TRIBHUVAN UNIVERSITY

Institute of Science and Technology

National College Of Computer Studies

Project Report On

Twitter Sentiment Analysis

Under The Supervision of

Mr. Kendra Regmi

Submitted By

Pratik Dhakal

Intaj Tamang

Jan 7,2024

Acknowledgement

We would like to express our heartfelt gratitude and appreciation to all those who contributed to the successful completion of this project. Our sincere thanks go to **National College Of Computer Studies** for beautiful platform and encouragement. And we would like to express our sincere gratitude to **Mr. Kendra Ghimire** for valuable support for the completion of the project.

We want to thank everyone who helped make this project a success. Their support, hard work, and knowledge were crucial in facing problems and reaching our goals. We're really grateful for their help, advice, and dedication during this project.

This project would not have been possible without the collaborative efforts and hard work of each individual involved. We extend our gratitude to everyone who played a part, no matter how big or small, in making this project a reality.

Thank you all for your invaluable contributions.

Sincerely,

Pratik Dhakal

Intaj Tamang

Jan 3

Abstract

In the project "Analyzing Social Media Sentiment," the focus lies on extracting insights from social media content available through the Twitter API or Kaggle's Sentiment Analysis Dataset . The project utilizes Python programming language along with Kaggle for collecting Twitter data and TextBlob for sentiment analysis.

The primary objective involves collecting social media data from Twitter or Kaggle's dataset, encompassing a range of sentiments expressed by users. Data preprocessing techniques, including data cleansing, normalization, tokenization, and removal of irrelevant elements, are employed to refine and standardize the dataset. This refined dataset serves as the foundation for precise sentiment analysis using TextBlob and various Natural Language Processing (NLP) methods.

This project is essential because it helps us understand what people feel and think on social media. By using Python, and TextBlob, we can gather and analyze data more easily. This lets us see trends in people's emotions and opinions on social media, giving us a better understanding of what users think overall.

Key tools and resources for this project include Python programming language, Kaggle for collecting Twitter data, TextBlob for sentiment analysis, and the Sentiment Analysis Dataset available on Kaggle.

Completing this project shows how to analyze feelings effectively. It also helps us learn more about how people feel on social media. This information can be really useful for things like advertising, understanding what people think, and making better decisions based on that.

Table of contents

Chapter 1

Introduction

Twitter is a popular social networking website where members create and interact with messages known as “tweets”. This serves as a mean for individuals to express their thoughts or feelings about different subjects. Various different parties such as consumers and marketers have done sentiment analysis on such tweets to gather insights into products or to conduct market analysis. Furthermore, with the recent advancements in machine learning algorithms, we are able improve the accuracy of our sentiment analysis predictions.

In this report, we will attempt to conduct sentiment analysis on “tweets” using various different machine learning algorithms. We attempt to classify the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label.

We use the dataset from Twitter which was crawled and labelled positive/negative. The data provided comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. We also need to extract useful features from the text such unigrams and bigrams which is a form of representation of the “tweet”.

We use various machine learning algorithms to conduct sentiment analysis using the extracted features. However, just relying on individual models did not give a high accuracy so we pick the top few models to generate a model ensemble. Ensemble is a form of meta learning algorithm technique where we combine different classifiers in order to improve the prediction accuracy. Finally, we report our experimental results and findings at the end.

1.1 Objectives of the project

- Build a web application which is used to do the sentiment analysis of the tweets.
- Build a thorough ML model to recognize sentiments regarding various highlights:
    - Business Insights
    - Public Opinion
    - Crisis Management
    - Market research

1.2 Technology and framework used

- Streamlit

1.3 Software Required

- VS Code
- Jupyter Lab

1.4 System Requirements

- CPU: Intel Core i5 or equivalent
- RAM: 8 GB or higher
- Storage: 256 GB or higher (SSD recommended)
- GPU: NVIDIA GeForce GTX 1050 Ti or higher (for accelerated training of deep learning models)
- OS: Windows 10/11 or Linux
- Python version: 3.6 or higher
- Streamlit version: 0.84 or higher
- VS Code version: 1.55 or higher

1.5 Machine Learning

In a world where nearly, all manual tasks are being automated, the definition of "manual" is changing. We are living in an era of constant technological progress, and one such advancement is in the field of machine learning.

Machine learning is a data analytics technique that teaches computers to do what comes naturally to humans and animals: learn from experience. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases. Machine learning algorithms find natural patterns in data that generate insight and help you make better decisions and predictions.

Machine learning uses two types of techniques: Supervised learning, which trains a model on known input and output data so that it can predict future outputs, and Unsupervised learning, which finds hidden patterns or intrinsic structures in input data.

Supervised machine learning builds a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

Supervised learning uses classification and regression techniques to develop machine learning models. Classification techniques predict discrete responses. we use classification if our data can be tagged, categorized, or separated into specific groups or classes. Here in our problem, we have two discrete responses as "positive" or "negative". So, we used classification technique.

Regression techniques predict continuous responses. Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labelled responses.

1.5.1 Advantages of Machine Learning

- Machine learning algorithms are continuously improving in accuracy and efficiency, which makes better decisions.

- Machine learning is responsible for cutting the workload and time. By automating things, we let the algorithm do the hard work for us.

- Machine learning algorithms are good at handling data that is multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments

- Machine learning has a wide variety of applications. This means that we can apply Machine learning to any of the major fields. Machine learning has a role everywhere, from medical, business, and banking to science and tech.

1.6 Natural Language Processing (NLP)

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modelling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There's a good chance you've interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

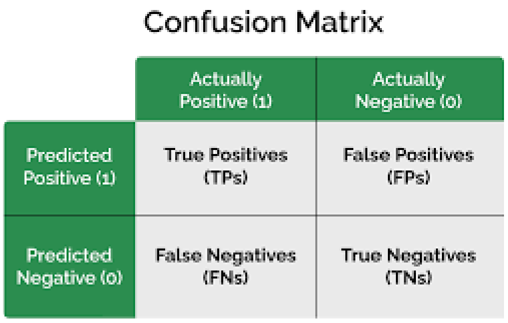
Chapter 2

Evaluation Metrics

2.1 Confusion Matrix

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 1 Confusion Matrix

Let’s decipher the matrix:

- The target variable has two values: Positive or Negative
- The columns represent the actual values of the target variable
- The rows represent the predicted values of the target variable

2.2 Accuracy

Accuracy is one metric for evaluating classification models. Informally, Accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\textit{Accuracy} = \textit{Number of correct predictions} / \textit{Number of Total Predictions} \tag{2.1}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{2.2}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

2.3 Precision and Recall

Precision is a good measure to determine, when the costs of False Positive is high. For instance, email spam detection. In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model. Medicare fraud detection model.

In the field of information retrieval precision is the fraction of retrieved documents that are relevant to the query:

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Recall calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

2.4 F1 Score

F1 is a function of Precision and Recall.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

Chapter 3

Data Pre-processing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

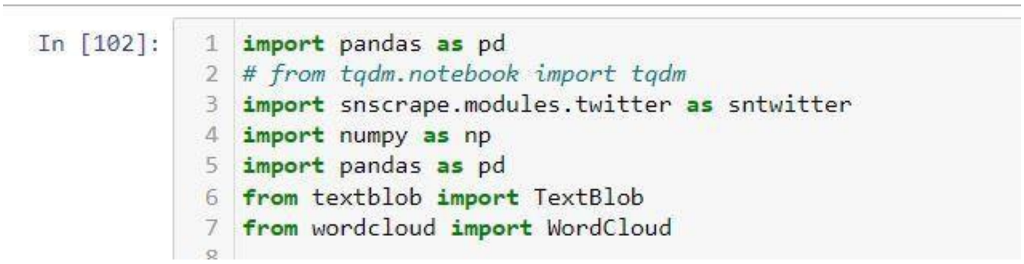
Before going into pre-processing and data exploration we will explain some of the concepts that allowed us to select our features.

3.1 Data Scrapping

We will be scrapping data directly from Twitter with help of ***snscrape***. *Snscrape* is a Python library that can be used to scrape tweets through Twitter's API without any restrictions or request limits.

