A PROJECT REPORT ON

TWITTER SENTIMENT OPINION MINING

Submitted By

Bazgha Razi
(Reg No: 191380214)

Under the supervision
Of

Mr. Kapil Singh

Partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

DELHI GLOBAL INSTITUTE OF TECHNOLOGY

Bahadurgarh, Jhajjar, Haryana 124201

Affiliated to

Maharshi Dayanand University

CERTIFICATE

This is to certify that the project "**Twitter Sentiment Opinion Mining**" submitted by "**Bazgha Razi, 191380214** ", in partial fulfillment of the requirement for the degree of Bachelor of Technology in Computer Science and Engineering under the Faculty of **"Delhi Global Institute of Technology** " during the academic session May 2023 , is a Bonafide record of word carried out under my guidance and supervision .

Signature
Delhi Global Institute of Technology

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project coordinator Mr. Kapil Singh Department of Computer Science and Engineering , **“Delhi Global Institute of Technology”** , for giving me the opportunity to work on this topic . It would never have been possible for me to take this project work to this level without his innovation idea and their relentless support and encouragement .

Bazgha Razi

Reg No: 191380214

B.Tech(CSE)

DECLARATION

I here by declare that the project report entitled **"Twitter Sentiment Opinion Mining"** submitted by me "**Bazgha Razi**" fulfillment the requirements for this project work under the guidance of professors.

I also declare that, the report is only prepared for my academic requirement not for any other purpose.

Bazgha Razi

Reg No: 191380214
B.Tech(CSE)

CONTENTS

INTRODUCTION

Now a days twitter, facebook, whatsapp are getting so much attention from people and also they are getting very much popular among people. Sentiment opinion mining provides many opportunities to develop a new application in the industrial field, sentiment analysis has big effect, like government organization and big companies, their desire is to know about what people think about their product, their market value.[1] The aim of sentiment opinion mining is to find out the mood, behavior and opinion of person from texts. for the sentiment analysis purpose, social networking used the various sentiment analysis techniques to take the public data. Sentiment opinion mining widely used in various domain such as finance, economics, defense, politics. The data available on the social networking sites can be unstructured and structured. Almost 80% data on the internet is unstructured. Sentiment opinion mining techniques are used to find out the people opinion on social media.[3] Twitter is also a huge platform in that different idea, thought, opinion are presented and exchanged. It does not matter where people came from, what religious opinions they hold, rich or poor, educated or uneducated, they comment, compliment, discuss, argue, insist.[2]

Data analysis is the process of applying organized and systematic statistical techniques to describe, recap, check and condense data. It is a multistep process that involves collecting, cleaning, organizing and analysing. Data mining is like applying techniques to mold data to suit our requirement. Data mining is needed because different sources like social media, transactions, public data, enterprises data etc. generates data of increasing volume, and it is important to handle and analyze such a big data. It won’t be wrong to say that social media is something we live by. In the 21st century social media has been the game changer, be it advertising, politics or globalization, it has been

estimated that data is increasing faster than before and by the year 2020; about 1.7 megabytes of additional data will be generated each instant for each person on the earth. More data has been generated in the past two years than ever before in the history of the mankind. It is clear from the fact that the number of internet users are now grown from millions to billions. Database which is opted for the proposed study is from Twitter. It is now day's very popular service which provides facility of micro blogging. In this people write short messages generally less than 140 characters, about 11 words on average. It is appropriate for analysis as the number of messages is large. It is much easier task as compared to searching blogs from the net. The objective of the proposed analysis, 'Sentiment Opinion Mining', is the analysis of the enormous amount of data easily available from social media. [4]

Algorithm generates an overall sentiment score from the inputted topic in terms of positive, negative or neutral, further it also works on finding the frequency of the words being used. Word cloud that is a pictorial representation of words based on frequency occurrence of words in the text is also generated.[3]

METHODOLOGY

Based on the different perspective. Sentiment opinion mining has different variety of class. In which only one is used in sentiment classification techniques. This is classified into two other approaches i.e. machine learning approach and lexicon based approach. We can add one more techniques i.e. hybrid approach. There are three main classification level i.e. sentence level, document level, and last one is aspect level. Based on sentiment analysis, polarities can be classified into three classes such as positive neutral or negative.

