# Table of Contents

## PROBLEM STATEMENT

Sentiment Analysis is a sub-field of Natural Language Processing. This is characterised as a method of recognizing and classifying opinions from a text document and is helpful in determining user's intention for specific subject is neutral, negative or positive. It is also termed as Opinion Mining. The motive of our proposed work is to try to implement a Twitter Sentiment Analysis Model that helps to overcome the challenges of identifying the sentiments of tweets. The necessary details regarding the dataset are:

The dataset provided is the Sentiment140Dataset which consists of 1,600,000 tweets that have been extracted using the Twitter API. The various columns present in the dataset are:

- **target:** the polarity of the tweet (positive or negative)
- **ids:** Unique id of the tweet
- **date:** the date of the tweet
- **flag:** It refers to the query. If no such query exists then it is NO QUERY.
- **user:** It refers to the name of the user that tweeted
- **text:** It refers to the text of the tweet

# CHAPTER 1: INTRODUCTION

Sentiment analysis refers to identifying as well as classifying the sentiments that are expressed in the text source. Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of the people about a variety of topics.

Therefore, we need to develop an **Automated Machine Learning Sentiment Analysis Model** in order to compute the customer perception. Due to the presence of non-useful characters (collectively termed as the noise) along with useful data, it becomes difficult to implement models on them.

We aim to analyze the sentiment of the tweets provided from the **1.6 million tweets dataset** by developing a machine learning pipeline involving the use of two classifiers (**Logistic Regression, SVM**) along with using **Term Frequency- Inverse Document Frequency** (**TF-IDF)**. The performance of these classifiers is then evaluated using **accuracy** and **F1 Scores**.

Twitter has more or less than 310 million monthly active users with total 500 million of tweets per day. Tweet is a 140-characters message that can contain opinion or information about recent happenings or even user’s emotion. Tweets are not properly structured because user do not care about spelling and grammatical construction when posting their tweets.

In order to classify the contents of the collected tweets based on the selected topics, machine learning can be used to provide computational intelligences to the technology so it can learn and adapt from the given data independently. Application of machine learning methods, especially on social networking sites, can show varied results from auto recommendation, classification about specific interests, sentiments and so on.

The Internet has been very useful in helping today’s world express their perspectives globally. This is done through blog entries, on the web discussions forums, item audit sites, and so on. Individuals worldwide depend on this client, produced content extensively. For example, if someone wants to buy some product, then they first look up its reviews and comments before finalizing it. But it is not humanly possible for a particular person to sit and look at every single review available. It would simply be a waste of time. Hence, to make this process easier, it can be automated. Machine Learning (ML) plays a significantly important part here. The process of Sentiment Analysis (SA) falling under ML helps the system understand the sentiment of a particular statement made. The system is built using several ML algorithms that can understand the nature of sentiment or a set of the same. In research, methods of ML have prevailed over knowledge- and dictionary-based methods to determine the polarity. Polarity here is a semantic orientation that lies between two extremities, 0 and 1 or positive and negative. The paper proposes a system wherein data from twitter will be extracted on which SA will be performed. That data is saved in data frame. Then, some cleaning and pre-processing steps are performed on it so that, accurate information is utilized to fit the ML model which helps to predict labels for unknown cleaned and pre-processed data samples. Twitter is one of the popular sources containing a relatively huge amount of data. For performing sentiment analysis, certain supervised machine learning methods (algorithms) have been utilized to accomplish precise outcomes. Some of them are multinomial naive Bayes, linear support vector classifiers, and logistic regression classifiers.

# CHAPTER 2: LITERATURE SURVEY

WordNet is utilised for identifying sentiment associated with a term in various ways. Distance metric was generated on WordNet and the sematic orientation of adjectives was found by them.

In 1970, Ekman et al. did immense research in multiple expressions on face and expressed that these are adequate for identifying emotions.

Akba et al. used collection of functionality-focused on information gain and chi-Square metrics upon completion of the lemmatization and stemming process, the insightful characteristics are selected. The tests performed have shown that the correlation of feature engineering measurements with support vector machine classifiers has increased relative to previous research.

Twitter corpus was developed by gathering twitter posts from application programming interface provided by twitter and interpreting them by making use of emojis. Sentiment analysis model was constructed by considering this corpus.

A technique to get specifications such as battery, processor, camera for a specific product was developed in. The main technical aspects of a product are found and classified. Depending on whether neutral, negative or positive scores were assigned for each and every specification. By combining all the scores of independent features, the overall rating of a product was identified.

For categorizing the feedback, an upgraded method from Support Vector Machine was proposed in. Depending on the words associated with emotions, SentiWordNet assigned the sentiment scores. By changing these scores, they developed a modified model.