Imports

Here we are importing the all of the basic libraries for scrapping and data pre-processing



```
In [102]:
import pandas as pd
# from tqdm.notebook import tqdm
import snscrape.modules.twitter as sntwitter
import numpy as np
import pandas as pd
from textblob import TextBlob
from wordcloud import WordCloud
```

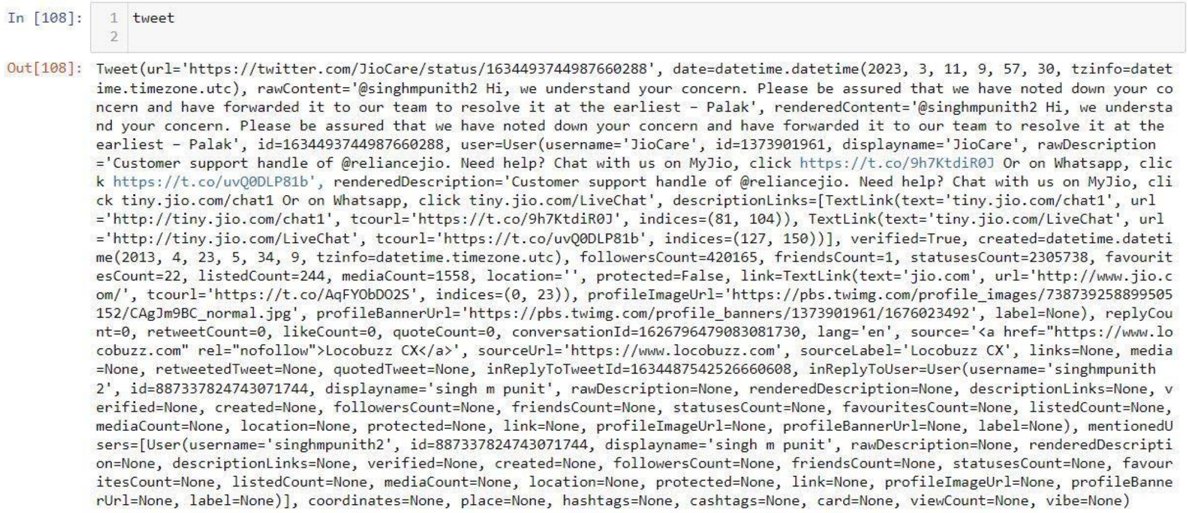
Figure 2 Scrapping and basic pre-processing Libraries

3.2 Data Description

The data given is in the form of a comma-separated tuple values with tweets and their corresponding sentiments. The training dataset is a tuple of type *<class 'snscrape.modules.twitter.Tweet'>* and consists tweet_id , sentiment, tweet where the tweet_id is a unique integer identifying the tweet, sentiment is either 0 (negative), 1 (neutral), 2 (positive) and tweet is the tweet enclosed in "". Similarly, the test dataset is a tuple file of type

tweet_id, tweet.

The dataset is a mixture of words, emoticons, symbols, URLs and references to people.



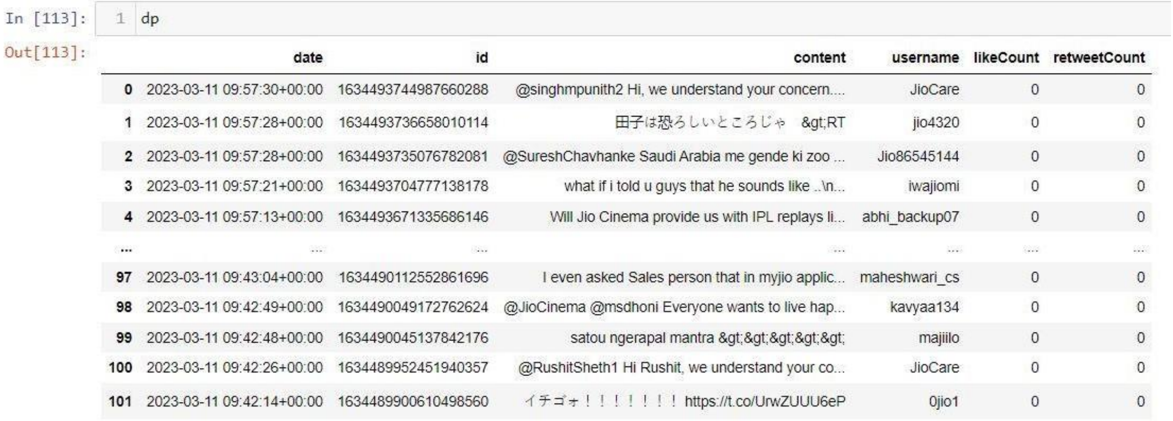
```
In [108]:  tweet

Out[108]: Tweet(url='https://twitter.com/JioCare/status/1634493744987660288', date=datetime.datetime(2023, 3, 11, 9, 57, 30, tzinfo=datet
          ime.timezone.utc), rawContent='@singhmpunith2 Hi, we understand your concern. Please be assured that we have noted down your co
          ncern and have forwarded it to our team to resolve it at the earliest - Palak', renderedContent='@singhmpunith2 Hi, we understa
          nd your concern. Please be assured that we have noted down your concern and have forwarded it to our team to resolve it at the
          earliest - Palak', id=1634493744987660288, user=User(username='JioCare', id=1373901961, displayname='JioCare', rawDescription
          ='Customer support handle of @reliancejio. Need help? Chat with us on MyJio, click https://t.co/9h7KtdiR0J Or on Whatsapp, clic
          k https://t.co/uvQ0DLP81b', renderedDescription='Customer support handle of @reliancejio. Need help? Chat with us on MyJio, cli
          ck tiny.jio.com/chat1 Or on Whatsapp, click tiny.jio.com/LiveChat', descriptionLinks=[TextLink(text='tiny.jio.com/chat1', url
          ='http://tiny.jio.com/chat1', tcourl='https://t.co/9h7KtdiR0J', indices=(81, 104)), TextLink(text='tiny.jio.com/LiveChat', url
          ='http://tiny.jio.com/LiveChat', tcourl='https://t.co/uvQ0DLP81b', indices=(127, 150))], verified=True, created=datetime.dateti
          me(2013, 4, 23, 5, 34, 9, tzinfo=datetime.timezone.utc), followersCount=420165, friendsCount=1, statusesCount=2305738, favourit
          esCount=22, listedCount=244, mediaCount=1558, location='', protected=False, link=TextLink(text='jio.com', url='http://www.jio.c
          om/', tcourl='https://t.co/AqFYObDO2S', indices=(0, 23)), profileImageUrl='https://pbs.twimg.com/profile_images/738739258899505
          152/CAgJm9BC_normal.jpg', profileBannerUrl='https://pbs.twimg.com/profile_banners/1373901961/1676023492', label=None), replyCou
          nt=0, retweetCount=0, likeCount=0, quoteCount=0, conversationId=1626796479083081730, lang='en', source='<a href="https://www.lo
          cobuzz.com" rel="nofollow">Locobuzz CX</a>', sourceUrl='https://www.locobuzz.com', sourceLabel='Locobuzz CX', links=None, media
          =None, retweetedTweet=None, quotedTweet=None, inReplyToTweetId=1634487542526660608, inReplyToUser=User(username='singhmpunith
          2', id=887337824743071744, displayname='singh m punit', rawDescription=None, renderedDescription=None, descriptionLinks=None, v
          erified=None, created=None, followersCount=None, friendsCount=None, statusesCount=None, favouritesCount=None, listedCount=None,
          mediaCount=None, location=None, protected=None, link=None, profileImageUrl=None, profileBannerUrl=None, label=None), mentionedU
          sers=[User(username='singhmpunith2', id=887337824743071744, displayname='singh m punit', rawDescription=None, renderedDescripti
          on=None, descriptionLinks=None, verified=None, created=None, followersCount=None, friendsCount=None, statusesCount=None, favour
          itesCount=None, listedCount=None, mediaCount=None, location=None, protected=None, link=None, profileImageUrl=None, profileBanne
          rUrl=None, label=None)], coordinates=None, place=None, hashtags=None, cashtags=None, card=None, viewCount=None, vibe=None)
```

Figure 3 Raw Data Format

3.3 Pre-processing

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as retweets, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We have applied an extensive number of pre-processing steps to standardize the dataset and reduce its size. We first do some general pre-processing on tweets which is as follows:

- Convert the tweets to data frame.
- Dropping unnecessary features.
- Removing URLs, user mentions, emoticon, hashtag, retweet



```
In [113]:  dp

Out[113]:
```

	date	id	content	username	likeCount	retweetCount
0	2023-03-11 09:57:30+00:00	1634493744987660288	@singhmpunith2 Hi, we understand your concern....	JioCare	0	0
1	2023-03-11 09:57:28+00:00	1634493736658010114	田子は恐ろしいところじゃ &gt;RT	jio4320	0	0
2	2023-03-11 09:57:28+00:00	1634493735076782081	@SureshChavhanke Saudi Arabia me gende ki zoo ...	Jio86545144	0	0
3	2023-03-11 09:57:21+00:00	1634493704777138178	what if i told u guys that he sounds like ..\n...	iwajiomi	0	0
4	2023-03-11 09:57:13+00:00	1634493671335686146	Will Jio Cinema provide us with IPL replays li...	abhi_backup07	0	0
...	...	...	...	...	...	...
97	2023-03-11 09:43:04+00:00	1634490112552861696	I even asked Sales person that in myjio applic...	maheshwari_cs	0	0
98	2023-03-11 09:42:49+00:00	1634490049172762624	@JioCinema @msdhoni Everyone wants to live hap...	kavyaa134	0	0
99	2023-03-11 09:42:48+00:00	1634490045137842176	satou ngerapal mantra &gt;&gt;&gt;&gt;&gt;	majiilo	0	0
100	2023-03-11 09:42:26+00:00	1634489952451940357	@RushitSheth1 Hi Rushit, we understand your co...	JioCare	0	0
101	2023-03-11 09:42:14+00:00	1634489900610498560	イチゴォ！！！！！！！ https://t.co/UrwZUUU6eP	0jio1	0	0

Figure 4 Raw Data after putting it into Data frame

We handle special twitter features as follows.

