TWEET SENTIMENTAL ANALYSIS

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

Shivam Pahariya [RA2011027010007]
Pratyush Vats [RA2011027010018]
Shivansh Sharma [RA2011027010039]

Under the guidance of

Dr. M. Mercy Theresa

Assistant Professor, Department of DSBS

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"Tweet Sentimental Analysis"** is the bona fide work of **Shivam Pahariya(RA2011027010007) Pratyush Vats(RA2011027010018) Shivansh Sharma (RA20110270100)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.M. Mercy Theresa
GUIDE
Assistant Professor
Department of DSBS

SIGNATURE

Dr. M. Lakshmi
HEAD OF THE DEPARTMENT
Professor & Head
Department of DSBS

ABSTRACT

In the field of social media data analytics, one popular area of research is the sentiment analysis of Twitter data. Twitter is one of the most popular social media platforms in the world, with 330 million monthly active users and 500 million tweets sent each day. By carefully analyzing the sentiment of these tweets — whether they are positive, negative, or neutral, for example — we can learn a lot about how people feel about certain topics.

Understanding the sentiment of tweets is important for a variety of reasons: business marketing, politics, public behavior analysis, and information gathering are just a few examples. Sentiment analysis of Twitter data can help marketers understand the customer response to product launches and marketing campaigns, and it can also help political parties understand the public response to policy changes or announcements.

The tweet data was processed through various NLP procedures and then it was trained on different ML algorithms (Random Forrest, Logistic Regression and Decision Tree). Random Forrest Classifier gave the best accuracy at 95.15%.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

NLP	Natural Language Processing
NLTK	Natural Language Toolkit
DT	Decision Tree
F1	Harmonic Mean of Precision and Recall
IDE	Integrated Development Environment

1. INTRODUCTION

Twitter sentiment analysis analyzes the sentiment or emotion of tweets. It uses natural language processing and machine learning algorithms to classify tweets automatically as positive, negative, or neutral based on their content. It can be done for individual tweets or a larger dataset related to a particular topic or event.

1. Understanding Customer Feedback: By analyzing the sentiment of customer feedback, companies can identify areas where they need to improve their products or services.
2. Reputation Management: Sentiment analysis can help companies monitor their brand reputation online and quickly respond to negative comments or reviews.
3. Political Analysis: Sentiment analysis can help political campaigns understand public opinion and tailor their messaging accordingly.
4. Crisis Management: In the event of a crisis, sentiment analysis can help organizations monitor social media and news outlets for negative sentiment and respond appropriately.
5. Marketing Research: Sentiment analysis can help marketers understand consumer behavior and preferences, and develop targeted advertising campaigns.

We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover, the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm.

OBJECTIVE:

1. To Implement an algorithm for automatic classification of text into positive, negative or neutral.
2. Sentimental analysis to determine the attitude of public towards a subject of interest.
3. Graphical Representation of the Analysis

2. LITERATURE SURVEY

2.1 Fundamentals of Sentiment Analysis and Its Applications

Author: Mohsen Farhadloo , Erik Rolland

Published on Research Gate

The problem of identifying people's opinions expressed in written language is a relatively new and very active field of research. Having access to huge amount of data due to the ubiquity of Internet, has enabled researchers in different fields—such as natural language processing, machine learning and data mining , text mining , management and marketing and even psychology—to conduct research in order to discover people's opinions and sentiments from the publicly available data sources. Sentiment analysis and opinion mining are typically done at various level of abstraction: document, sentence and aspect. Recently researchers are also investigating concept-level sentiment analysis , which is a form of aspect-level sentiment analysis in which aspects can be multi terms. Also recently research has started addressing sentiment analysis and opinion mining by using, modifying and extending topic modeling techniques. Topic models are probabilistic techniques for discovering the main themes existing in a collection of unstructured documents. In this book chapter we aim at addressing recent approaches to sentiment analysis, and explain this in the context of wider use. We start the chapter with a brief contextual introduction to the problem of sentiment analysis and opinion mining and extend our introduction with some of its applications in different domains. The main challenges in sentiment analysis and opinion mining are discussed, and different existing approaches to address these challenges are explained. Recent directions with respect to applying sentiment analysis and opinion mining are discussed. We will review these studies towards the end of this chapter, and conclude the chapter with new opportunities for research.