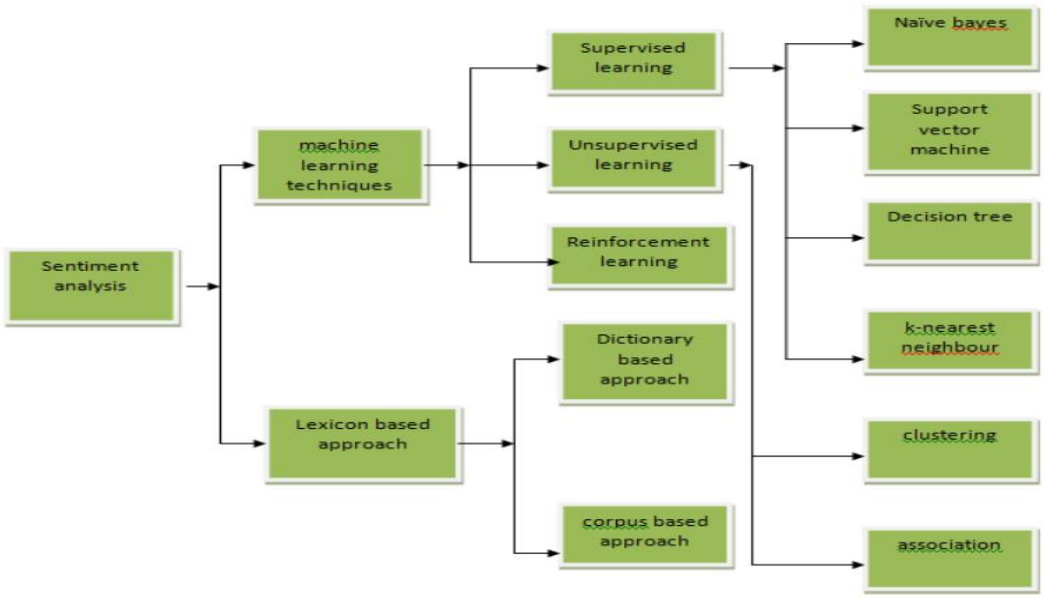


Figure: Sentiment Classification Techniques [5]

Sentiment opinion mining is area where we can classify the various techniques. It is most popular area of research. It is notably classified into two types such as machine learning based approach and lexicon-based approach. Lexicon based approach basically focused on negative and positive term and it is further classified into two types i.e. dictionary based and corpus based. Moreover, machine learning approach is focused on two techniques namely supervised and unsupervised approach.

PROJECT DETAILS

Name of the Project

Twitter Sentiment Opinion Mining

Sentiment opinion mining can be defined as a process that automates mining of attitudes, opinions, views and emotions from text, speech, tweets and database sources through Natural Language Processing (NLP).

Sentiment opinion mining involves classifying opinions in text into categories like "positive" or "negative" or "neutral". It's also referred as subjectivity analysis, opinion mining, and appraisal extraction.

The words opinion, sentiment, view and belief are used interchangeably but there are differences between them.

- **Opinion**: A conclusion open to dispute (because different experts have different opinions)
- **View**: subjective opinion
- **Belief**: deliberate acceptance and intellectual assent
- **Sentiment**: opinion representing one's feelings

Sentiment opinion mining is a term that include many tasks such as sentiment extraction, sentiment classification, subjectivity classification, summarization of opinions or opinion spam detection, among others.

It aims to analyze people's sentiments, attitudes, opinions emotions, etc. towards elements such as, products, individuals, topics, organizations, and services.

Hardware Used

Monitor: 1024 X 720 display

RAM: 8GB or more

Speed: 2.7GHZ and more

Cache: 512KB

Processor: Intel i5 or more

Software Used

Windows (7,10,11)

VS Code (Python IDE)

Libraries and Modules Used

Python

matplotlib

numpy

pandas

nltk

scikit-learn

Version Control/ IDE

Github

VS Code

Twitter Sentiment Opinion Mining

Twitter sentiment opinion mining analyses 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.[6]

Problem Statement

In this project, we try to implement an NLP **Twitter sentiment opinion mining model** that helps to overcome the challenges of sentiment classification of tweets. We will be classifying the tweets into positive or negative sentiments. The necessary details regarding the dataset involving the Twitter sentiment analysis project are:

The dataset provided is the **Sentiment140 Dataset** which consists of **1,600,000 tweets** that have been extracted using the Twitter API.[7] The various columns present in this Twitter data are:

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

Why Twitter Sentiment Opinion Mining?

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.

Application Area of Twitter Sentiment Opinion Mining

Applications that use Reviews from Websites:

Today Internet has a large collection of reviews and feedbacks on almost everything. This includes product reviews, feedbacks on political issues, comments about services, etc. Thus, there is a need for a sentiment analysis system that can extract sentiments about a particular product or services. It will help us to automate in provision of feedback or rating for the given product, item, etc. This would serve the needs of both the users and the vendors.