3.1.1 URL

Users often share hyperlinks to other webpages in their tweets. Any particular URL is not important for text classification as it would lead to very sparse features. Therefore, we replace all the URLs in tweets with the word URL. The regular expression used to match URLs is ((www\.[\S]+)|(https?://[\S]+)).

3.1.2 User Mention

Every twitter user has a handle associated with them. Users often mention other users in their tweets by @handle. We replace all user mentions with the word USER_MENTION. The regular expression used to match user mention is @[\S]+.

Emoticon(s) Type Regex Replacement
:), :), :-), (:, (:, (-:, :’) Smile (:\s?\)|:-\)|\(\s?:|\(-:|:\’\)) EMO_POS:D, : D, :-D, xD, x-D, XD, X-D Laugh (:\s?D|:-D|x-?D|X-?D) EMO_POS;-), ;), ;-D, ;D, (;, (-; Wink (:\s?\(|:-\(|\)\s?:|\)-:) EMO_POS<3, :* Love (<3|:*) EMO_POS:-(, : (, :(,):,)-: Sad (:\s?\(|:-\(|\)\s?:|\)-:) EMO_NEG
:,(, :’(, :"(Cry (:,\(|:\’\(|:"\() EMO_NEG

3.1.3 Emoticon

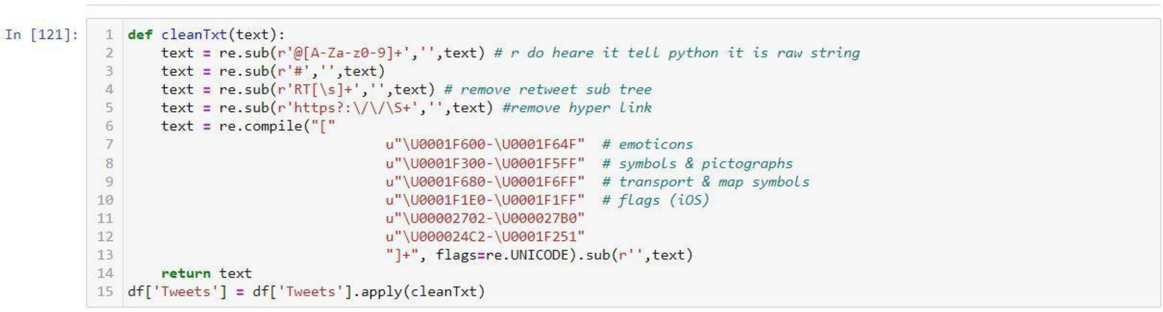
Users often use a number of different emoticons in their tweet to convey different emotions. It is impossible to exhaustively match all the different emoticons used on social media as the number is ever increasing. However, we match some common emoticons which are used very frequently. We replace the matched emoticons with either EMO_POS or EMO_NEG depending on whether it is conveying a positive or a negative emotion.

3.1.4 Hashtag

Hashtags are unspaced phrases prefixed by the hash symbol (#) which is frequently used by users to mention a trending topic on twitter. We replace all the hashtags with the words with the hash symbol. For example, #hello is replaced by hello. The regular expression used to match hashtags is #(\S+).

3.1.5 Retweet

Retweets are tweets which have already been sent by someone else and are shared by other users. Retweets begin with the letters RT. We remove RT from the tweets as it is not an important feature for text classification.



```
In [121]:
def cleanTxt(text):
    text = re.sub(r'@[A-Za-z0-9]+','',text) # r do heare it tell python it is raw string
    text = re.sub(r'#','',text)
    text = re.sub(r'RT[\s]+','',text) # remove retweet sub tree
    text = re.sub(r'https?:\/\/\S+','',text) #remove hyper link
    text = re.compile("["
                              u"\U0001F600-\U0001F64F"  # emoticons
                              u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                              u"\U0001F680-\U0001F6FF"  # transport & map symbols
                              u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                              u"\U00002702-\U000027B0"
                              u"\U000024C2-\U0001F251"
                              "]+", flags=re.UNICODE).sub(r'',text)
    return text
df['Tweets'] = df['Tweets'].apply(cleanTxt)
```

Figure 5 Data Cleaning

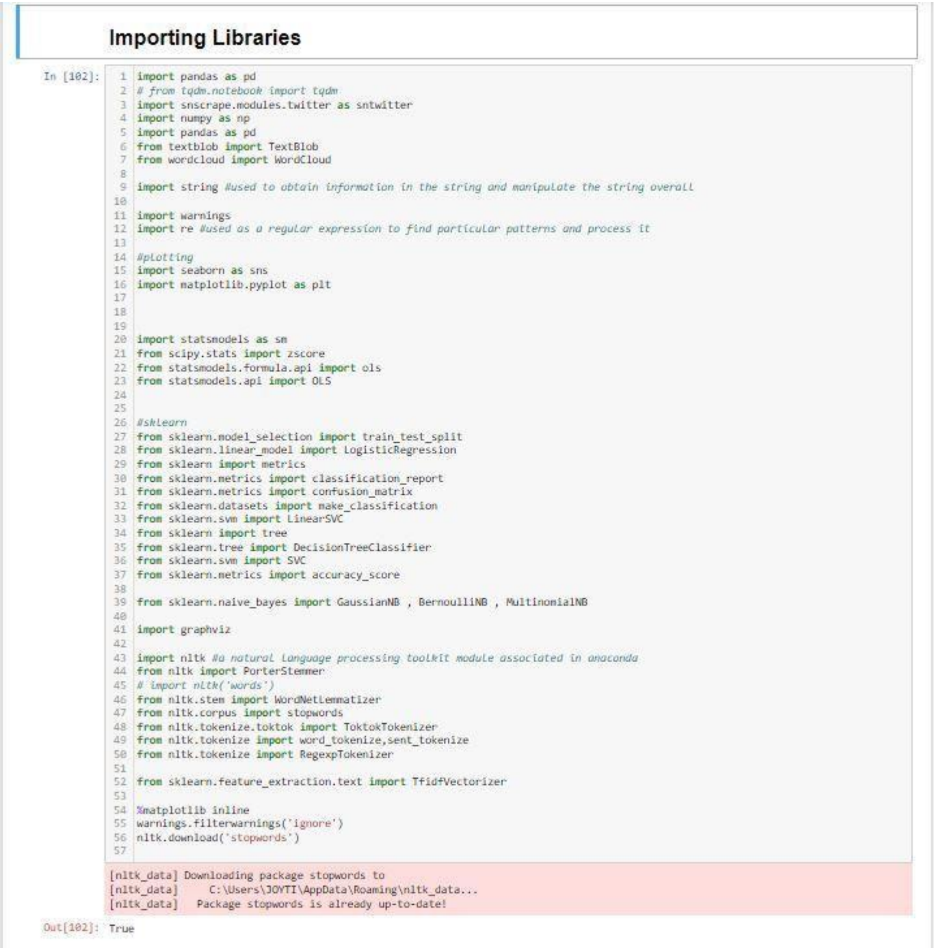
Chapter 4

Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

1. Understanding Data
2. Removing unnecessary features
3. Removing Stopwords and patterns
4. Tokenization
5. Applying Stemming and Lemmatization
6. Data Visualization

Here we are importing the all of the basic libraries for data scrapping, data pre-processing, Exploratory Data Analysis (EDA) and data Modelling.