2.2 Fundamentals of Sentiment Analysis and Its Applications

Author: Mohsen Farhadloo , Erik Rolland

Published on Research Gate

The problem of identifying people's opinions expressed in written language is a relatively new and very active field of research. Having access to huge amount of data due to the ubiquity of Internet, has enabled researchers in different fields—such as natural language processing, machine learning and data mining , text mining , management and marketing and even psychology—to conduct research in order to discover people's opinions and sentiments from the publicly available data sources. Sentiment analysis and opinion mining are typically done at various level of abstraction: document, sentence and aspect. Recently researchers are also investigating concept-level sentiment analysis , which is a form of aspect-level sentiment analysis in which aspects can be multi terms. Also recently research has started addressing sentiment analysis and opinion mining by using, modifying and extending topic modeling techniques. Topic models are probabilistic techniques for discovering the main themes existing in a collection of unstructured documents. In this book chapter we aim at addressing recent approaches to sentiment analysis, and explain this in the context of wider use. We start the chapter with a brief contextual introduction to the problem of sentiment analysis and opinion mining and extend our introduction with some of its applications in different domains. The main challenges in sentiment analysis and opinion mining are discussed, and different existing approaches to address these challenges are explained. Recent directions with respect to applying sentiment analysis and opinion mining are discussed. We will review these studies towards the end of this chapter, and conclude the chapter with new opportunities for research.

2.3 Sentiment Analysis for Social Media

Author: Sachira Chinthana Jayasanka , Thilina Madhushani

Published on Research Gate

Sentiment analysis, the automated extraction of expressions of positive or negative attitudes from text has received considerable attention from researchers during the past decade. In addition, the popularity of internet users has been growing fast parallel to emerging technologies; that actively use online review sites, social networks and personal blogs to express their opinions. They harbor positive and negative attitudes about people, organizations, places, events, and ideas. The tools provided by natural language processing and machine learning along with other approaches to work with large volumes of text, makes it possible to begin extracting sentiments from social media. In this paper we discuss some of the challenges in sentiment extraction, some of the approaches that have been taken to address these challenges and our approach that analyses sentiments from Twitter social media which gives the output beyond just the polarity but use those polarities in product profiling, trend analysis and forecasting. Promising results has shown that the approach can be further developed to cater business environment needs through sentiment analysis in social media. People make judgments about the world around them when they are living in the society. They make positive and negative attitudes about people, products, places and events. These types of attitudes can be considered as sentiments. Sentiment analysis is the study of automated techniques for extracting sentiments from written languages. Growth of social media has resulted in an explosion of publicly available, user generated text on the World Wide Web.

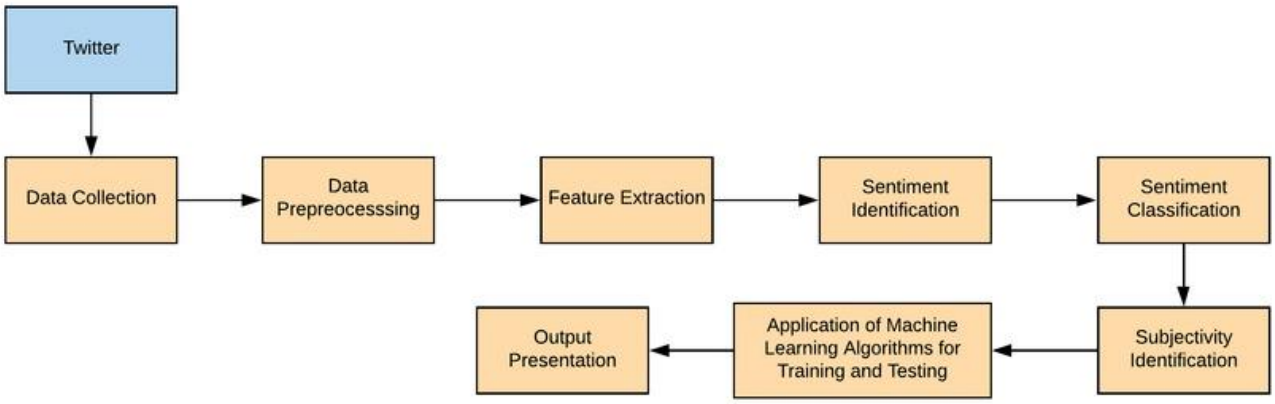
2.4 A Survey on Sentiment Analysis Algorithms and Datasets

Author: Reena Govindsingh Bhati

Published on Research Gate

Sentiment analysis is a subfield of natural language processing that aims to identify and extract subjective information from text. It has various applications in fields such as marketing, customer service, and politics. This survey focuses on the popular sentiment analysis algorithms and datasets. The algorithms can be classified into three categories: lexicon-based, machine learning-based, and hybrid. Lexicon-based approaches use pre-defined sentiment lexicons to determine the polarity of words in a text. Machine learning-based methods train models using labeled datasets to predict the sentiment of new texts. Hybrid methods combine the strengths of both lexicon-based and machine learning-based approaches. There are many datasets available for sentiment analysis, including movie reviews, product reviews, social media posts, and news articles. Some of the most commonly used datasets are the Stanford Sentiment Treebank, the IMDB movie reviews dataset, and the Amazon product reviews dataset. In conclusion, sentiment analysis is an important subfield of natural language processing with a wide range of applications. There are various algorithms and datasets available, and the choice of which to use depends on the specific task and the available resources.

