SENTIMENT ANALYSIS OF TEXTUAL AND TWITTER DATA

Contents

Abstract:

In today's world, digital networks play an essential role in information exchange and sharing opinions. People’s psychological impact plays a significant impact in their day-to-day life.

Sentiment Analysis is a method of examining a person's views and polarities of thinking. Twitter is a popular medium for expressing ideas, beliefs, and sentiments on a variety of topics. Sentimental Analysis on Twitter is a technique for assessing the views expressed in tweets (messages posted by the user on Twitter).
Twitter posts are useful for obtaining the user's emotional value. Polarity is indicated by the data's positive, negative, or unbiased numbers. It concentrates on a person's tweets and hashtags to comprehend the scenarios in each condition.

It is the use of content investigation tools to comprehend and organize feelings in relation to content information. It is based on data acquired from student responses to internet-based life debates. Massive amounts of content information are created, but they are difficult to study, interpret, and monotonous. The sentimental analysis aids in the reduction of massive unstructured data to a fundamental and justifiable level. Sentimental analysis is used in a variety of situations, most notably in business. Sentiment analysis can be used in a variety of ways.

Analyzing a topic's social media discussion, assessing survey replies, and determining if product evaluations are positive or negative are all examples of what we do.

Artificial intelligence's current study focus is sentiment analysis. It is one of the most essential decision-making sources, and it may be extracted, identified, and analyzed through online feelings reviews. The basic purpose is to connect on Twitter and search for tweets that contain a specific term, after which the orientation of the tweets is assessed as positive or negative. The buzzwords for this paper were gathered from Twitter by using Twitter API. On the premise of feature selection of reward words, the opinions of online tweets are assessed. We applied some classifiers for testing and training the attributes, as well as analyzing the sentimental aspect, in order to identify the optimal features. Python is used to implement the proposed system.

It's a Natural Language Processing and Data Extraction task that analyses provided words or tweets to extract user feelings reflected in positive or negative remarks. In general, sentiment analysis is concerned with determining a speaker's or writer's attitude toward a topic or the overall tone of textual material. The exponential increase in Internet usage and public opinion interchange over the last few years is the driving force behind Sentiment Analysis today.

Introduction:

Sentimental Analysis is defined as analyzing people's perceptions of their own reality. People are now accustomed to reviewing product comments and posts, which are referred to as the user's view, expression, mood, mentality, beliefs, or actions. Sentimental Analysis is a technique for determining how emotions are portrayed in texts.

In sentiment analysis, there are numerous assertions. The first is that a position that is considered favourable in one circumstance may be considered negative in another. The next claim is that most people do not consider their points of view in the same way. Almost all evaluations include both positive and negative comments, which can be deciphered by reading the sentences one by one. Discovering and tracking opinion sites on the internet is a challenge. As a result, robotic cluster analysis and a summarization system will be required.

Sentiment analysis is the method of automatically detecting emotional or opinionated material in a text segment, as well as determining the text's polarity. The goal of Twitter sentiment categorization is to categorize a tweet's sentiment polarity as positive, negative, or neutral.

People's communication has been modernized thanks to social media. Social network data is helpful in analyzing user opinion, such as evaluating feedback on a newly released product, examining the response to a policy change, or assessing the excitement of an ongoing event. Sifting through this data by hand is time-consuming and extremely costly.

Sentiment Analysis is a method of assessing people's sentiments, thoughts, and attitudes concerning a certain product, organization, or service. It is also known as sentiment classification, which is a type of text classification job in which you are given a phrase or a series of words and asked to determine whether the sentiment behind them is positive, negative, or neutral. To maintain it as a binary class task, the third property is sometimes ignored.

Hardware / Software Requirements:

For the implementation of Twitter textual analysis, the software requirements we are using are

- Windows
- Modern Web Browser
- Twitter API, Google API

We are implementing the python code in collab research scholar and we are taking the random data set to implement with different algorithms.

3. Existing System/approach/method:

In existing approach, they only classified separately each for the traditional machine learning algorithms like Naïve Bayes, Maximum Entropy, SVM etc. The input, which consists of Twitter tweets, is delivered to the aspect expression extractor in the proposed system. This extracts aspect and expression pairs from the provided reviews. Using the resulting pairs, the sentiment pattern extractor constructs the whole potential morphological sentence pattern.
These potential patterns are fed into a convolutional neural network that has been pre-trained on an existing corpus. The sentiment polarity of the input review is the output.

Drawback/limitations of existing System/approach/method:

The main drawbacks/limitations of the existing model are:

- It will take a lot of time to train a data set.
- We can get better accuracy by comparing with other algorithms.
- We can increase the F1 score.
- It is possible to decrease the training time.