Pang B. and Lee L in 2004, expressed that analysis of sentiments tries recognizing the perspectives of a basic content span; an example is classifying a film audit as positive or negative.  To decide the polarity, the creator defines a novel machine-learning strategy which applies content classification strategies to only the subjective parts of the text document. Separating these parts can be executed utilizing proficient strategies for discovering least cuts in graphs which significantly encourages consolidation of cross-sentence relevant limitations.

Gautam G.  and Yadav P in 2014 define another method for communicating the feelings and emotions of people. In general, it is a way which tremendously measures the information where clients can see the emotions of different clients which are categorized into various classes of sentiments and are progressively developing as a main factor in basic leadership. The work defined is useful to view the data as the quantity of tweets where sentiments or emotions are either good or bad, or neutral.

Twitter is critical stage for take after estimation of opinion which is an exceptionally difficult issue. Public feeling examination is an exceptionally basic to investigate, break down and sort out clients sees for better basic leadership.  Sentiment examination is procedure of recognizing good and bad sentiment, feelings and examinations in content.  It is valuable for product users to examine the conclusion of items, or organizations need to screen people in general opinion of their brands.

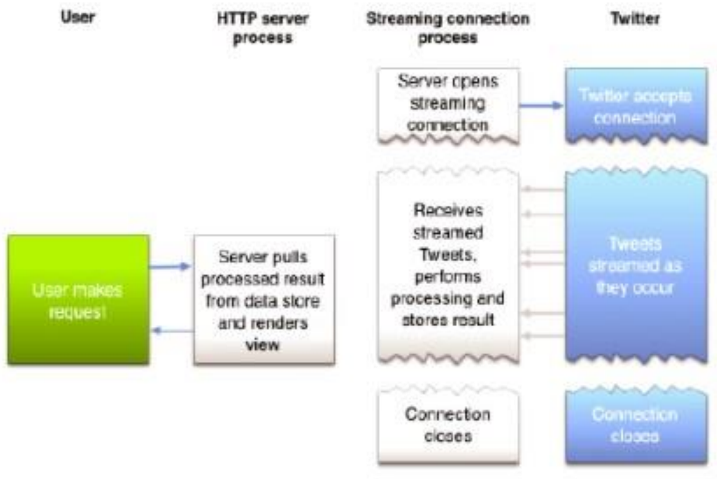
John C and Jonathon R. specified an imperative sub-errand of emotions examination is classification of polarity, in which content is delegated being good or bad. Machine learning methods can play out this classification adequately. Be that as it may, it requires a substantial corpus of preparing information, and various examinations have shown that the great execution of supervised models is reliant on a decent match between the preparation and testing information as for the area, subject, and time period. Pitifully supervised procedures utilize an extensive accumulation of unlabelled content to decide sentiment, thus their execution might be less subject to the area, subject.

Hassan S. described in their paper that analysis of sentiment over Twitter gives companies a very best way to analyse the public's sentiment for their brand, business, products, etc. An extensive variety of features and techniques for preparing opinion classifiers for Twitter datasets have been inquired about as of late with varying.

## PIPELINE: -

1) Fetching Tweets:

   Fetching tweets is a process of getting the tweets directly from Twitter server using Twitter API (Application Programming Interface) based on keywords used.


User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
User makes request
Server pulls processed result from data store and renders view
Connection closes
Connection closes

2) Preprocessing:

   Preprocessing is a process of cleaning noise like links, punctuation or stop words that does not contain any useful information in the text. There are few steps of preprocessing used in this research such as:

   - Removing Stop words. to remove words like a, most, and, is and so on in the text because those words do not contain any useful information.
   - Removing Punctuations.
   - Removing repeating character.
   - Removing URL's.
   - Removing Numbers.
   - Tokenizing, to break text into each word.

- Stemming, to change a word in the text into its base term or root term. Example, happiness to happy.
- Lemmatizing, the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.

3) Visualization:

Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs.

- Matplotlib.
- Seaborn.
- NumPy.
- WordCloud.

4) Text Feature Extraction:

Text feature extraction is a process of converting text into set of features in real number form side a vector that will be used as input for classification.

- Tf-idf Vectorizer:

  TFIDF is another way to convert textual data to numeric form, and is short for Term Frequency-Inverse Document Frequency. The vector value it yields is the product of these two terms; TF and IDF.

  Let's first look at Term Frequency. We have already looked at term frequency with count vectorizer, but this time, we need one more step to calculate the relative frequency. Let's say we have two documents in our corpus as below.