3.SYSTEM ARCHITECTURE AND DESIGN



NLP (Natural Language Processing) techniques are used for processing and analyzing human language data, such as text and speech. The architecture of an NLP system can vary depending on the specific task and the techniques used. However, a typical NLP architecture can include the following components:

1. Data Collection and Pre-processing: The first step in any NLP system is to collect the relevant data and pre-process it. This includes tasks such as tokenization, normalization, stop word removal, and stemming, depending on the specific task and language.

2.Feature Extraction: Once the data is pre-processed, the next step is to extract features from it. This can include techniques such as bag-of-words, TF-IDF, and word embeddings, which convert the raw text into a numerical representation that can be processed by machine learning algorithms.

3.Machine Learning Models: The extracted features are then fed into machine learning models, such as decision trees, Naive Bayes, or neural networks, to perform the specific NLP task, such as sentiment analysis, named entity recognition, or machine translation.

4.Evaluation and Refinement: The performance of the NLP system is evaluated using metrics such as accuracy, precision, recall, and F1-score. The system can then be refined by adjusting the parameters of the machine learning models or by incorporating additional features or technique NLP (Natural Language Processing) techniques are used for processing and analyzing human language data, such as text and speech.

ML Algorithms used for Training and Classification of Data are:

Logistic Regression: This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds.

Random Forest: Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Decision trees: A decision tree is a popular supervised learning algorithm. It is a tree-like model that represents a sequence of decisions and their possible consequences. The decision tree algorithm works by recursively splitting the training data into subsets based on the feature values of the instances, and selecting the feature that provides the best split. The process continues until all instances in a branch belong to the same class, or until a stopping condition is met. n the case of hate speech recognition, the decision tree algorithm can be trained on a dataset of text samples labelled as either hate speech or not hate speech. The algorithm analyses the text samples and selects the most informative attributes (e.g., certain words or phrases) that can be used to classify the samples.

4.METHODOLOGY

Tweet Sentimental analysis is a complex task that requires a well-defined methodology in order to achieve accurate results. Various methods and techniques have been developed to identify and classify tweets, including machine learning algorithms, natural language processing techniques, and rule-based systems. The methodology for sentimental analysis involves several steps, such as data collection, data pre-processing, feature extraction, model training, and evaluation. Each step is crucial for the success of the analysis, and different approaches can be applied depending on the specific context and the type of tweets being targeted.

4.1 Algorithm Used for Sentimental Analysis

We decided to use Natural Language Processing to understand text and spoken words in much the same way human beings can. This was implemented on the dataset. The Decision Tree, Random Forrest Classifier, Logistic Regression algorithm were used to train and predict our model to further classify it into “Neutral Tweet” and “Negative tweet”.

4.1.1 Natural language Processing

Natural Language Processing or NLP (also called Computational Linguistics) can be defined as the automatic processing of human languages. As NLP is a large and multidisciplinary field, but comparatively a new area, there are many definitions out there practiced by different people. NLP can be a powerful tool for Tweet Sentimental Analysis.

NLP Techniques used in Tweet Sentimental Analysis are:

1.Tokenization:

Tokenization is the process of splitting a text document into individual units, called tokens. In Natural Language Processing (NLP), tokenization is a fundamental technique used to preprocess text data for further analysis. The goal of tokenization is to break down a text document into smaller, meaningful units that can be analyzed and processed by NLP algorithms. Tokenization is an important step in hate speech detection, as it helps pre-process the text data to make it suitable for analysis by machine learning algorithms. The goal of tokenization in hate speech detection is to break down the input text data into individual units, called tokens, which can be further analyzed for hate speech content.

2.Stemming:

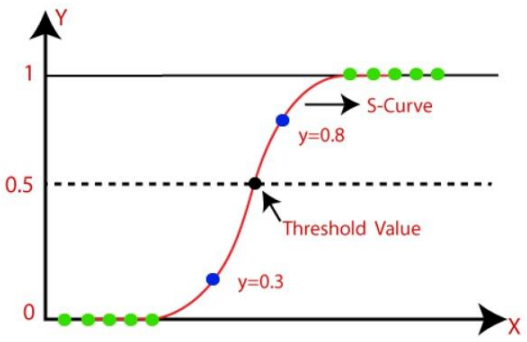
Stemming is the process of reducing a word to its base or root form, by removing any suffixes or prefixes that may be attached to it. The resulting base form is called the "stem" of the word. Stemming can be a useful pre-processing step in hate speech detection, as it can help to reduce the dimensionality of the text data and to group together different forms of the same word. By reducing the number of unique words in the text corpus, stemming can also help to improve the accuracy of hate speech detection models and to reduce overfitting.

3.Filtering Stop words:

Filtering stop words is a common pre-processing step in Natural Language Processing (NLP) that involves removing common words that do not carry much meaning in the text. These words are known as stop words, and they include words like "the", "and", "in", "of", "to", "a", and so on. The goal of filtering stop words is to reduce the dimensionality of the text data and to improve the accuracy of NLP models. Filtering stop words can be a useful pre-processing step in hate speech detection, as it can help to reduce the dimensionality of the text data and to focus on the more meaningful and informative words that are likely to be associated with hate speech. By removing stop words, the remaining words in the text may be more indicative of the underlying sentiment and intention of the text.