Proposed/Developed Model

We are using more different machine learning classifiers to test the data set that we have and we are comparing them with each other and finding which will have the best accuracy among them. There is a training and testing phase in the method that we implemented and differentiating the positive and negative sentiments and we will get a comparison using graphs that are implemented. The Tweets are taken from the Twitter that is being taken in the data set.

Module Wise Description:

Count Vectorizer:

To transform a group of text documents into a vector of term/token counts, use Count Vectorizer. It also allows text data to be pre-processed before being converted into a vector representation. Because of this, it's a text feature representation module with a lot of flexibility. Count Vectorizer tokenizes tokenization as the process of breaking down a sentence, paragraph, or any text into words. the text, as well as performing very basic preparation such as removing punctuation marks and converting all words to lowercase, etc.

Standard Scaler:

The mean is removed and each feature/variable is scaled to unit variance using Standard Scaler. This operation is carried out in a feature-by-feature manner. Because it includes estimating the empirical mean and standard deviation of each feature, Standard Scaler can be influenced by outliers (assuming they exist in the dataset). It converts the data so that the mean is 0 and the standard deviation is 1. In a nutshell, it standardizes data. Standardization is beneficial for data with negative values. It arranges the data according to a conventional normal distribution. It's better for classification than regression.

Random Forest Classifier:

A random forest is a Meta estimator that employs averaging to increase predicted accuracy and control over-fitting by fitting a number of decision tree classifiers on different sub-samples of the dataset. If bootstrap=True (the default), the sub-sample size is controlled by the max samples argument; otherwise, the entire dataset is utilized to create each tree. The decision tree is the fundamental unit of random forest classifiers. The decision tree is a hierarchical structure constructed from the features (or independent variables) of a data collection. The decision tree is divided into nodes based on a measure connected with a subset of the features. The random forest is a set of decision trees associated with a set of bootstrap samples created from the original data set. The nodes are divided according to the entropy (or Gini index) of a subset of the characteristics. The subsets formed by bootstrapping from the original data set are the same size as the original data set.

Logistic Regression:

Logistic regression is a statistical model that uses a logistic function to represent a binary dependent variable in its most basic form, however, many more advanced extensions exist. Logistic regression (or logit regression) in regression analysis is used to estimate the parameters of a logistic model (a form of binary regression). A binary logistic model, mathematically, contains a dependent variable with two possible values, such as pass/fail, which is represented by an indicator variable, with the two values labelled "0" and "1." The log-chances (logarithm of the odds) for the value labelled "1" in the logistic model is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The associated likelihood of the value labelled "1" might range between 0 (surely the value "0") and 1 (definitely the value "1"), therefore the labelling; the logistic function, hence the name, transforms log-odds to probability. The logit, from the logistic unit, is the unit of measurement for the log-odds scale, hence the alternative names. Analogous models, such as the probit model, can be used instead of the logistic function; the defining feature of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable, this generalizes the odds ratio.

Decision Tree Classifier:

In the shape of a tree structure, the decision tree constructs classification or regression models. It incrementally divides a dataset into smaller and smaller sections while also developing an associated decision tree. The end result is a tree containing leaf nodes and decision nodes. A decision node (for example, Outlook) may have two or more branches (e.g., Sunny, Overcast and Rainy). A classification or judgement is represented by a leaf node (for example, Play). The root node is the topmost decision node in a tree that corresponds to the best predictor.
Both category and numerical data can be handled by decision trees. The purpose of this algorithm is to develop a model that predicts the value of a target variable, for which the decision tree employs the tree representation to solve the problem, where the leaf node corresponds to a class label and attributes are represented on the tree's internal node.