  1.I love dogs
  2.I hate dogs and knitting

  Relative term frequency is calculated for each term within each document as below.

$$TF(t,d) = \frac{number\ of\ times\ term(t)\ appears\ in\ document(d)}{total\ number\ of\ terms\ in\ document(d)}$$

For example, if we calculate relative term frequency for 'I' in both document 1 and 2, it will be

$$TF('I',d1) = \frac{1}{3} \approx 0.33$$

$$TF('I',d2) = \frac{1}{5} = 0.2$$

Next, we need to get Inverse Document Frequency, which measures how important a word is to differentiate each document by following the calculation as below.

$$IDF(t,D) = \log\left(\frac{\textit{total number of documents}(D)}{\textit{number of documents with the term}(t)\textit{ in it}}\right)$$

If we calculate inverse document frequency for ‘I’.

$$IDF('I',D) = \log\left(\frac{2}{2}\right) = 0$$

Once we have the values for TF and IDF, now we can calculate TFIDF as below.

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

Following the case of our example, TFIDF for the term ‘I’ in both documents will be as below.

$$TFIDF('I',d1,D) = TF('I',d1) \cdot IDF('I',D) = 0.33 \times 0 = 0$$
$$TFIDF('I',d2,D) = TF('I',d2) \cdot IDF('I',D) = 0.2 \times 0 = 0$$

As you can see, the term ‘I’ appeared equally in both documents, and the TFIDF score is 0, which means the term is not really informative in differentiating documents. The rest is same as count vectorizer, TFIDF vectorizer will calculate these scores for terms in documents, and convert textual data into the numeric form.

5) Machine Learning:

Machine Learning is one of computer science branches that focuses on providing the technology the ability to learn and adapt from given data so technology can learn and grow independently without being programmed explicitly by developers. One of machine learning methods named Logistic Regression classifier and SVM (support vector machine) classifier is used for classification task in this research.

- Logistic Regression:

  Logistic regression is a supervised machine learning technique for classification problems. Supervised machine learning algorithms train on a labeled dataset along with an answer key which it uses to train and evaluate its accuracy. The goal of the model is to learn and approximate a mapping function f(Xi) = Y from input variables {x1, x2, xn} to output variable(Y). It is called supervised because the model predictions are iteratively evaluated and corrected against the output values, until an acceptable performance is achieved.

  Logistic regression achieves the best predictions using the maximum likelihood technique. Sigmoid is a mathematical function having a characteristic that can take any real value between -∞ and +∞ and map it to a real value between 0 to 1. So, if the outcome of sigmoid function is more than 0.5 then we classify it as positive class and if it is less than 0.5 then we can classify it as negative class. $S(z) = \frac{1}{1+e^{-z}}$

- SVM (support vector machine):

  It is a non-probabilistic model that utilizes a portrayal of text models as focuses in a multidimensional space. These examples are mapped with the goal that the instances of the diverse categories (sentiments) have a place with particular areas of that space. Later, the new messages are mapped onto that equivalent space and are predicted to have a place with a classification dependent on which category they fall into. In the SVM algorithm, the fundamental goal is to boost the edge between information points and the hyperplane. The loss function that helps with this is called a hinge loss. The equation of the hyperplane is given as:

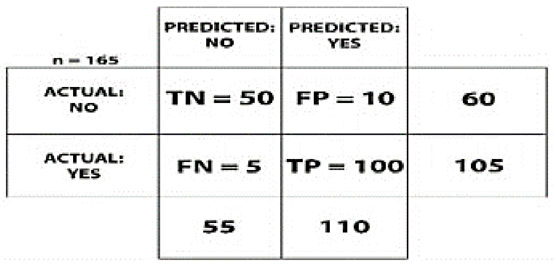
$$w.x - b = 0$$

$$c\left(x, y, f\left(x\right)\right) = \begin{cases} 0, ify * f\left(x\right) \geq 1 \\ 1 - y * f\left(x\right), \text{else} \end{cases}$$

The cost is 0 if the predicted, and the actual value is of a similar sign. On the off chance that they are not, at that point, figure the loss esteem. For performing sentiment analysis, logistic regression is considered over naive Bayes because naive Bayes assumes all the features used in model building to be conditionally independent whereas logistic regression splits feature space linearly and typically works reasonably well even when some of the variables are correlated, and on the other hand, logistic regression and SVM with a linear kernel have similar performance but depending on the features, one may be more efficient than the other.

- Confusion matrix:

  Confusion Matrix is often used in classification task to measure the accuracy rate of the used classifier. Confusion Matrix uses new data that is not used in training process. Below is the example of using Confusion Matrix in classification task:

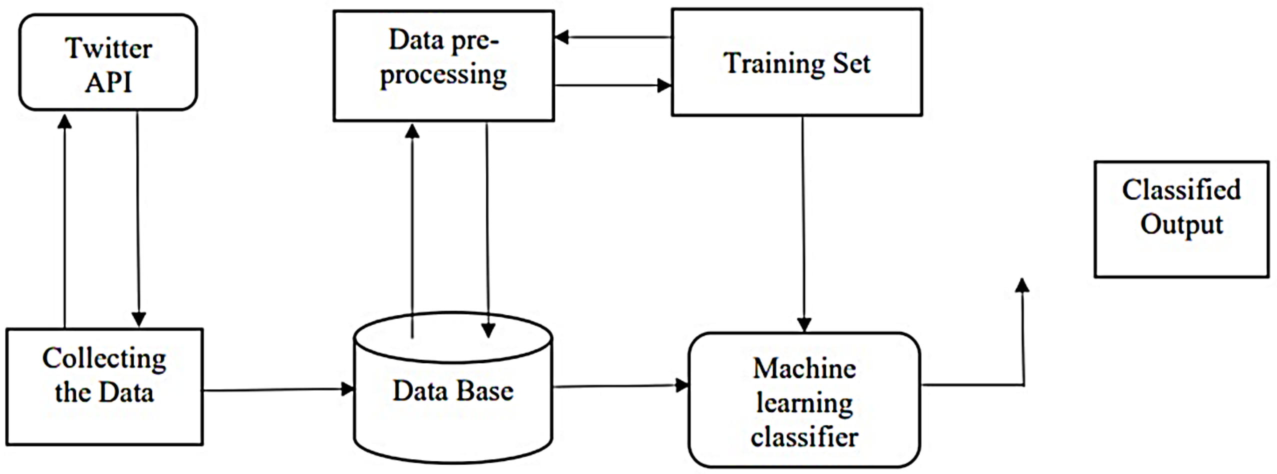


| n = 165 | PREDICTED: NO | PREDICTED: YES | |
|---|---|---|---|
| ACTUAL: NO | TN = 50 | FP = 10 | 60 |
| ACTUAL: YES | FN = 5 | TP = 100 | 105 |
| | 55 | 110 | |

From the example above, the calculation to determine the accuracy rate of the used classifier can be formulated as follows:

$$Accuracy = \frac{(TP+TN)}{total} = \frac{(100+50)}{165} = 0.91$$

# CHAPTER 3: DEAILED DESIGN



# CHAPTER 4: PROJECT SPECIFIC REQUIREMENTS

1) Twitter API.

2) NLTK (natural language tool kit)

3) Libraries: NumPy, pandas, seaborn, word cloud etc.…

4) Software used: Google colab

5) Language: python

# CHAPTER 5: IMPLEMENTATION

```
import re
import numpy as np
import pandas as pd
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from nltk.stem import WordNetLemmatizer
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, classification_report

df = pd.read_csv('/content/drive/MyDrive/twitter/dataset.csv',
                 encoding='ISO-8859-1',
                 names=[
                        'target',
                        'id',
                        'date',
                        'flag',
                        'user',
                        'text'
                 ])

df.head()

df.info()
df.shape

df.dtypes

np.sum(df.isnull().any(axis=1))

df['target'].unique()

df['target'].nunique()

ax = df.groupby('target').count().plot(kind='bar', title='Distribution of data',leg
end='False')
ax.set_xticklabels(['Negative','Positive'], rotation=0)
text, sentiment = list(df['text']), list(df['target'])

sns.countplot(x='target', data=df)
```