4.1.2 Machine Learning Algorithms used

1. Logistic Regression



Steps in Logistic Regression:

To implement the Logistic Regression using Python .Below are the steps:

A. Data Pre-processing step

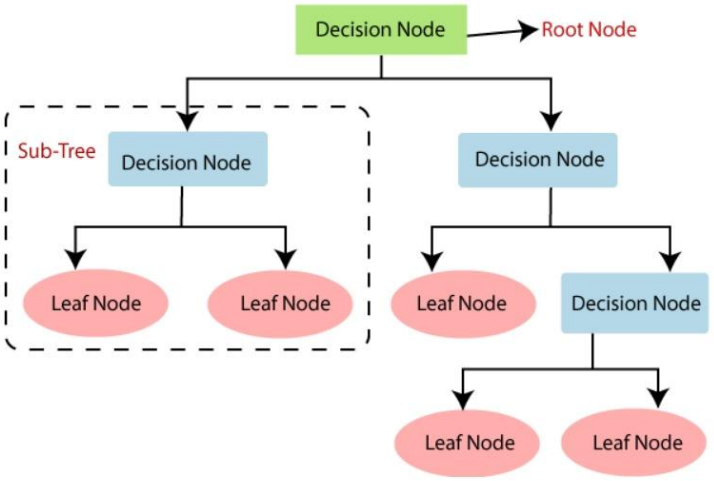
B. Fitting Logistic Regression to the Training set

C. Predicting the test result

D. Test accuracy of the result(Creation of Confusion matrix)

E. Visualizing the test set result.

2. Decision Tree



In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

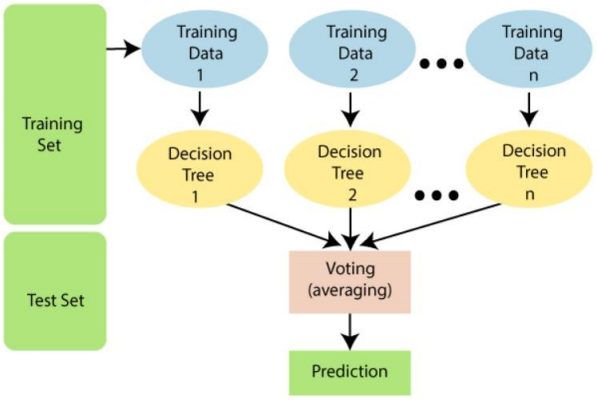
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

3. Random Forest Classifier



Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

4.2 Steps in Tweet Sentimental Analysis

Creating a Tweet Sentimental Analysis model using various ML Algos and NLP involves several steps. Here is a general methodology:

1. Data Collection: The first step is to collect a large dataset of labelled tweets and. This dataset can be sourced from public datasets or can be created by manually annotating text data.

2. Data Pre-processing: Once the dataset is collected, the next step is to pre-process the data. This includes removing noise, stop words, and punctuation marks. Data cleaning can be done using regular expressions and other text pre-processing techniques.

3. Feature Extraction: The next step is to extract relevant features from the text data. This can be done using Natural Language Processing (NLP) techniques such as bag of words, TF-IDF, and word embeddings. These techniques help to transform the text data into numerical representations that can be used as inputs for the decision tree model.

4. Model Training: The next step is to train ML model using the pre-processed data and extracted features. The model can be implemented using Python libraries such as scikit-learn.

5. Model Evaluation: Once the decision tree model is trained, the next step is to evaluate its performance. This can be done by using evaluation metrics such as accuracy, precision, recall, and F1-score.

6. Model Selection: After evaluating the model, the next step is to select the model with better performance.

7. Model Deployment: Once the model is optimized and selected, it can be deployed for use in real-world applications. This can be done using web APIs or other methods depending on the application.

5. CODING AND TESTING

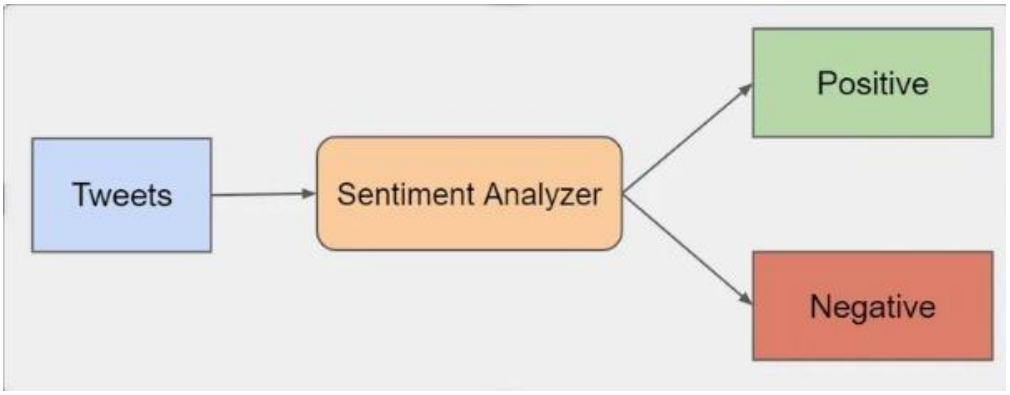
5.1 About the Dataset

The dataset provided is the Sentiment140 Dataset which consists of 31962 tweets that have been extracted using the Twitter API.