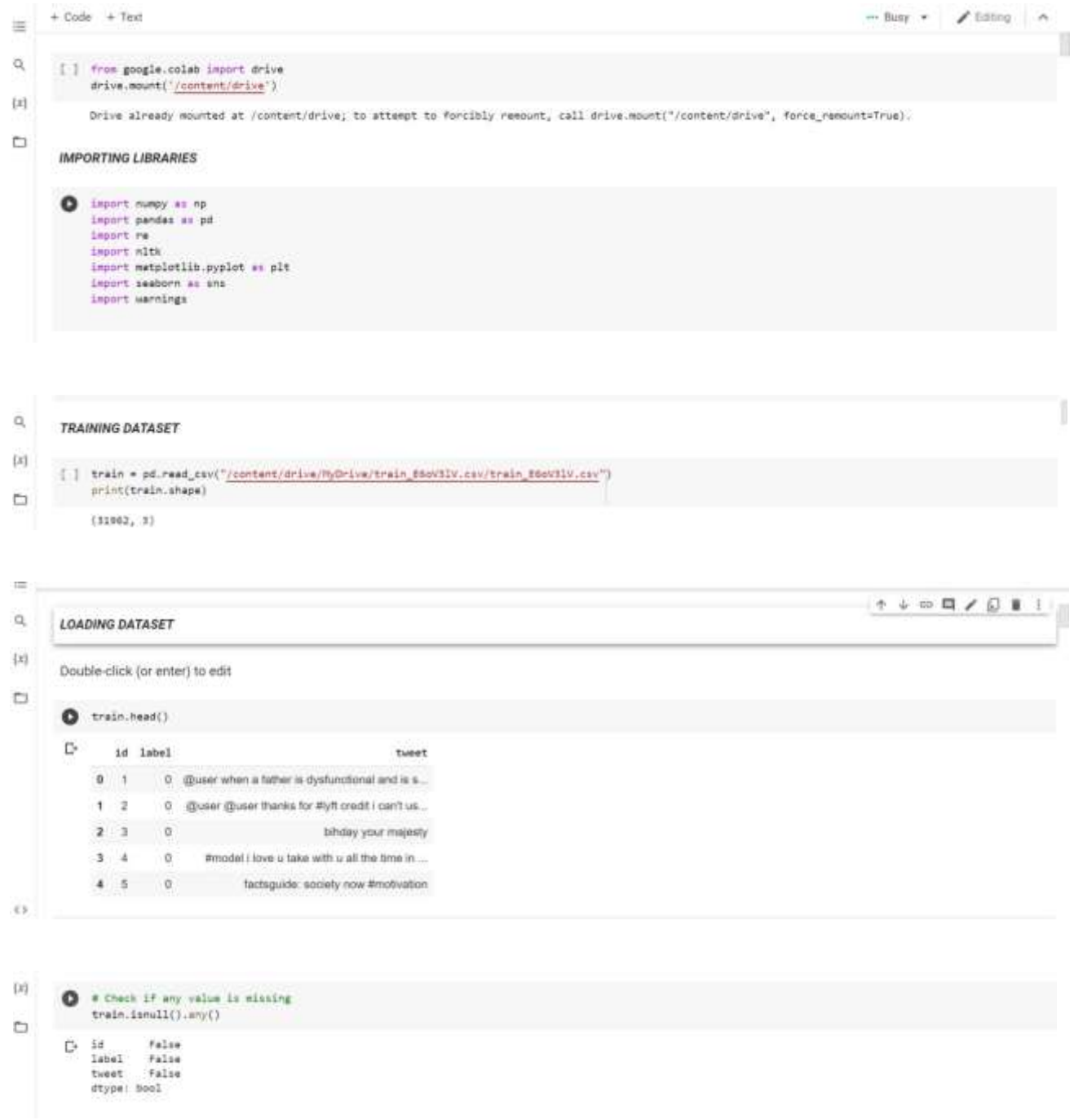
XGB Classifier:

XGboost is the most extensively used algorithm in machine learning, regardless of whether the task is classification or regression. It is well-known for outperforming all other machine learning algorithms. XGBoost is a newly dominant technique in applied machine learning and Kaggle contests for structured or tabular data. XGBoost is a gradient-boosted decision tree implementation optimized for speed and performance.

Support Vector Classifier:

A support vector machine (SVM) is a supervised machine learning model that solves two-group classification problems using classification techniques. They may categorize new text after providing an SVM model with sets of labelled training data for each category. SVM can be applied to both linear and nonlinear situations for classification. They were the greatest general-purpose algorithm for machine learning until 2006.

Implementation:



```
+ Code  + Text                                                    Busy   Editing

[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

TRAINING DATASET

```
[ ] train = pd.read_csv("/content/drive/MyDrive/train_E6oV3lV.csv/train_E6oV3lV.csv")
    print(train.shape)

    (31962, 3)
```

LOADING DATASET

Double-click (or enter) to edit

```
train.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

```
# Check if any value is missing
train.isnull().any()

id       False
label    False
tweet    False
dtype: bool
```

Checking out the negative comments from the train set

Double-click (or enter) to edit

```
# using head to print first 10 rows
train[train['label'] == 0].head(10)
```