Importing Libraries

In [102]:

```
import pandas as pd
# from tqdm.notebook import tqdm
import snscrape.modules.twitter as sntwitter
import numpy as np
import pandas as pd
from textblob import TextBlob
from wordcloud import WordCloud

import string #used to obtain information in the string and manipulate the string overall

import warnings
import re #used as a regular expression to find particular patterns and process it

#plotting
import seaborn as sns
import matplotlib.pyplot as plt



import statsmodels as sm
from scipy.stats import zscore
from statsmodels.formula.api import ols
from statsmodels.api import OLS


#sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.datasets import make_classification
from sklearn.svm import LinearSVC
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

from sklearn.naive_bayes import GaussianNB , BernoulliNB , MultinomialNB

import graphviz

import nltk #a natural language processing toolkit module associated in anaconda
from nltk import PorterStemmer
# import nltk('words')
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.tokenize import word_tokenize,sent_tokenize
from nltk.tokenize import RegexpTokenizer

from sklearn.feature_extraction.text import TfidfVectorizer

%matplotlib inline
warnings.filterwarnings('ignore')
nltk.download('stopwords')

```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\JOYTI\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[102]: True

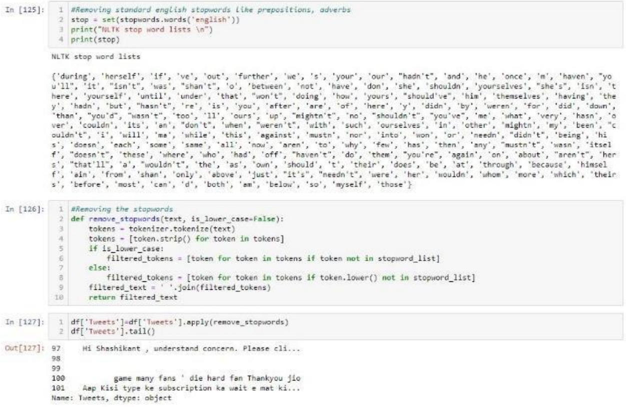
Figure 6 Libraries Imported

4.1 Stopwords:

The words which are generally filtered out before processing a natural language are called stop words. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and does not add much information to the text. Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”.

Stop words are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to the important information. In order words, we can say that the removal of such words does not show any negative consequences on the model we train for our task.

Removal of stop words definitely reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training.



```
In [125]:
#Removing standard english stopwords like prepositions, adverbs
stop = set(stopwords.words('english'))
print("NLTK stop word lists \n")
print(stop)
```

```
NLTK stop word lists

{'during', 'herself', 'if', 've', 'out', 'further', 'we', 's', 'your', 'our', "hadn't", 'and', 'he', 'once', 'm', 'haven', "yo
u'll", 'it', "isn't", 'was', "shan't", 'o', 'between', 'not', 'have', 'don', 'she', 'shouldn', 'yourselves', "she's", 'isn', 't
here', 'yourself', 'until', 'under', 'that', "won't", 'doing', 'how', 'yours', "should've", 'him', 'themselves', 'having', 'the
y', 'hadn', 'but', "hasn't", 're', 'is', 'you', 'after', 'are', 'of', 'here', 'y', 'didn', 'by', 'weren', 'for', 'did', 'down',
'than', "you'd", "wasn't", 'too', 'll', 'ours', 'up', "mightn't", 'no', "shouldn't", "you've", 'me', 'what', 'very', 'hasn', 'o
ver', 'couldn', 'its', 'an', "don't", 'when', "weren't", 'with', 'such', 'ourselves', 'in', 'other', 'mightn', 'my', 'been', "c
ouldn't", 'i', 'will', 'ma', 'while', 'this', 'against', 'mustn', 'nor', 'into', 'won', 'or', 'needn', "didn't", 'being', 'hi
s', 'doesn', 'each', 'some', 'same', 'all', 'now', 'aren', 'to', 'why', 'few', 'has', 'then', 'any', "mustn't", 'wasn', 'itsel
f', "doesn't", 'these', 'where', 'who', 'had', 'off', "haven't", 'do', 'them', "you're", 'again', 'on', 'about', "aren't", 'her
s', "that'll", 'a', "wouldn't", 'the', 'as', 'own', 'should', 't', 'their', 'does', 'be', 'at', 'through', 'because', 'himsel
f', 'ain', 'from', 'shan', 'only', 'above', 'just', "it's", "needn't", 'were', 'her', 'wouldn', 'whom', 'more', 'which', 'their
s', 'before', 'most', 'can', 'd', 'both', 'am', 'below', 'so', 'myself', 'those'}
```

```
In [126]:
#Removing the stopwords
def remove_stopwords(text, is_lower_case=False):
    tokens = tokenizer.tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stopword_list]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stopword_list]
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text
```

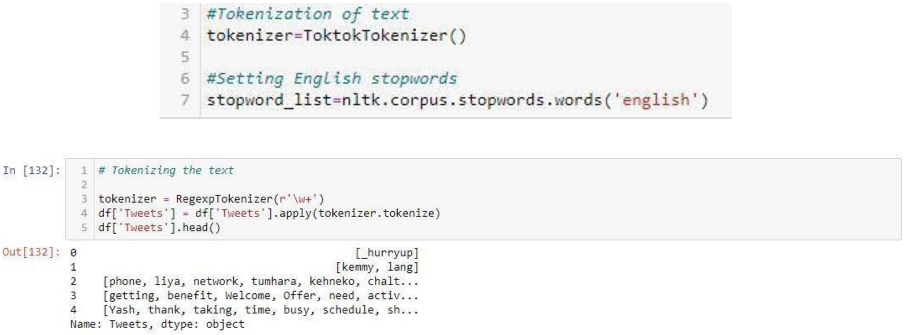
```
In [127]:
df['Tweets']=df['Tweets'].apply(remove_stopwords)
df['Tweets'].tail()
```

```
Out[127]:
97      Hi Shashikant , understand concern. Please cli...
98
99
100            game many fans ' die hard fan Thankyou jio
101     Aap Kisi type ke subscription ka wait e mat ki...
Name: Tweets, dtype: object
```

Figure 7 Removing Stopwords

4.2 Tokenization:

Tokenization is used in natural language processing to split paragraphs and sentences into smaller units that can be more easily assigned meaning.



```
#Tokenization of text
tokenizer=ToktokTokenizer()

#Setting English stopwords
stopword_list=nltk.corpus.stopwords.words('english')
```

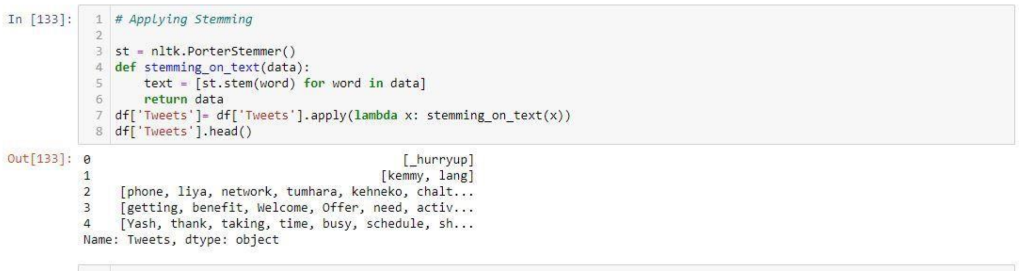
```
In [132]:
# Tokenizing the text

tokenizer = RegexpTokenizer(r'\w+')
df['Tweets'] = df['Tweets'].apply(tokenizer.tokenize)
df['Tweets'].head()
```

```
Out[132]:
0                                      [_hurryup]
1                                   [kemmy, lang]
2    [phone, liya, network, tumhara, kehneko, chalt...
3    [getting, benefit, Welcome, Offer, need, activ...
4    [Yash, thank, taking, time, busy, schedule, sh...
Name: Tweets, dtype: object
```

Figure 8 Tokenization

4.3 Stemming:

Stemming is a natural language processing technique that lowers inflection in words to their root forms, hence aiding in the pre-processing of text, words, and documents for text normalization.