	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

The various columns present in this Twitter data are:

1. id : Unique Id of each tweet
2. label : '0' for Negative sentiment & '1' for Neutral Sentiment
3. tweet: Text tweet



5.2 Coding

Step-1: Import the Necessary Dependencies

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
```

```
import re
import nltk

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

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. NumPy is used for numerical operations. Count vectorizer is a method to convert text to numerical data. Train test split is used to split the dataset into training and testing parts

We then imported NLTK (The Natural Language Toolkit) library, used for symbolic and statistical natural language processing for English written in the Python programming language.

After importing all the necessary libraries, we load the datasets(train and test).

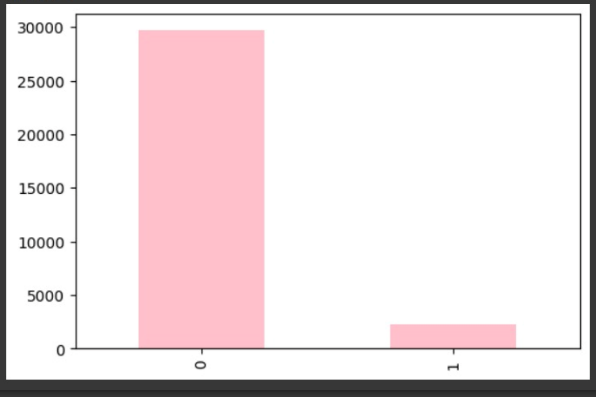
```
[ ] train = pd.read_csv('train_tweet.csv')
    test = pd.read_csv('test_tweets.csv')

    print(train.shape)
    print(test.shape)

    (31962, 3)
    (17197, 2)
```

Step-2: Exploratory Data Analysis

Checking the distribution of labels in the dataset

```
train['label'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

```
<Axes: >
```



Adding a column to represent the length of the tweet

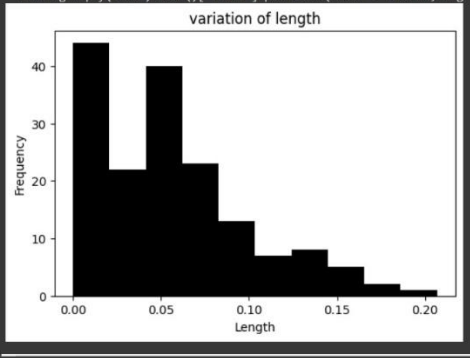
```
train['len'] = train['tweet'].str.len()
test['len'] = test['tweet'].str.len()

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

Variation of Length in tweets

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

```
<ipython-input-13-250f85d53a70>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.
  train.groupby('len').mean()['label'].plot.hist(color = 'black', figsize = (6, 4),)
```

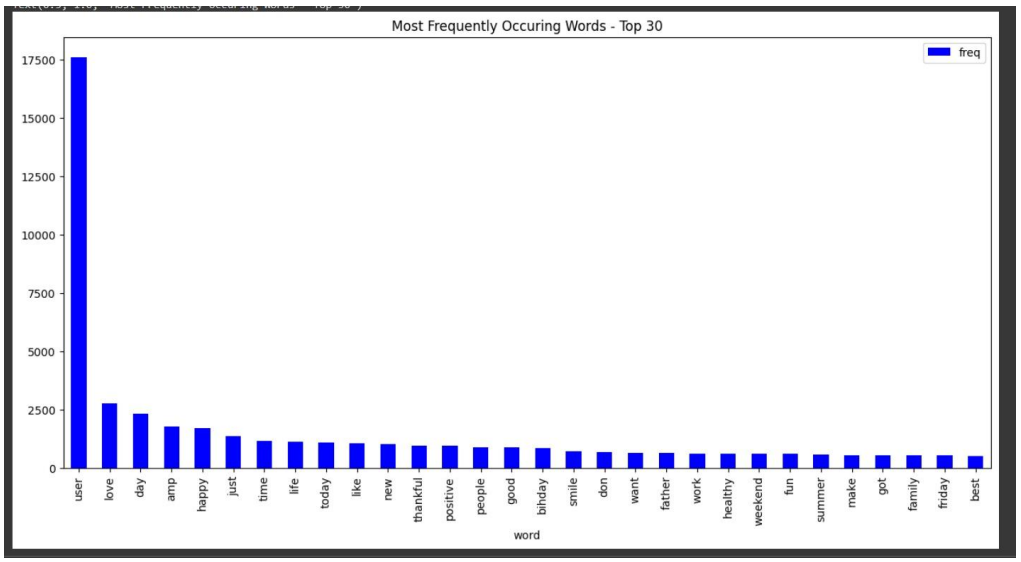


Step 3: Data Pre-Processing

In Data pre-processing, we prepare the raw data and make it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So for this, we use the data pre-processing task