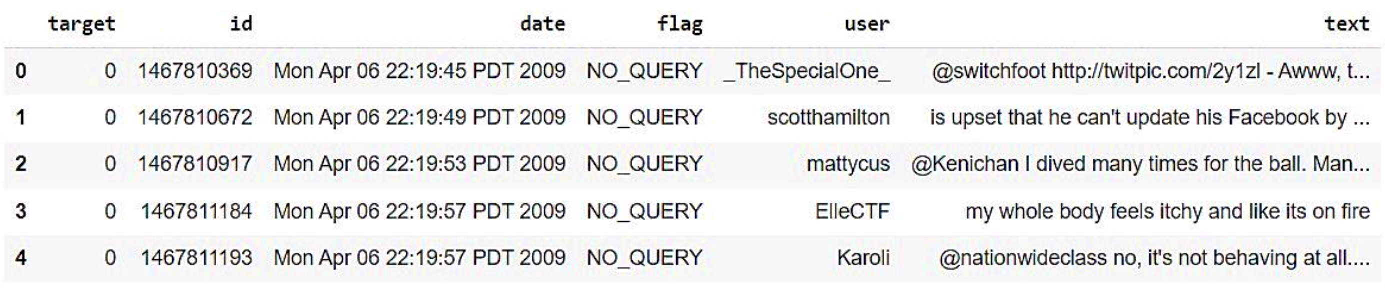
```
data=df[['text','target']]

data['target'] = data['target'].replace(4,1)

data['target'].unique()

data_pos = data[data['target'] == 1]
data_neg = data[data['target'] == 0]

data_pos = data_pos.iloc[:int(20000)]
data_neg = data_neg.iloc[:int(20000)]

dataset = pd.concat([data_pos, data_neg])

dataset['text']=dataset['text'].str.lower()
dataset['text'].tail()

stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all', 'am', 'an',
             'and','any','are', 'as', 'at', 'be', 'because', 'been', 'before',
             'being', 'below', 'between','both', 'by', 'can', 'd', 'did', 'do',
             'does', 'doing', 'down', 'during', 'each','few', 'for', 'from',
             'further', 'had', 'has', 'have', 'having', 'he', 'her', 'here',
             'hers', 'herself', 'him', 'himself', 'his', 'how', 'i', 'if', 'in',
             'into','is', 'it', 'its', 'itself', 'just', 'll', 'm', 'ma',
             'me', 'more', 'most','my', 'myself', 'now', 'o', 'of', 'on', 'once',
             'only', 'or', 'other', 'our', 'ours','ourselves', 'out', 'own', 're','
s', 'same', 'she', "shes", 'should', "shouldve",'so', 'some', 'such',
             't', 'than', 'that', "thatll", 'the', 'their', 'theirs', 'them',
             'themselves', 'then', 'there', 'these', 'they', 'this', 'those',
             'through', 'to', 'too','under', 'until', 'up', 've', 'very', 'was',
             'we', 'were', 'what', 'when', 'where','which','while', 'who', 'whom',
             'why', 'will', 'with', 'won', 'y', 'you', "youd","youll", "youre",
             "youve", 'your', 'yours', 'yourself', 'yourselves']


STOPWORDS = set(stopwordlist)
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
dataset['text'] = dataset['text'].apply(lambda text: cleaning_stopwords(text))
dataset['text'].head()

import string
english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
    translator = str.maketrans('', '', punctuations_list)
    return text.translate(translator)
dataset['text']= dataset['text'].apply(lambda x: cleaning_punctuations(x))
dataset['text'].tail()
```

```
def cleaning_repeating_char(text):
    return re.sub(r'(.)1+', r'1', text)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_repeating_char(x))
dataset['text'].tail()

def cleaning_URLs(data):
    return re.sub('((www.[^s]+)|(https?://[^s]+))',' ',data)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_URLs(x))
dataset['text'].tail()

def cleaning_numbers(data):
    return re.sub('[0-9]+', '', data)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_numbers(x))
dataset['text'].tail()

from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer(r'\w+')
dataset['text'] = dataset['text'].apply(tokenizer.tokenize)
dataset['text'].head()

import nltk
st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data
dataset['text']= dataset['text'].apply(lambda x: stemming_on_text(x))
dataset['text'].head()

from nltk.stem import WordNetLemmatizer

lm = nltk.WordNetLemmatizer()
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
    return data
dataset['text'] = dataset['text'].apply(lambda x: lemmatizer_on_text(x))
dataset['text'].head()

X=data.text
y=data.target

data_neg = data['text'][:800000]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(data_neg))
plt.imshow(wc)

data_pos = data['text'][800000:]
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
              collocations=False).generate(" ".join(data_pos))
plt.figure(figsize = (20,20))
plt.imshow(wc)
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.05, random_st
ate =26105111)

vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)
vectoriser.fit(X_train)
print('No. of feature_words: ', len(vectoriser.get_feature_names()))

X_train = vectoriser.transform(X_train)
X_test  = vectoriser.transform(X_test)

def model_Evaluate(model):
  y_pred = model.predict(X_test)
  print(classification_report(y_test, y_pred))
  cf_matrix = confusion_matrix(y_test, y_pred)
  categories = ['Negative','Positive']
  group_names = ['True Neg','False Pos', 'False Neg','True Pos']
  group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() / n
p.sum(cf_matrix)]
  labels = [f'{v1}n{v2}' for v1, v2 in zip(group_names,group_percentages)]
  labels = np.asarray(labels).reshape(2,2)
  sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
  xticklabels = categories, yticklabels = categories)
  plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
  plt.ylabel("Actual values" , fontdict = {'size':14}, labelpad = 10)
  plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)

LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
y_pred3 = LRmodel.predict(X_test)

from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred3)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc
_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()

SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)
model_Evaluate(SVCmodel)
y_pred2 = SVCmodel.predict(X_test)
```

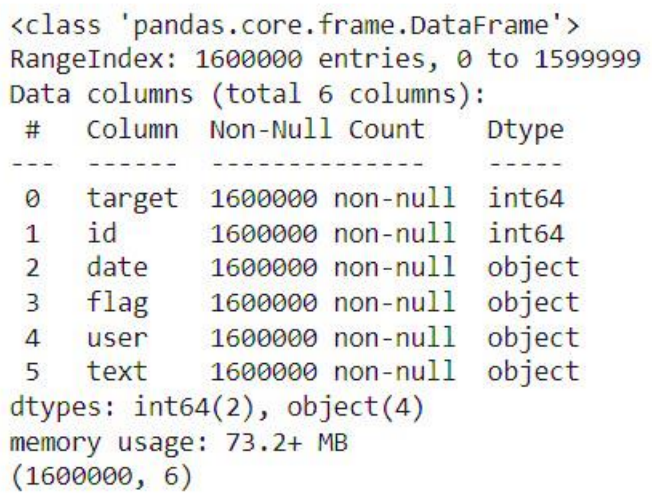
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred2)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc
_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

