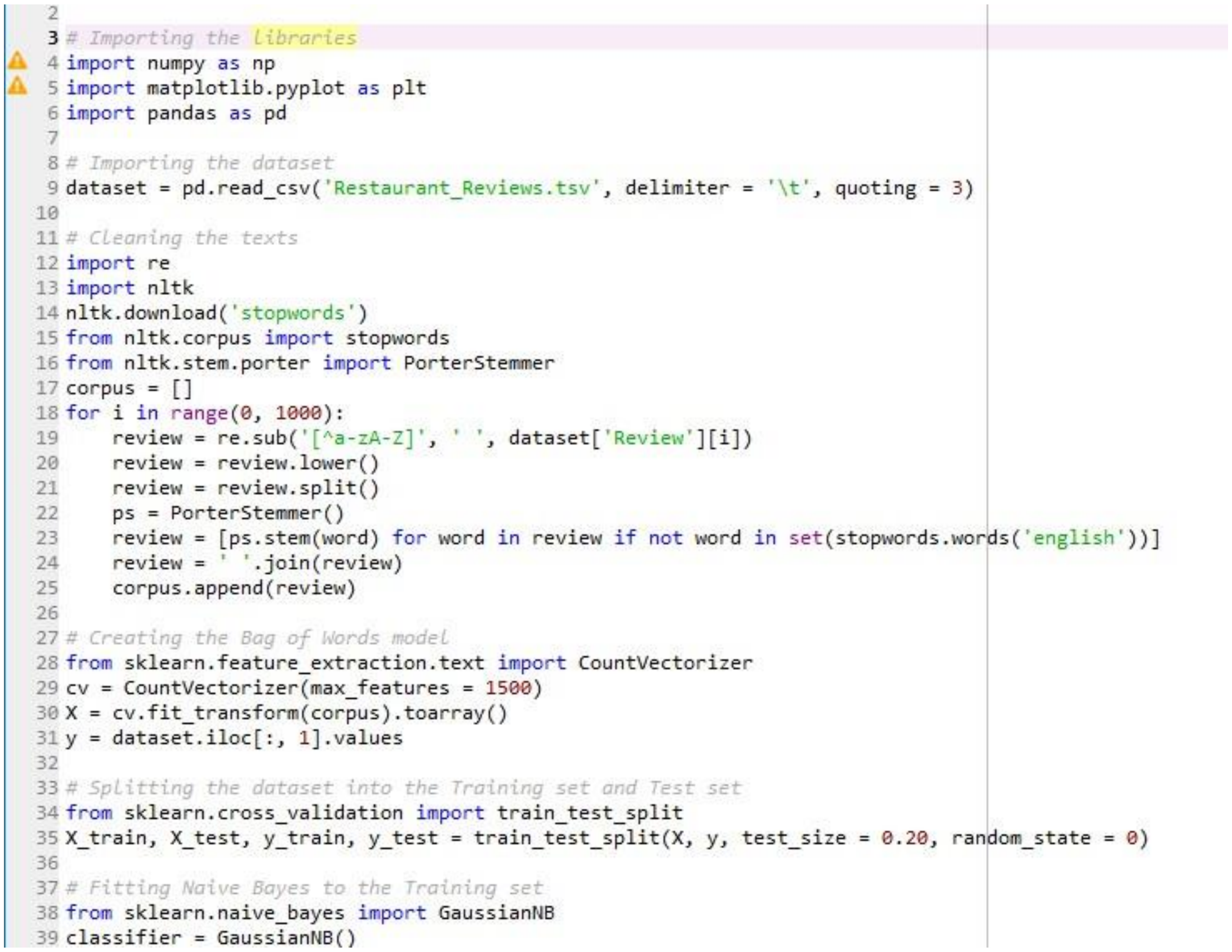
$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

It works well only if each class has equal number of samples.

For example, consider class A having 98% samples and Class B having 2% samples in our training set. So, it was easy for our model to attain **98% training accuracy** by simply predicting every training sample present in class A.

When the same model is tested on a test set with Class A having 60% of the samples and Class B with 40% of the samples, then the **test accuracy would drop down to 60%.** Classification Accuracy is great, but sometimes gives us the false sense of achieving high accuracy.