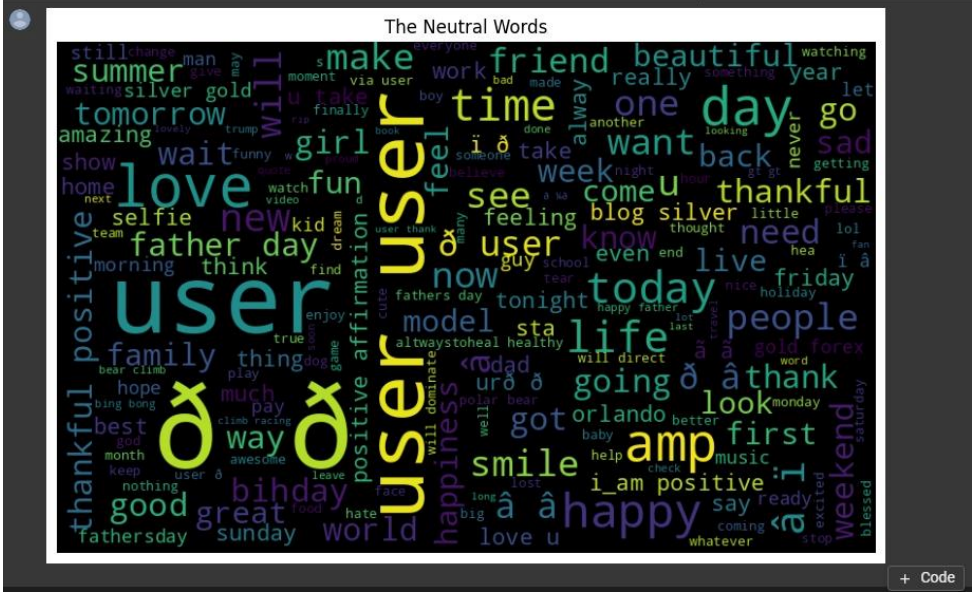
We have used two important Natural Language processing terms in the project, stop word and stemmer. Stop words are the useless words (data), in natural language processing. We can avoid those words from the input. Stemming is the process of producing morphological variants of a root word. We have to find the stem word for each text for better and easier prediction.

```
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")
```



WordCloud for 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.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Neutral Words')
plt.show()
```



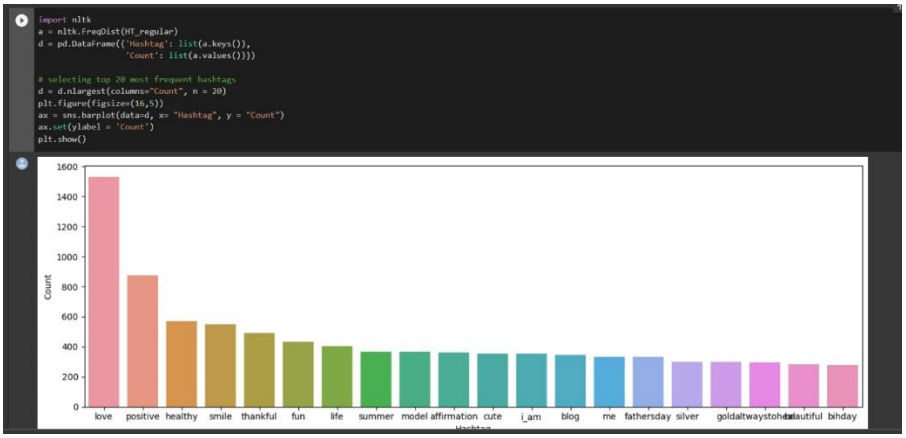
WordCloud For Negative Words