# CHAPTER 6: RESULTS



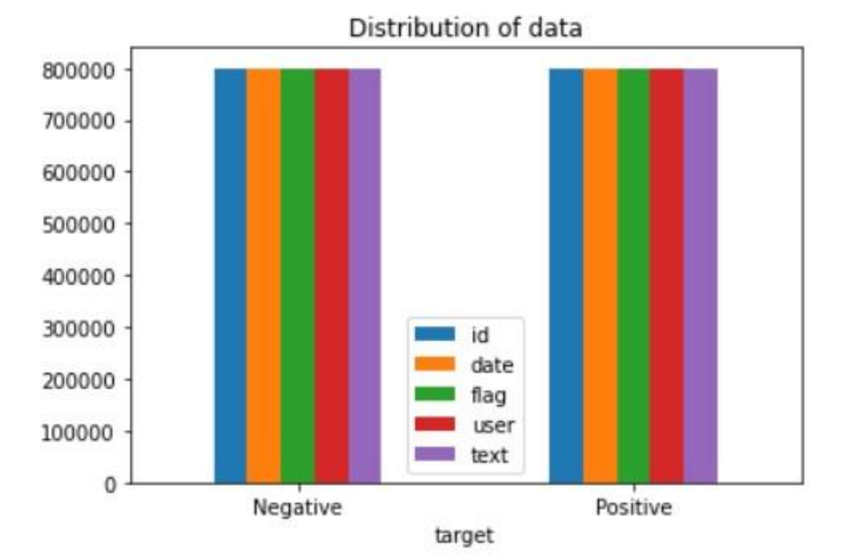
| | target | id | date | flag | user | text |
|---|---|---|---|---|---|---|
| **0** | 0 | 1467810369 | Mon Apr 06 22:19:45 PDT 2009 | NO_QUERY | _TheSpecialOne_ | @switchfoot http://twitpic.com/2y1zl - Awww, t... |
| **1** | 0 | 1467810672 | Mon Apr 06 22:19:49 PDT 2009 | NO_QUERY | scotthamilton | is upset that he can't update his Facebook by ... |
| **2** | 0 | 1467810917 | Mon Apr 06 22:19:53 PDT 2009 | NO_QUERY | mattycus | @Kenichan I dived many times for the ball. Man... |
| **3** | 0 | 1467811184 | Mon Apr 06 22:19:57 PDT 2009 | NO_QUERY | ElleCTF | my whole body feels itchy and like its on fire |
| **4** | 0 | 1467811193 | Mon Apr 06 22:19:57 PDT 2009 | NO_QUERY | Karoli | @nationwideclass no, it's not behaving at all.... |