The serious issue emerges, when the expense of misclassification of the minor class tests are extremely high. On the off chance that we manage an uncommon however lethal ailment, the expense of neglecting to analyze the illness of a wiped-out individual is a lot higher than the expense of sending a solid individual to more tests.



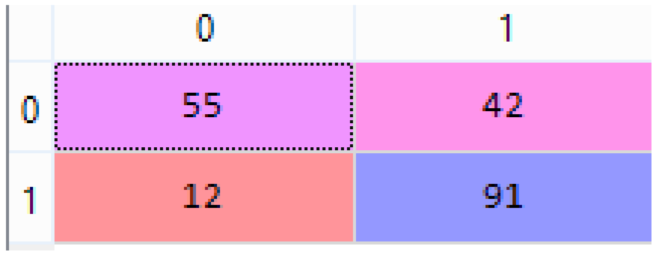
```python

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Restaurant_Reviews.tsv', delimiter = '\t', quoting = 3)

# Cleaning the texts
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
corpus = []
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)

# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
```

Fig. Code Snippet #2

MAX FEATURES

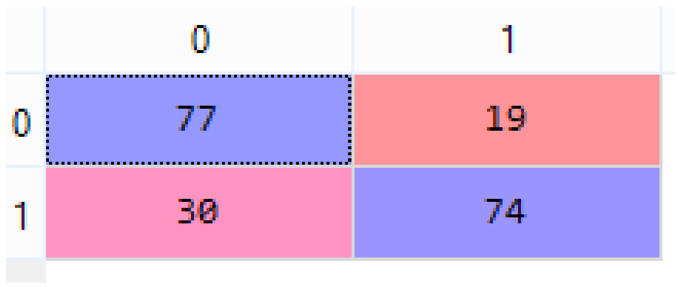
#max_features argument is used to form columns of only 1500 most frequent unique words in reviews. This removes most of the words which are only occurring once or twice for example since their presence don't affect our model much. We can decide no. of max features by reducing no. of unique words (which we can get by not taking "max_features" argument) to some extent. Parameter "max_features" reduces sparsity in this project.

Comparing the result of both the model using the confusion matrix created by both of the predictive model which are naïve Bayes classifiers and the random forest classifier.



	0	1
0	55	42
1	12	91

Fig. Confusion matrix for Naïve Bayes Classifier



	0	1
0	77	19
1	30	74

Fig. Confusion matrix for Random Forest Classifier

There for the accuracy for the both the models are as follows:

Accuracy for Naïve Bayes= (55+91) / (55+91+12+42)=146/200=73%

Accuracy for Random Forest Classifier= (77+74) / (77+19+74+30)=151/200=75.5%

So, the accuracy of random forest classifier is more than the accuracy of naïve bayes classifier.

RESULT:

Random Forest Classifier is better than naïve Bayes classifier as it gives better result for large number of test cases than naïve Bayes and it gives better result for the numerical and categorical data.

CHAPTER 5

CONCLUSION

Natural Language Processing, typically abbreviated as NLP, is a part of man-made reasoning that manages the collaboration among PCs and people utilizing the normal language.
A definitive target of NLP is to peruse, interpret, comprehend, and understand the human dialects in a way that is important. Most NLP methods depend on AI to get importance from human dialects.

As the measure of data accessible online is developing, the need to get to it turns out to be progressively significant and the estimation of regular language handling applications turns out to be clear. Machine interpretation causes us vanquish language obstructions that we frequently experience by deciphering specialized manuals, bolster substance or inventories at an essentially decreased expense. The test with machine interpretation advancements isn't in deciphering words, however in understanding the importance of sentences to give a genuine interpretation.

The objective of supposition examination is to recognize slant among a few posts or even in a similar post where feeling isn't in every case unequivocally communicated. Organizations utilize normal language handling applications, for example, notion examination, to distinguish opinons and assessment online to enable them to comprehend customers' opinion of their items and administrations (i.e., "I love the new iPhone" and, a couple of lines later "Yet once in a while it doesn't function admirably" where the individual is as yet discussing the iPhone) and by and large markers of their notoriety. Past deciding basic extremity, estimation investigation comprehends notion in setting to assist you with bettering comprehend what's behind a communicated supposition, which can be amazingly applicable in understanding and driving acquiring choices.
Natural Language preparing is viewed as a troublesome issue in software engineering. It's the idea of the human language that makes NLP troublesome.