```
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.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Negative Words')
plt.show()
```

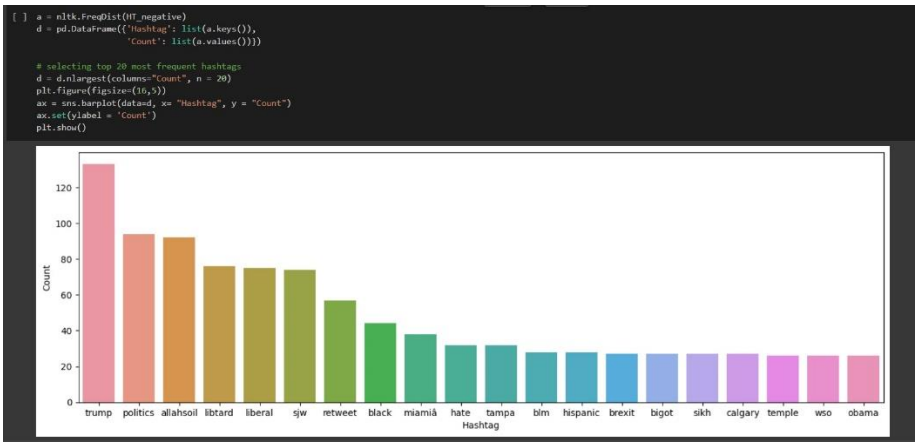
```
[ ]  # collecting the hashtags
     import re
     def hashtag_extract(x):
         hashtags = []

         for i in x:
             ht = re.findall(r"#(\w+)", i)
             hashtags.append(ht)

         return hashtags
```

```
[ ]  # extracting hashtags from non racist/sexist tweets
     HT_regular = hashtag_extract(train['tweet'][train['label'] == 0])

     # extracting hashtags from racist/sexist tweets
     HT_negative = hashtag_extract(train['tweet'][train['label'] == 1])

     # unnesting list
     HT_regular = sum(HT_regular,[])
     HT_negative = sum(HT_negative,[])
```

Top 20 Most Frequent hashtags in Neutral tweets



Top 20 Most Frequent hashtags in Negative tweets



Tokenization of Tweets

```
[ ]  # tokenizing the words present in the training set
     tokenized_tweet = train['tweet'].apply(lambda x: x.split())
     tokenized_tweet.head(5)

     0    [@user, when, a, father, is, dysfunctional, an...
     1    [@user, @user, thanks, for, #lyft, credit, i, ...
     2                               [bihday, your, majesty]
     3    [#model, i, love, u, take, with, u, all, the, ...
     4              [factsguide:, society, now, #motivation]
     Name: tweet, dtype: object
```

Stemming of dataset

```
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)
```

```
[ ]  train_corpus[:5]

['user father dysfunct selfish drag kid dysfunct run',
 'user user thank lyft credit use caus offer wheelchair van pdx disapoint getthank',
 'bihday majesti',
 'model love u take u time ur',
 'factsguid societi motiv']
```

Step 4: Splitting the Dataset

```
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)
```

Count vectorizer makes it easy for text data to be used directly in machine learning and deep learning models such as text classification. Each text input is preprocessed, tokenized, and represented as a sparse matrix.

```
[ ] 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,)
```

The data was split into approximately 75% training data and 25% testing data, using train_test_split module. The two columns considered for building the model are tweets and labels.

Standardization of Data

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

Step 5: Training and evaluating the accuracy of the model

LOGISTIC REGRESSION

```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import f1_score
    from sklearn.metrics import confusion_matrix

    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]]
```

DECISION TREE

```
[ ] 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 : 0.5388513513513512
[[7126  306]
 [ 240  319]]
```

RANDOM FOREST

```
[ ] 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 : 0.9515705168314353
F1 score : 0.60790273556231
[[7304  128]
 [ 259  300]]
```

As Random Forest Classifier gave the best performance, it is used for further classification of test data and unseen data for sentimental analysis.

6. RESULTS AND DISCUSSION

The Tweet Sentimental Analysis model was built and implemented successfully. The accuracy achieved by the model was around 95.1% percent, using the random forest classifier algorithm. Among various classification algorithms used for Tweet Sentimental Analysis, the random forest classification algorithm gave the best accuracy as compared to logistic regression, decision tree, support vector machine etc.

Upon evaluating all the models, we can conclude the following details i.e.

Accuracy: As far as the accuracy of the model is concerned, random forest performs better than Logistic regression, which in turn performs better than decision tree.

Training Accuracy:

RANDOM FOREST CLASSIFIER (99.1) ==DECISION TREE (99.1)>LOGISTIC REGRESSION (98.5)

Validation Accuracy:

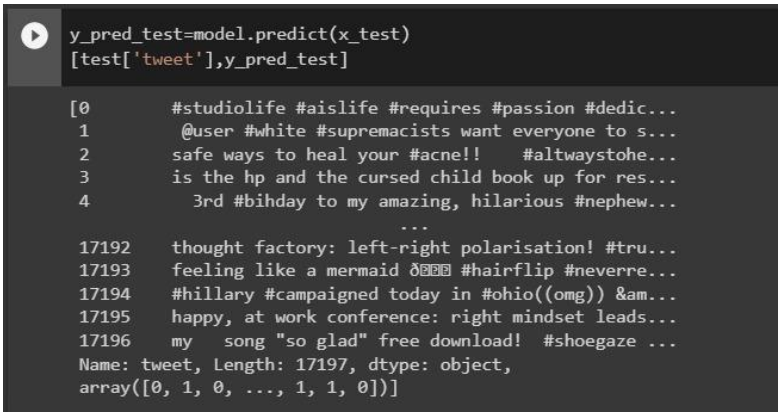
RANDOM FOREST CLASSIFIER (95.1)>LOGISTIC REGRESSION (94.1)>DECISION TREE (93.1)

F1 Scores:

RANDOM FOREST CLASSIFIER (60.7)>LOGISTIC REGRESSION (59.3)>DECISION TREE (53.8)

We, therefore, conclude that the RANDOM FOREST CLASSIFIER is the best model for the above-given dataset.

Model was then also used to predict sentiments of unlabeled test data.