Loaded dataset of tweets



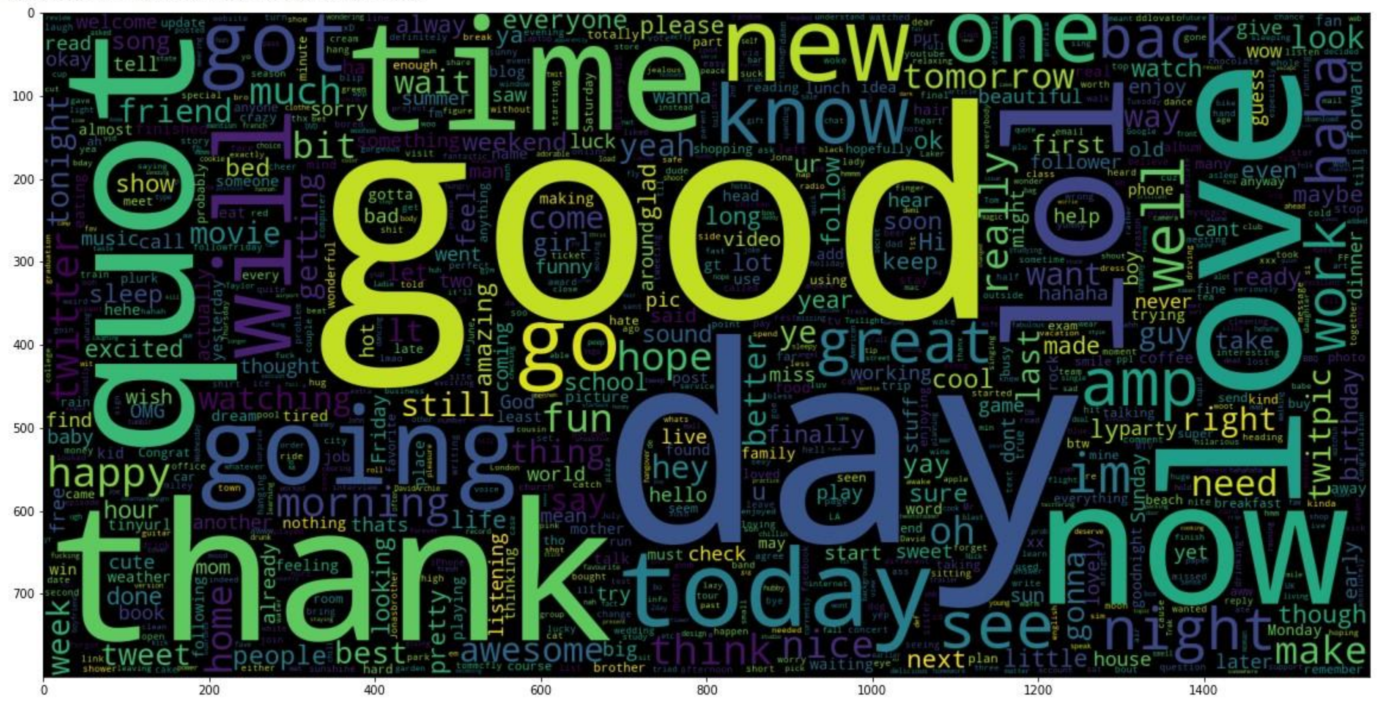
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype 
---  ------  --------------    ----- 
 0   target  1600000 non-null  int64 
 1   id      1600000 non-null  int64 
 2   date    1600000 non-null  object
 3   flag    1600000 non-null  object
 4   user    1600000 non-null  object
 5   text    1600000 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
(1600000, 6)
```

Dataset information



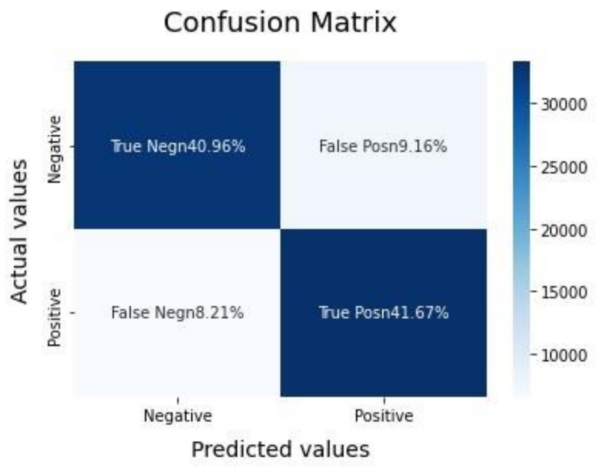
Distribution of data

```
<matplotlib.image.AxesImage at 0x7fe9fd2d3dd0>
```

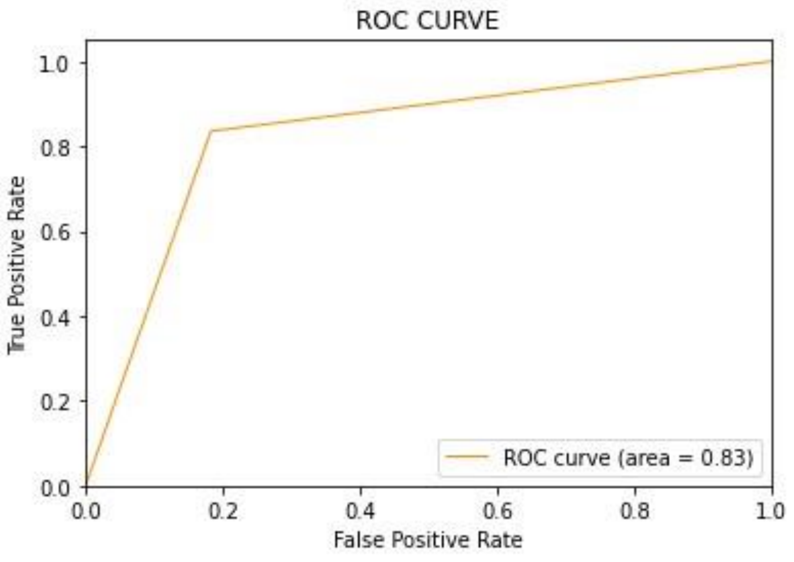


WordCloud Visualization

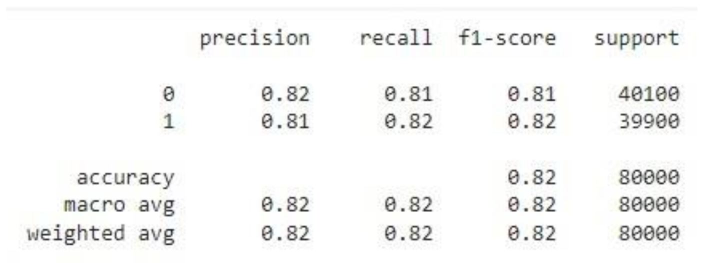
```
              precision    recall  f1-score   support