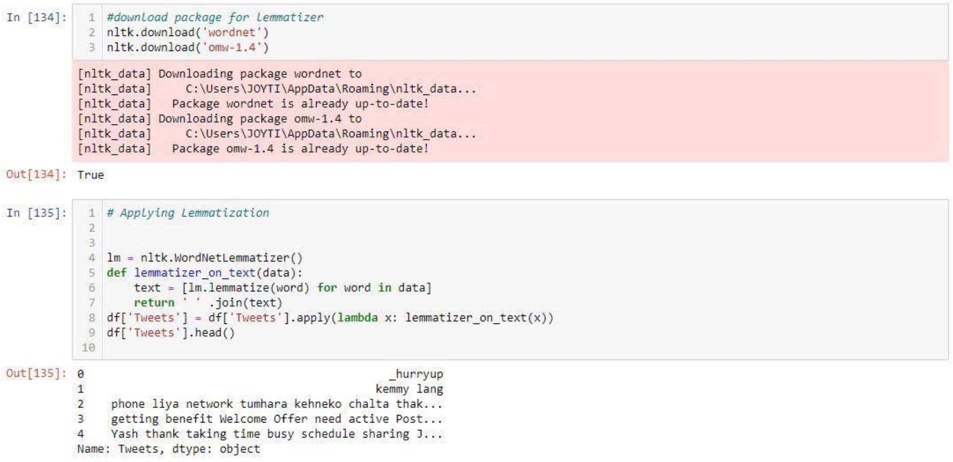
```
In [133]: # Applying Stemming

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

Out[133]: 0                                         [_hurryup]
          1                                      [kemmy, lang]
          2    [phone, liya, network, tumhara, kehneko, chalt...
          3    [getting, benefit, Welcome, Offer, need, activ...
          4    [Yash, thank, taking, time, busy, schedule, sh...
          Name: Tweets, dtype: object
```

Figure 9 Stemming

4.4 Lemmatization:

Lemmatization is a text normalization technique used in Natural Language Processing (NLP), that switches any kind of a word to its base root mode. Lemmatization is responsible for grouping different inflected forms of words into the root form, having the same meaning.

The purpose of lemmatization is same as that of stemming but overcomes the drawbacks of stemming. In stemming, for some words, it may not give may not give meaningful representation such as “Histori”. Here, lemmatization comes into picture as it gives meaningful word.



```
In [134]: #download package for lemmatizer
          nltk.download('wordnet')
          nltk.download('omw-1.4')

          [nltk_data] Downloading package wordnet to
          [nltk_data]     C:\Users\JOYTI\AppData\Roaming\nltk_data...
          [nltk_data]   Package wordnet is already up-to-date!
          [nltk_data] Downloading package omw-1.4 to
          [nltk_data]     C:\Users\JOYTI\AppData\Roaming\nltk_data...
          [nltk_data]   Package omw-1.4 is already up-to-date!

Out[134]: True

In [135]: # Applying Lemmatization


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


Out[135]: 0                                          _hurryup
          1                                        kemmy lang
          2    phone liya network tumhara kehneko chalta thak...
          3    getting benefit Welcome Offer need active Post...
          4    Yash thank taking time busy schedule sharing J...
          Name: Tweets, dtype: object
```

Figure 10 Lemmatization

4.5 Data Visualization:

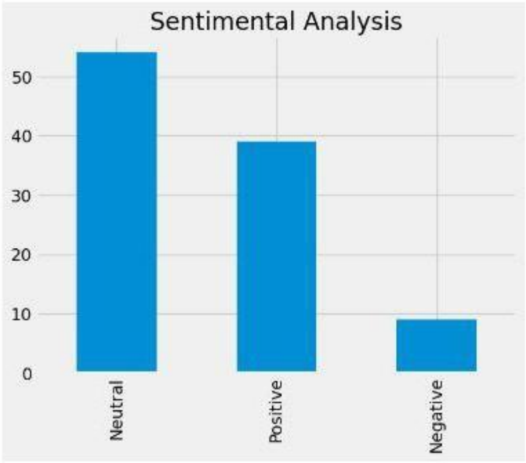


Figure 11 Bar Graph

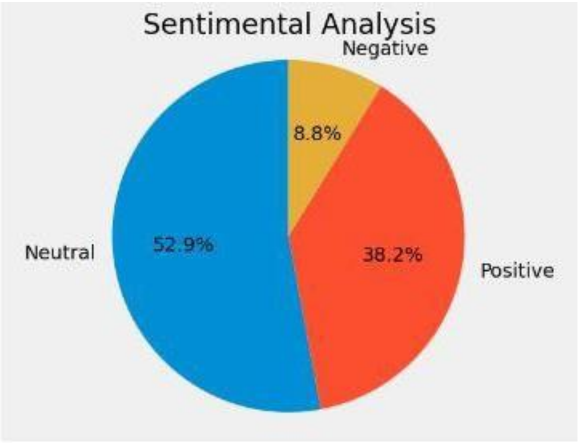


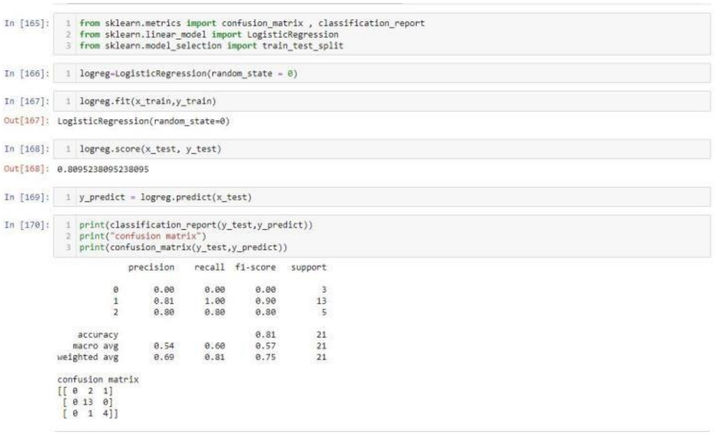
Figure 12 Pie Chart

Chapter 5

Model Building

5.1 Logistic Regression

Despite its name, it is implemented as a linear model for classification rather than regression in terms of the scikit-learn/ML nomenclature. We obtained accuracy score of 80.95%.



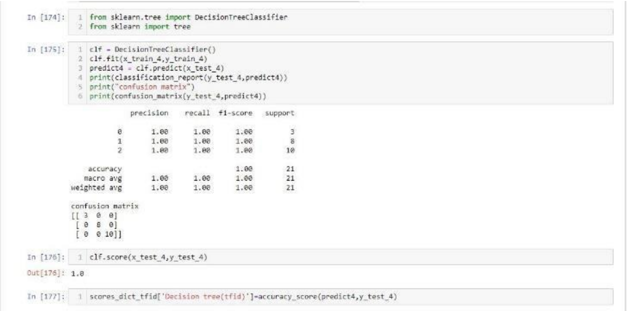
```
In [165]:  from sklearn.metrics import confusion_matrix , classification_report
           from sklearn.linear_model import LogisticRegression
           from sklearn.model_selection import train_test_split

In [166]:  logreg=LogisticRegression(random_state = 0)

In [167]:  logreg.fit(x_train,y_train)
Out[167]: LogisticRegression(random_state=0)

In [168]:  logreg.score(x_test, y_test)
Out[168]: 0.8095238095238095

In [169]:  y_predict = logreg.predict(x_test)

In [170]:  print(classification_report(y_test,y_predict))
           print("confusion matrix")
           print(confusion_matrix(y_test,y_predict))
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.81      1.00      0.90        13
           2       0.80      0.80      0.80         5

    accuracy                           0.81        21
   macro avg       0.54      0.60      0.57        21
weighted avg       0.69      0.81      0.75        21

confusion matrix
[[ 0  2  1]
 [ 0 13  0]
 [ 0  1  4]]
```

Figure 13 Logistic Regression

5.2 Decision Tree

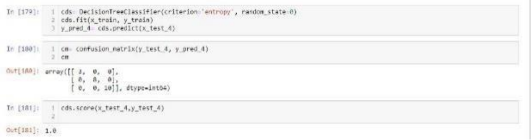
Decision trees are a classifier model in which each node of the tree represents a test on the attribute of the data set, and its children represent the outcomes. The leaf nodes represent the final classes of the data points. It is a supervised classifier model which uses data with known labels to form the decision tree and then the model is applied on the test data. For each node in the tree the best test condition or decision has to be taken. We use the GINI factor to decide the best split. We choose a split that minimizes the GINI factor.