```
y_pred_test=model.predict(x_test)
[test['tweet'],y_pred_test]
```

```
[0        #studiolife #aislife #requires #passion #dedic...
 1         @user #white #supremacists want everyone to s...
 2        safe ways to heal your #acne!!    #altwaystohe...
 3        is the hp and the cursed child book up for res...
 4          3rd #bihday to my amazing, hilarious #nephew...
                              ...
 17192    thought factory: left-right polarisation! #tru...
 17193    feeling like a mermaid δ□□□ #hairflip #neverre...
 17194    #hillary #campaigned today in #ohio((omg)) &am...
 17195    happy, at work conference: right mindset leads...
 17196    my   song "so glad" free download!  #shoegaze ...
Name: tweet, Length: 17197, dtype: object,
array([0, 1, 0, ..., 1, 1, 0])]
```

The model trained for Tweet Sentimental Analysis performs well on all kinds of inputs, thus proving to be a great scope for implementation with real-world scenarios. The use of Natural language processing (NLP) and its features helped get a clean and neat data set for training. This is so because using the stemming property of NLP we could stem the tweets leaving only the root words which makes it easier for the model to train on.

The monitoring of tweets on social media platforms is of critical importance to detect hate speech occurrences as soon as possible to prevent any further escalations which may result in violence or to know the impression of the audience about a certain product/person/political party. Our model provides an efficient and effective way to do so.

Finally, we conclude that our classification approach provides improvement inaccuracy by using even the simplest features and small amount of data set. However, there are still a number of things we would like to consider as future work which we mention in the next section.

7. CONCLUSION AND FUTURE ENHANCEMENTS

The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. So, we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance. Right now, we have worked with only the very simplest unigram models; we can improve those models by adding extra information like closeness of the word with a negation word. We could specify a window prior to the word (a window could for example be of 2 or 3 words) under consideration and the effect of negation may be incorporated into the model if it lies within that window. The closer the negation word is to the unigram word whose prior polarity is to be calculated, the more it should affect the polarity. For example, if the negation is right next to the word, it may simply reverse the polarity of that word and farther the negation is from the word the more minimized ifs effect should be.

One more feature we that is worth exploring is whether the information about relative position of word in a tweet has any effect on the performance of the classifier. Although Pang et al. explored a similar feature and reported negative results, their results were based on reviews which are very different from tweets and they worked on an extremely simple model.

One potential problem with our research is that the sizes of the classes are not equal. The problem with unequal classes is that the classifier tries to increase the overall accuracy of the system by increasing the accuracy of the majority class, even if that comes at the cost of a decrease in the accuracy of the minority classes. That is the very reason why we report significantly higher accuracies for negative class as compared to neutral class.

To overcome this problem and have the classifier exhibit no bias towards any of the classes, it is necessary to label more data (tweets) so that all classes are almost equal.

In this research we are focusing on general sentiment analysis. There is potential of work in the field of sentiment analysis with partially known context. For example, we noticed that users generally use our website for specific types of keywords which can divided into a couple of distinct classes, namely: politics/politicians, celebrities, products/brands, sports/sportsmen, media/movies/music. So, we can attempt to perform separate sentiment analysis on tweets that only belong to one of these classes (i.e., the training data would not be general but specific to one of these categories) and compare the results we get if we apply general sentiment analysis on it instead.

8. REFERENCES

[1] Albert Biffet and Eibe Frank. Sentiment Knowledge Discovery in Twitter Streaming Data. Discovery Science, Lecture Notes in Computer Science, 2010, Volume 6332/2010, 1-15, DOI: 10.1007/978-3-642-16184-1_1

[2] Alec Go, Richa Bhayani and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Project Technical Report, Stanford University, 2009.

[3] Alexander Pak and Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of international conference on Language Resources and Evaluation (LREC), 2010.

[4] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner and Isabell M. Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2010.

[5] S. Batra and D. Rao," Entity Based Sentiment Analysis on Twitter", Stanford University,2010.

[6] Saif M.Mohammad and Xiaodan zhu ,Sentiment Analysis on of social media texts:,2014.

[7] Ekaterina kochmar,University of Cambridge,at the Cambridge coding Academy DataScience.2016.

[8] Manju Venugopalan and Deepa Gupta, Exploring Sentiment Analysis on Twitter Data, IEEE2015