The principles that direct the death of data utilizing regular dialects are difficult for PCs to comprehend. A portion of these principles can be high-levelled and unique; for instance, when somebody utilizes a snide comment to pass data.

Then again, a portion of these guidelines can be low-levelled; for instance, utilizing the character "s" to connote the majority of things. Completely understanding the human language requires understanding both the words and how the ideas are associated with convey the planned message. While people can without much of a stretch ace a language, the vagueness and uncertain attributes of the regular dialects are what make NLP hard for machines to execute.

So, this model firstly extracts the data from twitter API and then applies text cleaning with the help of python regular expressions and removed stop words with the help of natural language processing libraries. The model then creates a bag of words model and frames a sparse matrix.

Few sparsity minimisation techniques were applied followed by the implementation of a Gaussian Naïve Bayes Classifier and then finally analysing the model performance with the help of confusion matrix and classification Report.

The model was trained on the random forest classifier and we found that the random classifier is better than the naïve bayes and it has better accuracy as random forest classifier works better for large number of datasets as it works on the concept of the decision tree and can used for clustering, statistical inference and feature selection as well and works good with numerical data and the categorical data too yielding better accuracy for the model.

References

1. Acronym list. [Online]. Available: http://www.noslang.com/dictionary/
2. Afinn-111. [Online]. Available: http://www2.imm.dtu.dk/pubdb/views/ publication details.php?id=6010
3. Dataset. [Online]. Available: http://alt.qcri.org/semeval2014/task9/index. php?id=data-and-tools
4. Emoticon list. [Online]. Available: http://en.wikipedia.org/wiki/List of emoticons
5. Libsvm. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
6. Tweet downloader. [Online]. Available: http://alt.qcri.org/semeval2014/ task9/index.php?id=data-and-tools
7. Twitter. [Online]. Available: http://twitter.com
8. Twitter nlp. [Online]. Available: https://github.com/brendano/ ark-tweet-nlp/tree/master/src/cmu/arktweetnlp
9. Efthymios Kouloumpis and Johanna Moore,IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 3, July 2012 (pg 16-19)
10. S. Batra and D. Rao, ”Entity Based Sentiment Analysis on Twitter”, Stanford University,2010(pg 31-37)
11. Saif M.Mohammad and Xiaodan zhu ,Sentiment Analysis on of social media texts:,2014 (pg 13-15)

12. Ekaterina kochmar,University of Cambridge,at the Cambridge coding Academy Data Science.2016

13. Manju Venugopalan and Deepa Gupta ,Exploring Sentiment Analysis on Twitter Data, IEEE 2015

14. Brett Duncan and Yanqing Zhang, Neural Networks for Sentiment Analysis on Twitter.2017

15. Afroze Ibrahim Baqapuri, Twitter Sentiment Analysis: The Good the Bad and the OMG!, Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media.2011

16. Ben Parr. Twitter Has 100 Million Monthly Active Users; 50% Log In Everyday.
<http://mashable.com/2011/10/17/twitter-costolo-stats/>

17. Google App Engine
https://developers.google.com/appengine/

18. Google Chart API
https://developers.google.com/chart/

19. Jinja2: Templating Language for Python
http://jinja.pocoo.org/

20. Maggie Shields, Technology Reporter, BBC News. Twitter co-founder Jack Dorsey rejoins company.
http://www.bbc.co.uk/news/business-12889048

21. Multi Perspective Question Answering (MPQA) Online Lexicon

    http://www.cs.pitt.edu/mpqa/subj_lexicon.html/

22. Tweet Stream: Simple Twitter Streaming API Access

    http://pypi.python.org/pypi/tweetstream/

23. Twitter REST API

    https://dev.twitter.com/docs/api/

jejejej

ORIGINALITY REPORT

18%	10%	3%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	researchweb.iiit.ac.in Internet Source	3%
2	towardsdatascience.com Internet Source	3%
3	Submitted to VIT University Student Paper	2%
4	Submitted to University of Stirling Student Paper	1%
5	Submitted to University of Florida Student Paper	1%
6	Submitted to Computer College Student Paper	1%
7	Submitted to University of Wales Institute, Cardiff Student Paper	1%
8	rcciit.org Internet Source	1%
9	Submitted to Texas A & M University, Kingville	