We implemented DecisionTreeClassifier from from sklearn.tree using 'Entropy' criterion and random_state=0. We obtained accuracy score of 1.0.



```
In [174]:  from sklearn.tree import DecisionTreeClassifier
           from sklearn import tree

In [175]:  clf = DecisionTreeClassifier()
           clf.fit(x_train_4,y_train_4)
           predict4 = clf.predict(x_test_4)
           print(classification_report(y_test_4,predict4))
           print("confusion matrix")
           print(confusion_matrix(y_test_4,predict4))
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         8
           2       1.00      1.00      1.00        10

    accuracy                           1.00        21
   macro avg       1.00      1.00      1.00        21
weighted avg       1.00      1.00      1.00        21

confusion matrix
[[ 3  0  0]
 [ 0  8  0]
 [ 0  0 10]]

In [176]:  clf.score(x_test_4,y_test_4)
Out[176]: 1.0

In [177]:  scores_dict_tfid['Decision tree(tfid)']=accuracy_score(predict4,y_test_4)
```

Figure 14 Decision Tree (GINI)



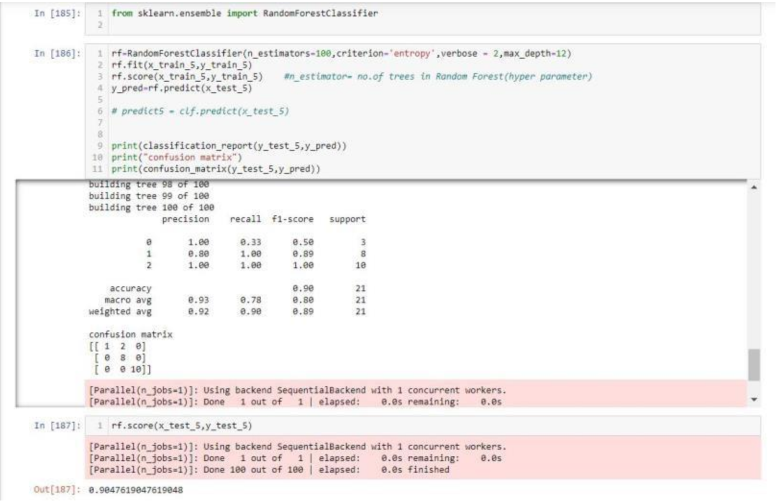
```
In [179]:  cds= DecisionTreeClassifier(criterion='entropy', random_state=0)
           cds.fit(x_train, y_train)
           y_pred_4= cds.predict(x_test_4)

In [180]:  cm= confusion_matrix(y_test_4, y_pred_4)
           cm

Out[180]: array([[ 3,  0,  0],
                 [ 0,  8,  0],
                 [ 0,  0, 10]], dtype=int64)

In [181]:  cds.score(x_test_4,y_test_4)

Out[181]: 1.0
```

Figure 15 Decision Tree (Entropy)

5.3 Random Forrest

In random forests (see RandomForestClassifier and RandomForestRegressor classes), each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max_features. (See the parameter tuning guidelines for more details). The purpose of these two sources of randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.

We implemented random forest algorithm by using RandomForestClassifier from sklearn.ensemble provided by scikit-learn. We experimented using 100 estimators (trees) using entropy criterion, verbose =2 and max_depth of tree = 12. We obtained accuracy score of 90.47%.



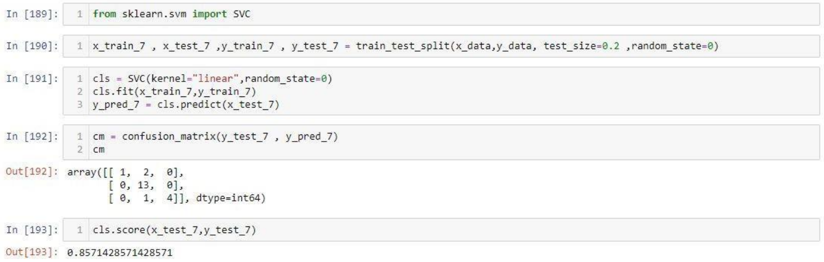
```
In [185]:  from sklearn.ensemble import RandomForestClassifier

In [186]:  rf=RandomForestClassifier(n_estimators=100,criterion='entropy',verbose = 2,max_depth=12)
           rf.fit(x_train_5,y_train_5)
           rf.score(x_train_5,y_train_5)    #n_estimator= no.of trees in Random Forest(hyper parameter)
           y_pred=rf.predict(x_test_5)

           # predict5 = clf.predict(x_test_5)


           print(classification_report(y_test_5,y_pred))
           print("confusion matrix")
           print(confusion_matrix(y_test_5,y_pred))

           building tree 98 of 100
           building tree 99 of 100
           building tree 100 of 100
                         precision    recall  f1-score   support

                      0       1.00      0.33      0.50         3
                      1       0.80      1.00      0.89         8
                      2       1.00      1.00      1.00        10

               accuracy                           0.90        21
              macro avg       0.93      0.78      0.80        21
           weighted avg       0.92      0.90      0.89        21

           confusion matrix
           [[ 1  2  0]
            [ 0  8  0]
            [ 0  0 10]]
           [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
           [Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s

In [187]:  rf.score(x_test_5,y_test_5)

           [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
           [Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
           [Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.0s finished

Out[187]: 0.9047619047619048
```

Figure 16 Random Forrest

5.4 SVM

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

We utilise the SVM classifier available in *sklearn*. We set the C term to be 0.1. C term is the penalty parameter of the error term. In other words, this influences the misclassification on the objective function. We obtain accuracy score of 85.71%.

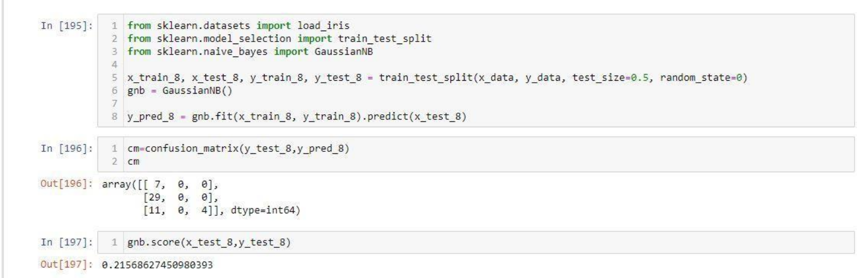


```
In [189]:  from sklearn.svm import SVC

In [190]:  x_train_7 , x_test_7 ,y_train_7 , y_test_7 = train_test_split(x_data,y_data, test_size=0.2 ,random_state=0)

In [191]:  cls = SVC(kernel="linear",random_state=0)
           cls.fit(x_train_7,y_train_7)
           y_pred_7 = cls.predict(x_test_7)

In [192]:  cm = confusion_matrix(y_test_7 , y_pred_7)
           cm

Out[192]: array([[ 1,  2,  0],
                 [ 0, 13,  0],
                 [ 0,  1,  4]], dtype=int64)

In [193]:  cls.score(x_test_7,y_test_7)

Out[193]: 0.8571428571428571
```

Figure 17 SVM

5.5 Naïve Bayes

We used GausianNB from sklearn.naive_bayes package of scikit-learn for Naive Bayes classification. We used Laplace smoothed version of Naive Bayes with the smoothing parameter _ set
to its default value of 1. We used sparse vector representation for classification and ran experiments using both presence and frequency feature types. We found that presence features outperform frequency
features because Naive Bayes is essentially built to work better on integer features rather
8
than floats. We obtain a best validation accuracy of 21.5% using Naive Bayes.



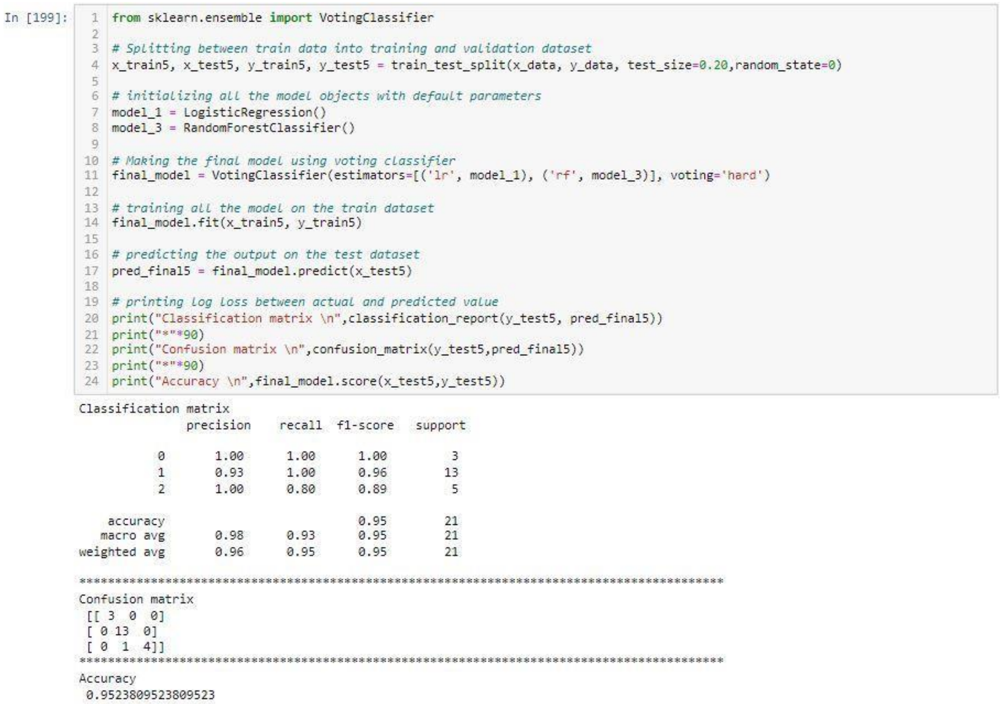
```
In [195]:  from sklearn.datasets import load_iris
           from sklearn.model_selection import train_test_split
           from sklearn.naive_bayes import GaussianNB

           x_train_8, x_test_8, y_train_8, y_test_8 = train_test_split(x_data, y_data, test_size=0.5, random_state=0)
           gnb = GaussianNB()

           y_pred_8 = gnb.fit(x_train_8, y_train_8).predict(x_test_8)

In [196]:  cm=confusion_matrix(y_test_8,y_pred_8)
           cm

Out[196]: array([[ 7,  0,  0],
                 [29,  0,  0],
                 [11,  0,  4]], dtype=int64)

In [197]:  gnb.score(x_test_8,y_test_8)

Out[197]: 0.21568627450980393
```