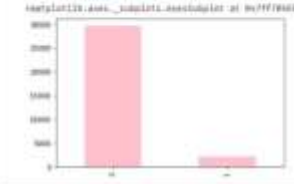
	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
5	6	0	[2/2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...
7	8	0	the next school year is the year for exams.ð□□...
8	9	0	we won!!! love the land!!! #allin #cavs #champ...
9	10	0	@user @user welcome here ! i'm it's so #gr...

```
# checking out the positive comments from the train sets
train[train['label'] == 1].head(10)
```

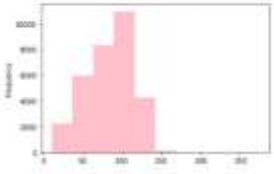
	id	label	tweet
13	14	1	@user #cnn calls #michigan middle school 'bull...
14	15	1	no comment! in #australia #opkillingbay #se...
17	18	1	retweet if you agree!
23	24	1	@user @user lumpy says i am a . prove it lumpy.
34	35	1	it's unbelievable that in the 21st century we'...
56	57	1	@user lets fight against #love #peace
68	69	1	ð□□the white establishment can't have blk fol...
77	78	1	@user hey, white people: you can call people '...
82	83	1	how the #altright uses & insecurity to su...
111	112	1	@user i'm not interested in a #linguistics tha...

PREPROCESSING DATASET

```
# showing frequency of negative and positive tweets
train['label'].value_counts().plot.bar(color = 'pink', figsize = (6,4))
```



```
# checking the distribution of tweets in the data
length_train = train['tweet'].str.len().plot.hist(color = 'pink', figsize = (6,4))
```



```
# adding a column to represent the length of the tweet
train['len'] = train['tweet'].str.len()
train.head(10)
```

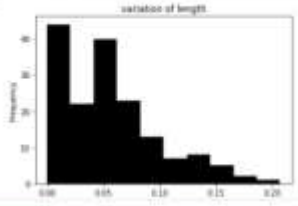
	id	label	tweet	len
0	1	0	@user when a father is dysfunctional and is s...	102
1	2	0	@user @user thanks for #lyft credit i can't us...	122
2	3	0	bihday your majesty	21
3	4	0	#model i love u take with u all the time in ...	86
4	5	0	factsguide: society now #motivation	39
5	6	0	[2/2] huge fan fare and big talking before the...	116
6	7	0	@user camping tomorrow @user @user @user @use...	74
7	8	0	the next school year is the year for exams.ð...	143
8	9	0	we won!!! love the land!!! #allin #cavs #champ...	87
9	10	0	@user @user welcome here ! i'm it's so #gr...	50

```
# Group wise Distribution of Dataset
train.groupby('label').describe()
```

	id								len							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
label																
0	29720.0	15814.454441	9223.783409	1.0	7961.75	15871.5	23865.25	31962.0	29720.0	84.328934	29.569484	11.0	62.0	88.0	107.0	274.0
1	2242.0	16074.886975	9267.955758	14.0	8075.25	16086.0	24022.00	31961.0	2242.0	90.187779	27.375502	12.0	69.0	96.0	111.0	152.0

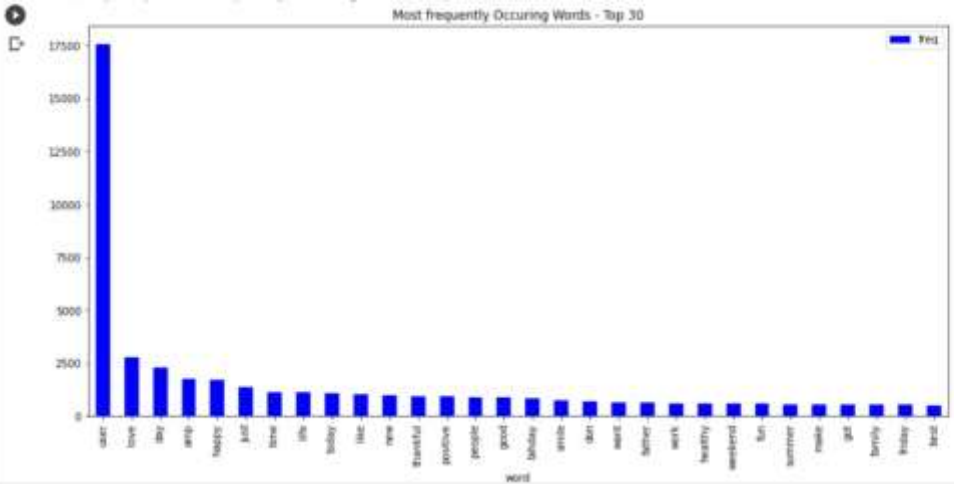
Double-click (or enter) to edit

```
# Graphical representation of Dataset
train.groupby('len').mean()['label'].plot.hist(color = 'black',figsize = (6,4),)
plt.title('variation of length')
plt.xlabel('length')
plt.show()
```




```
# Checking most frequently used words and Graphical Representation
from pandas.core.frame import DataFrame
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(train.tweet)
sum_words = words.sum(axis=0)
words_freq = [(word, sum_words[0,i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1],reverse = True)
frequency = pd.DataFrame(words_freq, columns=['word','freq'])
frequency.head(30).plot(x='word', y ='freq', kind= 'bar', figsize=(15,7), color = 'blue')
plt.title("Most frequently Occuring Words - Top 30")
```