           0       0.83      0.82      0.83     40100
           1       0.82      0.84      0.83     39900

    accuracy                           0.83     80000
   macro avg       0.83      0.83      0.83     80000
weighted avg       0.83      0.83      0.83     80000
```



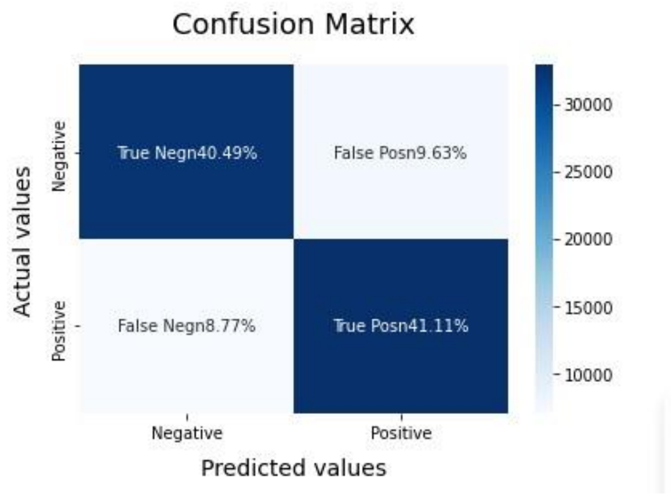
Logistic Regression result with accuracy and confusion matrix



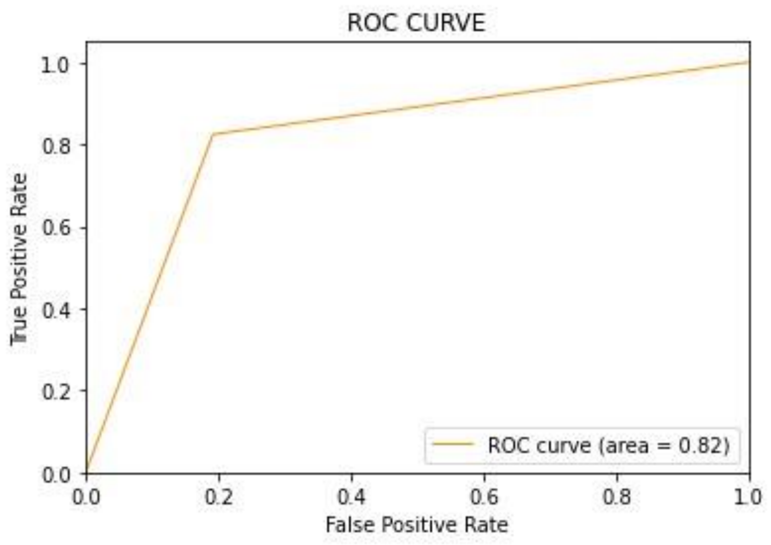
ROC curve of logistic regression classifier



```
              precision    recall  f1-score   support

           0       0.82      0.81      0.81     40100
           1       0.81      0.82      0.82     39900

    accuracy                           0.82     80000
   macro avg       0.82      0.82      0.82     80000
weighted avg       0.82      0.82      0.82     80000
```



SVM classifier result with accuracy and confusion matrix



ROC curve of SVM classifier

# CHAPTER 7: WORK TO BE CARRIED OUT IN PHASE 2

Aim is to build web-based application which can fetch real time tweets from twitter and do the following:

1) Application can fetch tweets by selected topics using Twitter API.
2) Application can store fetched tweets into database.
3) Application can perform pre-processing that consists of 5 steps described before.
4) Application can perform text feature extraction to tweet by converting tweet into set of features in real number form inside a vector which will be used as input.
5) Application can be used to give label (topic) to each tweet used as training data.
6) Application can learn through training process conducted using training data.
7) Application can perform evaluation process to measure accuracy rate of the trained classifier.

## REFERENCES

1) https://www.researchgate.net/publication/314667612_Using_logistic_regression_method_to_classify_tweets_into_the_selected_topics
2) https://www.geeksforgeeks.org/data-visualization-with-python/
3) https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-4-count-vectorizer-b3f4944e51b5
4) KrommydaM.;RigosA.;BouklasK.and AmditisA.;"An Experimental Analysis of Data Annotation Methodologies for Emotion Detection in Short Text Posted on Social Media".Informatics, 8(1), 19, 2021.
5) MandaK. R.;"Sentiment Analysis of Twitter Data Using Machine Learning and Deep Learning Methods".(June), 2019.
6) https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/
7)