Figure 18 Naive Bayes

5.6 Ensemble

The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions.

In learning models, noise, variance, and bias are the major sources of error. The ensemble methods in machine learning help minimize these error-causing factors, thereby ensuring the accuracy and stability of machine learning (ML) algorithms. We obtain a best validation accuracy of 95.23% using Ensemble algorithm.

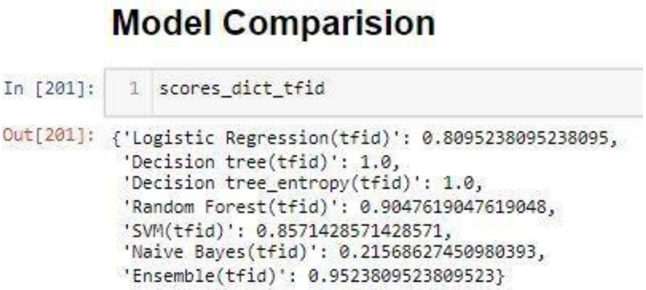


```
from sklearn.ensemble import VotingClassifier

# Splitting between train data into training and validation dataset
x_train5, x_test5, y_train5, y_test5 = train_test_split(x_data, y_data, test_size=0.20,random_state=0)

# initializing all the model objects with default parameters
model_1 = LogisticRegression()
model_3 = RandomForestClassifier()

# Making the final model using voting classifier
final_model = VotingClassifier(estimators=[('lr', model_1), ('rf', model_3)], voting='hard')

# training all the model on the train dataset
final_model.fit(x_train5, y_train5)

# predicting the output on the test dataset
pred_final5 = final_model.predict(x_test5)

# printing log loss between actual and predicted value
print("Classification matrix \n",classification_report(y_test5, pred_final5))
print("*"*90)
print("Confusion matrix \n",confusion_matrix(y_test5,pred_final5))
print("*"*90)
print("Accuracy \n",final_model.score(x_test5,y_test5))
```

```
Classification matrix 
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.93      1.00      0.96        13
           2       1.00      0.80      0.89         5

    accuracy                           0.95        21
   macro avg       0.98      0.93      0.95        21
weighted avg       0.96      0.95      0.95        21

******************************************************************************************
Confusion matrix 
 [[ 3  0  0]
 [ 0 13  0]
 [ 0  1  4]]
******************************************************************************************
Accuracy 
 0.9523809523809523
```

Figure 19 Ensemble Model

5.7 Model Comparison:



Model Comparision

```
scores_dict_tfid
```

```
{'Logistic Regression(tfid)': 0.8095238095238095,
 'Decision tree(tfid)': 1.0,
 'Decision tree_entropy(tfid)': 1.0,
 'Random Forest(tfid)': 0.9047619047619048,
 'SVM(tfid)': 0.8571428571428571,
 'Naive Bayes(tfid)': 0.21568627450980393,
 'Ensemble(tfid)': 0.9523809523809523}
```

Figure 20 Model Score Comparison

Chapter 6

Deployment

6.1 Streamlit

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app.

For deployment we used VSCode and following are the libraries we used:

- import snscrape.modules.twitter as sntwitter
- from textblob import TextBlob
- from wordcloud import WordCloud
- import pandas as pd
- import streamlit as st
- import re
- import matplotlib.pyplot as plt
- import plotly.express as px
- import plotly.graph_objects as go
- from PIL import Image
- import seaborn as sns
- import langid
- from langdetect import detect
- import nltk
- from nltk.sentiment.vader import SentimentIntensityAnalyzer
- import datetime

6.2 Steps to run streamlit:

Step1: Creating environment

- pip install pipenv
- pipenv install streamlit pandas
- pip install -r requrements.txt
- pipenv shell

Step 2: Import all libraries mention in requriment.txt file

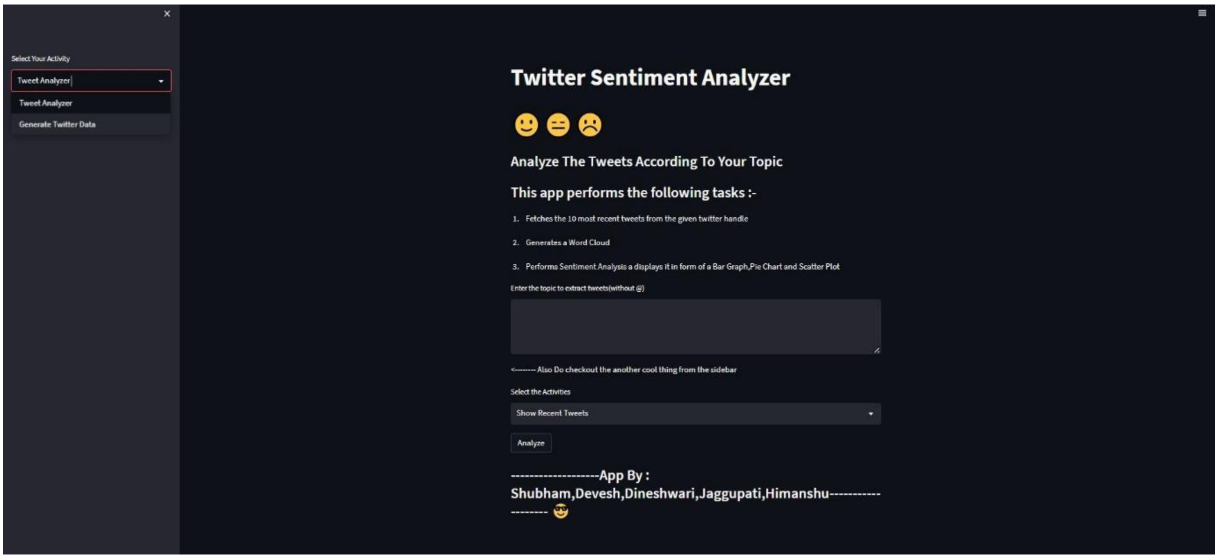
Step 3: Save streamlit code as app.py file.

Step 4: To run streamlit simply on terminal change directory to file location then type command streamlit run app.py and press enter OR in terminal on existing location enter streamlit run full_path_of_file_location/app.py and press enter. It will generate an URL and move to browser. In Browser we see localhost_no. and app UI interface.

Here are some steps you can try to resolve if any issue facing:

The error message suggests that the locking process has failed due to a resolution failure. This could be caused by a mismatch in sub-dependencies, which can often occur when different packages require different versions of the same package.

- Check your pipenv version and make sure it's up to date with the latest version.
- Run the command pipenv lock --pre to install any pre-release dependencies that may be causing the issue.
- Run the command pipenv clean to remove any cached dependencies.
- Run the command pipenv install --skip-lock to bypass the locking mechanism and install the packages without locking them.
- Run the command pipenv graph to inspect the dependencies and their versions to help identify any mismatches.
- If none of the above steps work, try creating a new virtual environment and installing the packages again. This can often solve issues caused by conflicting dependencies in the current environment.
- If there is a *scrapperexception* error, try updating snscrape(if snscrape is not upto date then uninstall and install updated version of snscrape And run streamlit App again following above step).