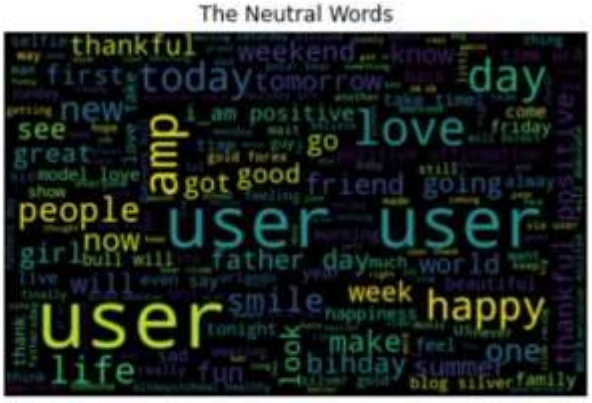
```
Text(0.5, 1.0, 'Most frequently Occuring Words - Top 30')
```



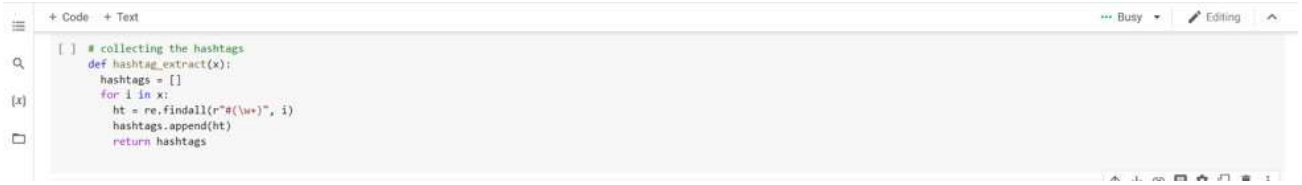
```
# Using wordcloud to represent frequency of words
from wordcloud import WordCloud
wordcloud = WordCloud(background_color = 'white', width = 1000, height = 1000).generate_from_frequencies(dict(words_freq))
plt.figure(figsize=(10,8))
plt.imshow(wordcloud)
plt.title("WordCloud - Vocabulary from Reviews", fontsize =22)
```

```
Text(0.5, 1.0, 'WordCloud - Vocabulary from Reviews')
```

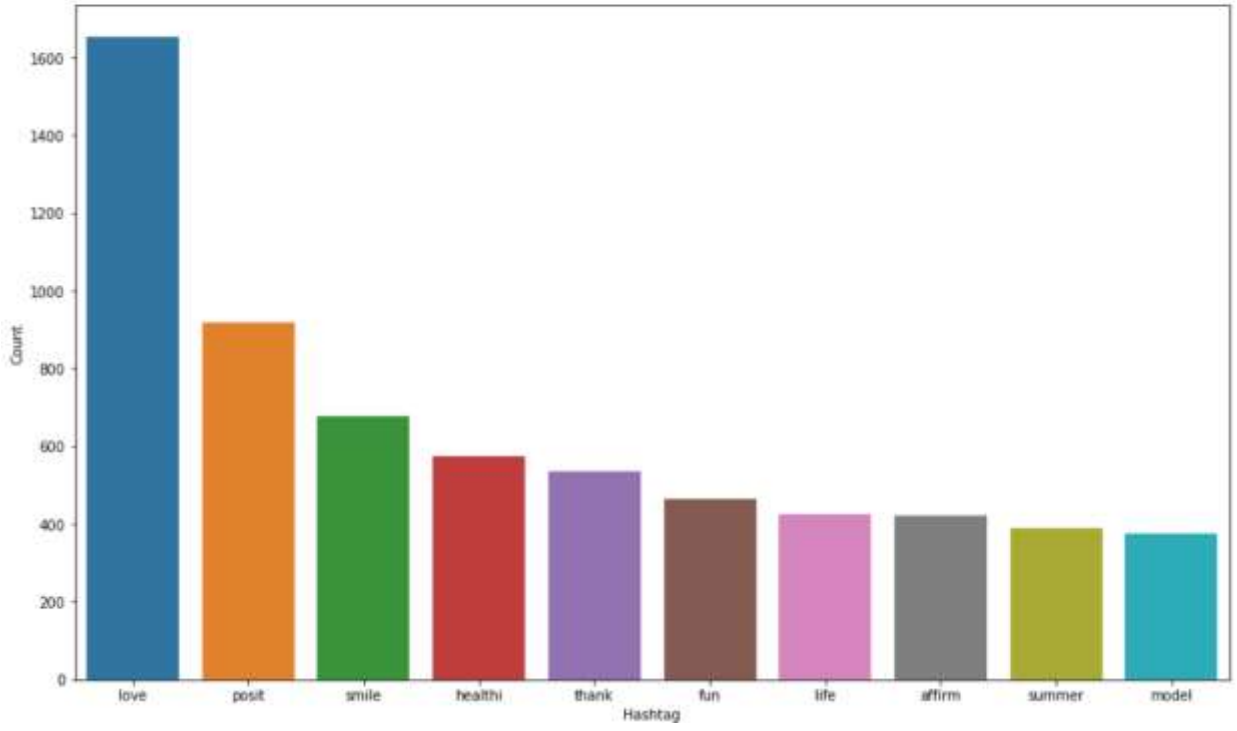

```
# Using wordcloud to represent frequency of Neutral words
normal_words =' '.join([text for text in train['tweet'][train['label']==0]])
wordcloud =WordCloud(width=800, height=500, random_state = 0,max_font_size=110).generate(normal_words)
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis('off')
plt.title('The Neutral Words')
plt.show()
```



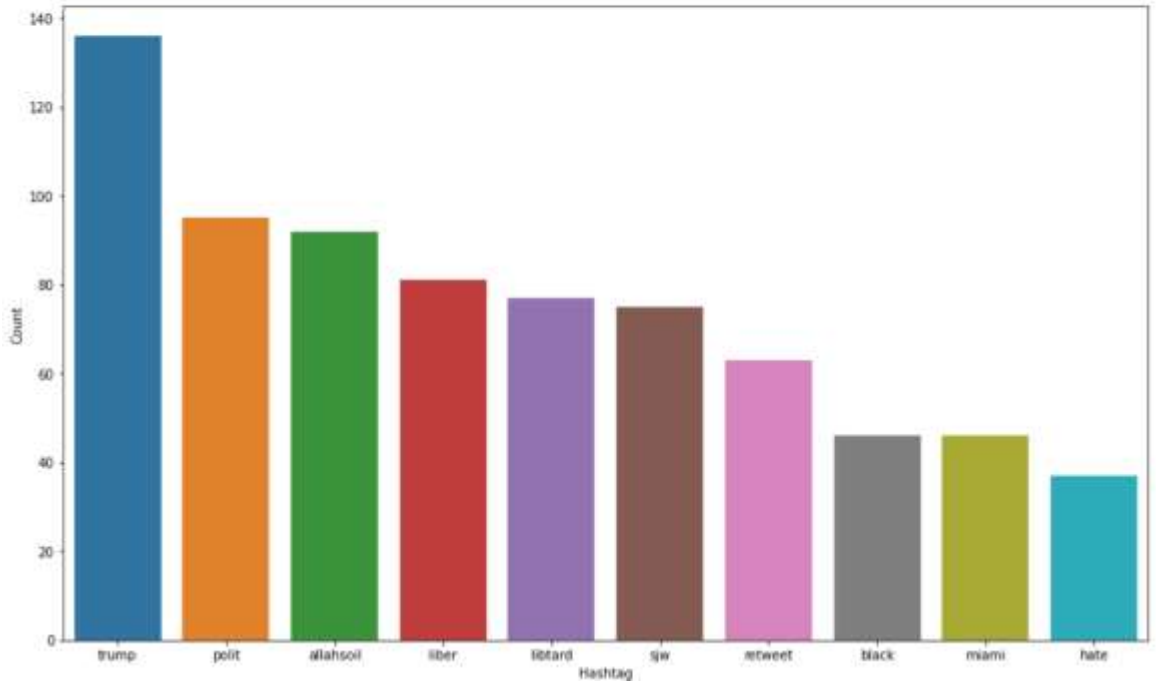

```
# Using wordcloud to represent frequency of Negative words
negative_words = ' '.join([text for text in train['tweet'][train['label']==1]])
wordcloud = WordCloud(background_color='cyan', width=800, height=500, random_state = 0,max_font_size =110).generate(negative_words)
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis('off')
plt.title('the Negative Words')
plt.show()
```




```
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
plt.show()
```



```
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
plt.show()
```




```
# tokenizing the words present in the training set
tokenized_tweet = train['tweet'].apply(lambda x: x.split())

# importing gensim
import gensim

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

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