Twitter Sentiment Analyzer

🙂 😑 ☹️

Analyze The Tweets According To Your Topic

This app performs the following tasks :-

1. Fetches the 10 most recent tweets from the given twitter handle
2. Generates a Word Cloud
3. Performs Sentiment Analysis a displays it in form of a Bar Graph,Pie Chart and Scatter Plot

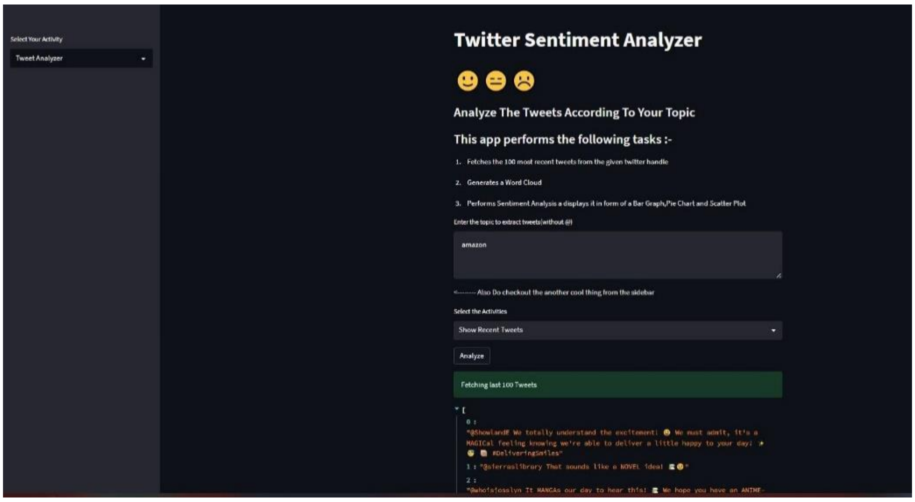
Enter the topic to extract tweets(without @)

<------- Also Do checkout the another cool thing from the sidebar

Select the Activities: Show Recent Tweets

-------------------App By : Shubham,Devesh,Dineshwari,Jaggupati,Himanshu------------------ 😎

Figure 21 Homepage



Twitter Sentiment Analyzer

🙂 😑 ☹️

Analyze The Tweets According To Your Topic

This app performs the following tasks :-

1. Fetches the 100 most recent tweets from the given twitter handle
2. Generates a Word Cloud
3. Performs Sentiment Analysis a displays it in form of a Bar Graph,Pie Chart and Scatter Plot

Enter the topic to extract tweets(without @): amazon

<------- Also Do checkout the another cool thing from the sidebar

Select the Activities: Show Recent Tweets

Fetching last 100 Tweets

```
[
  0 :
  "@ShowlandE We totally understand the excitement! 😃 We must admit, it's a
  MAGICal feeling knowing we're able to deliver a little happy to your day! ✨
  😊 📦 #DeliveringSmiles"
  1 : "@sierraslibrary That sounds like a NOVEL idea! 📚😊"
  2 :
  "@whoisjosslyn It MANGAs our day to hear this! 📚 We hope you have an ANIME-
```

Figure 22 Recent Tweets

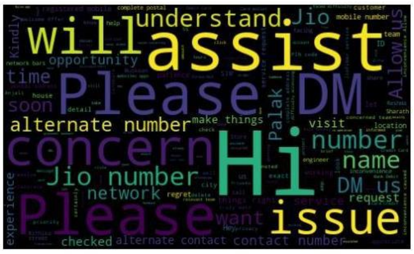


Figure 23 Word Cloud

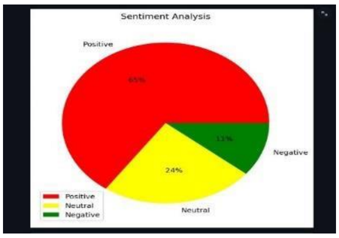


Figure 25 Pie Chart

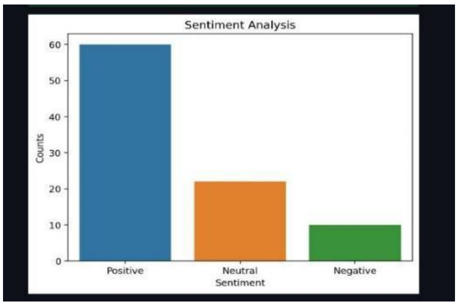


Figure 24 Bar Graph

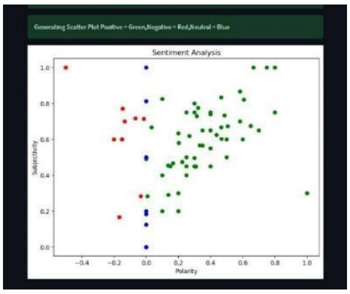
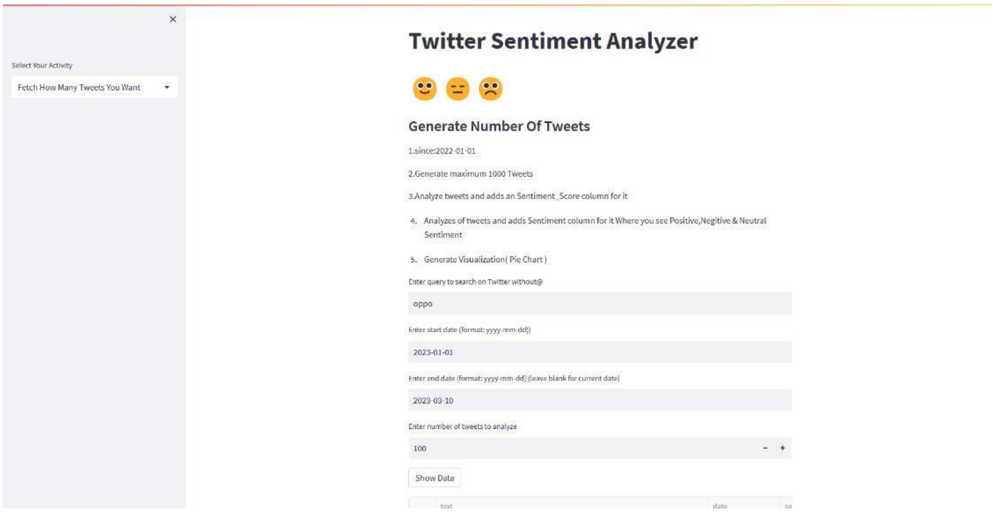


Figure 26 Scatter Plot



Select Your Activity

Fetch How Many Tweets You Want

Twitter Sentiment Analyzer

Generate Number Of Tweets

1.since:2022-01-01

2.Generate maximum 1000 Tweets

3.Analyze tweets and adds an Sentiment_Score column for it

4. Analyzes of tweets and adds Sentiment column for it Where you see Positive,Negitive & Neutral Sentiment

5. Generate Visualization(Pie Chart)

Enter query to search on Twitter without@

oppo

Enter start date (format: yyyy-mm-dd))

2023-01-01

Enter end date (format: yyyy-mm-dd) (leave blank for current date)

2023-03-10

Enter number of tweets to analyze

100

Figure 27 Fetch Number of Tweets

Chapter 7

Conclusion

Twitter sentiment analysis is necessary because it provides valuable insights that can help businesses and organizations make informed decisions, improve their products and services, and stay connected with their customers.

With the help of our web application "Twitter Sentiment Analyzer" businesses, organizations and an individual can perform sentiment analysis about any topic and gather required information.

Our web application provides following features:

i. Get recent tweets of a topic or user
ii. Generate Word Cloud
iii. Sentiment Visualization
iv. Fetch how many tweets user wants according to date.

7.1 Scope of Improvement:

1. All three options can be merged into one option where we can see all the tweets according to timestamp, also we need to provide a calendar option instead of manually entering the dates.
2. Implementation of live sentiment analysis about a topic to get the trend.
3. Scatter plot being plotted can be plotted on the basis of whole tweet rather than a single word. Also selecting a datapoint can show the respective tweet.

References

1. Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1-2), 1-135.
2. A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(12), 2009.
3. Kiritchenko, S., & Mohammad, S. M. (2016). Sentiment analysis of short informal texts. Journal of Artificial Intelligence Research, 47, 1-49.
4. Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1), 1-167.
5. Pak, A., & Paroubek, P. (2010, May). Twitter as a corpus for sentiment analysis and opinion mining. In LREC (pp. 1320-1326).
6. Zhang, Y., & Wallace, B. C. (2017). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.038201)Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1-2), 1-135.