```
(6109193, 8811580)
```

```
[ ] from tqdm import tqdm
    tqdm.pandas(desc="progress-bar")
    from gensim.models.doc2vec import LabeledSentence

[ ] def add_label(twt):
      output = []
      for i, s in zip(twt.index, twt):
        output.append(LabeledSentence(s, ["tweet_" + str(i)]))
      return output

      #label all the tweets
      labeled_tweets = add_label(tokenized_tweet)

      labeled_tweets[:6]
```

```
#removing unwanted patterns from the data

import re
import nltk

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
train_corpus = []

for i in range(0, 31962):
  review = re.sub('[^a-zA-Z]', ' ', train['tweet'][i])
  review = review.lower()
  review = review.split()

  ps = PorterStemmer()

  #stemming
  review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]

  # joining them back with space
  review = ' '.join(review)
  train_corpus.append(review)

  #creating bag of words

  from sklearn.feature_extraction.text import CountVectorizer

  cv = CountVectorizer(max_features = 2500)
  x = cv.fit_transform(train_corpus).toarray()
  y = train.iloc[:, 1]

  print(x.shape)
  print(y.shape)
```

```
(31962,)
(31950, 2500)
(31962,)
(31951, 2500)
(31962,)
(31952, 2500)
(31962,)
(31953, 2500)
(31962,)
(31954, 2500)
(31962,)
(31955, 2500)
(31962,)
(31956, 2500)
(31962,)
(31957, 2500)
(31962,)
(31958, 2500)
(31962,)
(31959, 2500)
(31962,)
(31960, 2500)
(31962,)
(31961, 2500)
(31962,)
(31962, 2500)
(31962,)
```

```
[16] # splitting the training data into train and valid sets

     from sklearn.model_selection import train_test_split

     x_train, x_valid, y_train, y_valid = train_test_split(x,y, test_size = 0.25, random_state = 42)

     print(x_train.shape)
     print(x_valid.shape)
     print(y_train.shape)
     print(y_valid.shape)
```

```
(23971, 2500)
(7991, 2500)
(23971,)
(7991,)
```

```
[17] # standardization

     from sklearn.preprocessing import StandardScaler

     sc = StandardScaler()

     x_train = sc.fit_transform(x_train)
     x_valid = sc.transform(x_valid)
```

```
[ ]  from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import confusion_matrix
     from sklearn.metrics import f1_score

     model = RandomForestClassifier()
     model.fit(x_train, y_train)

     y_pred = model.predict(x_valid)

     print("Training Accuracy :", model.score(x_train, y_train))
     print("Validation Accuracy :", model.score(x_valid, y_valid))

     # calculating the f1 score for the validation set
     print("F1 score :", f1_score(y_valid, y_pred))

     #confusion matrix
     cm = confusion_matrix(y_valid, y_pred)
     print(cm)
```

```
Training Accuracy : 0.9991656585040257
Validation Accuracy : 0.9538230509322988
F1 score : 0.6246185147507629
[[7315  117]
 [ 252  307]]
```

```
[ ] from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
    model.fit(x_train, y_train)

    y_pred = model.predict(x_valid)

    print("Training Accuracy :", model.score(x_train, y_train))
    print("Validation Accuracy :", model.score(x_valid, y_valid))

    # calculating the f1 score for the validation set
    print("F1 score :", f1_score(y_valid, y_pred))

    #confusion matrix
    cm = confusion_matrix(y_valid, y_pred)
    print(cm)
```

```
Training Accuracy : 0.9851487213716574
Validation Accuracy : 0.9416843949443123
F1 score : 0.5933682373472949
[[7185  247]
 [ 219  340]]
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
[ ] from sklearn.tree import DecisionTreeClassifier
    model = DecisionTreeClassifier()
    model.fit(x_train, y_train)

    y_pred = model.predict(x_valid)

    print("Training Accuracy :", model.score(x_train, y_train))
    print("Validation Accuracy :", model.score(x_valid, y_valid))

    # calculating the f1 score for the validation set
    print("F1 score :", f1_score(y_valid, y_pred))

    #confusion matrix
    cm = confusion_matrix(y_valid, y_pred)
    print(cm)
```

```
Training Accuracy : 0.9991656585040257
Validation Accuracy : 0.931673132273808
F1 score : 0.5357142857142857
[[7130  302]
 [ 244  315]]
```

```
[ ] from xgboost import XGBClassifier
    model = XGBClassifier()
    model.fit(x_train, y_train)

    y_pred = model.predict(x_valid)

    print("Training Accuracy :", model.score(x_train, y_train))
    print("Validation Accuracy :", model.score(x_valid, y_valid))

    # calculating the f1 score for the validation set
    print("F1 score :", f1_score(y_valid, y_pred))

    #confusion matrix
    cm = confusion_matrix(y_valid, y_pred)
    print(cm)
```

```
Training Accuracy : 0.9445997246673064
Validation Accuracy : 0.9433112251282693
F1 score : 0.35378031383737524
[[7414   18]
 [ 435  124]]
```

```
[ ] from sklearn.svm import SVC
    model = SVC()
    model.fit(x_train, y_train)

    y_pred = model.predict(x_valid)

    print("Training Accuracy :", model.score(x_train, y_train))
    print("Validation Accuracy :", model.score(x_valid, y_valid))

    # calculating the f1 score for the validation set
    print("F1 score :", f1_score(y_valid, y_pred))

    #confusion matrix
    cm = confusion_matrix(y_valid, y_pred)
    print(cm)
```

```
Training Accuracy : 0.978181969880272
Validation Accuracy : 0.9521962207483419
F1 score : 0.4986876640419947
[[7419   13]
 [ 369  190]]
```

Results and Discussion

In this report, we have developed a model to analyze the sentiment of Twitter data which allows the processing of the data and training of the dataset according to the sentiment and classifies its polarity and sentiment using different classifiers like Decision Tree, Random Forest, and others. The Random Forest are found to outperform the traditional systems in most cases, thereby establishing their utility in the field of natural language processing, including sentiment analysis. Through these, we can get the excellent accuracy and the best of all was given by Random Forest. Our built classifier can be utilized and used as a data analysis tool in NLTK.
Therefore, overall, we can propose our technique to analyze sentiment in any device, public figure or issue than another existing model.

Name	Training Accuracy	Validity Accuracy	F1 Score
Random Forest Classifier	0.999	0.954	0.625
Logistic Regression	0.985	0.941	0.593
Decision Tree Classifier	0.999	0.932	0.536
XGB Classifier	0.945	0.943	0.353
Support Vector Classifier	0.978	0.952	0.498

Conclusion

Through this model, we have presented our approach for sentiment analysis of Twitter data and we have tried different classifiers where some of which like RandomForest gave excellent results. For training of our data, it took a lot of time which can be reduced to a certain amount. An individual's sentiment toward a certain might be influenced by several factors like mood, result, etc. Throughout time the sentiment toward a particular opinion might change as people can see this through different approaches over time. We can increase the accuracy and train the data set to predict the sentiment over changing time and opinion.

References

- General Implementation and Debugging help: https://www.stackoverflow.com
- V. Sahayak, V. Shete, and A. Pathan, "Sentiment analysis on Twitter data," International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 2, no. 1, pp. 178–183, 2015.
- F. Ceci, A. L. Goncalves, and R. Weber, "A model for sentiment analysis based on ontology and cases," IEEE Latin America Transactions, vol. 14, no. 11, pp. 4560–4566, 2016.
- B. Liu, Sentiment analysis: mining opinions, sentiments, and emotions. Cambridge University Press, 2015.
- J. Serrano-Guerrero, J. A. Olivas, F. P. Romero, and E. Herrera- Viedma, "Sentiment analysis: A review and comparative analysis of web services," Information Sciences, vol. 311, pp. 18–38, 2015.
- E. Cambria, "Affective computing and sentiment analysis," IEEE Intelligent Systems, vol. 31, no. 2, pp. 102–107, 2016.
- R. Upadhyay and A. Fujii, "Semantic knowledge extraction from research documents," in Computer Science and Information Systems(FedCSIS), 2016 Federated Conference on. IEEE, 2016, pp. 439–445.
- N. P. M. Vadivukarassi and P. Aruna, "Sentimental analysis of tweets using naive Bayes algorithm," World Applied Sciences Journal, vol. 35, no. 1, pp. 54–59, 2017.
- S. Pal and S. Ghosh, "Sentiment analysis using averaged histogram," International Journal of Computer Applications, vol. 162, no. 12, 2017.
- V. Kharde, P. Sonawane et al., "Sentiment analysis of Twitter data: a survey of techniques," 2016.
- R. Hasan, "Sentiment analysis with NLP on Twitter data," Jun 2019.[Online]. Available: https://github.com/Rakib-Hasan031/Sentiment-Analysis-with-NLP-on-Twitter-Data