Applications in Business Intelligence:

It has been observed that people nowadays tend to look upon reviews of products which are available online before they buy them. And for many businesses, the online opinion decides the success or failure of their product. Thus, Sentiment Analysis plays an important role in businesses. Businesses also wish to extract sentiment from the online reviews in order to improve their products and in turn their reputation and help in customer satisfaction.

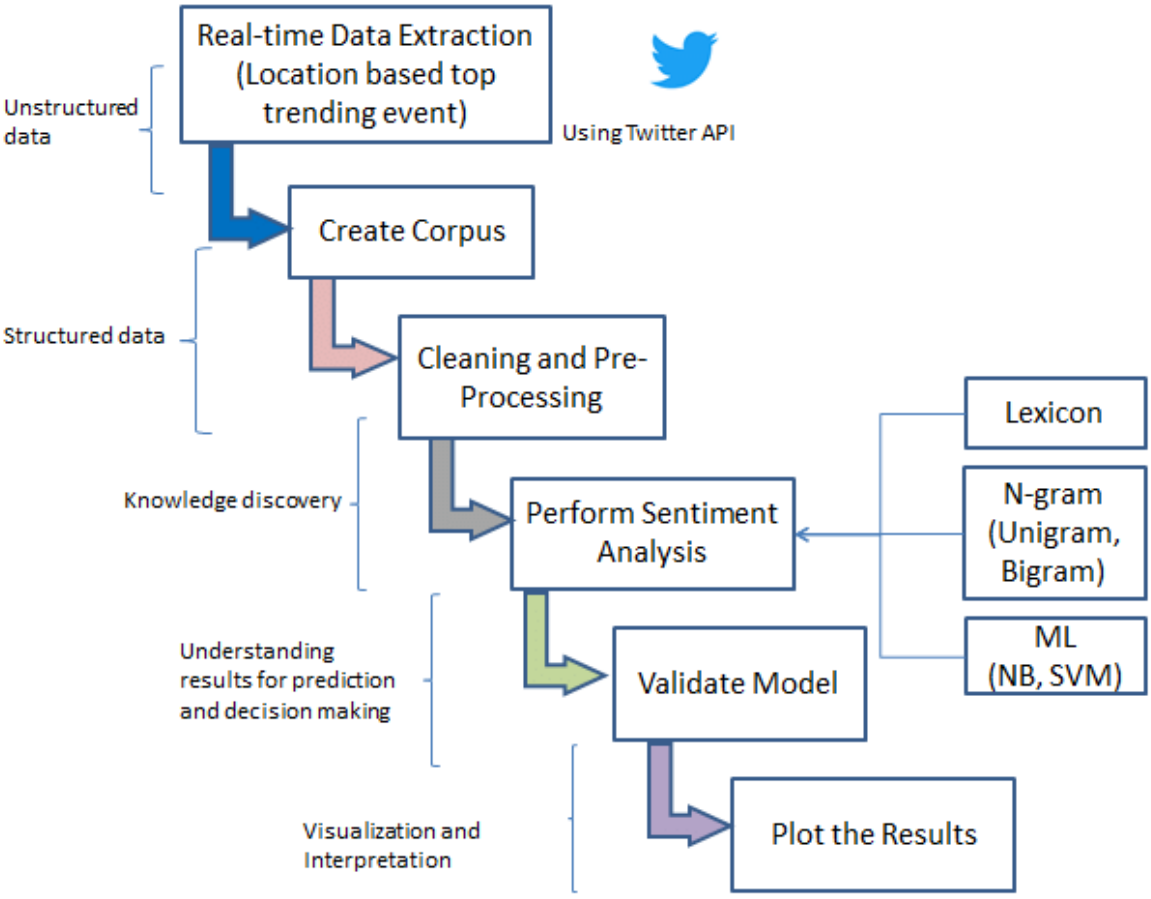
Applications in Smart Homes:

Smart homes are supposed to be the technology of the future. In future entire homes would be networked and people would be able to control any part of the home using a tablet device.

Recently there has been lot of research going on Internet of Things (IoT). Sentiment Analysis would also find its way in loT. Like for example, based on the current sentiment or emotion of the user, the home could alter its ambiance to create a soothing and peaceful environment.[9]

Sentiment Analysis can also be used in trend prediction. By tracking public views, important data regarding sales trends and customer satisfaction can be extracted.

Block Diagram



Block Diagram of Twitter Sentiment Opinion Mining [10]

Weekly Work For Project

Week 1	Defining scope for the project, idea brainstorming.
Week 2	Add all the necessary components using some libraries and modules.
Week 3	Collect dataset and code for twitter sentiment opinion mining.
Week 4	Evaluation and testing of the twitter sentiment opinion mining.

What is an IDE?

An integrated development environment (IDE) is software for building applications that combines common developer tools into a single graphical user interface (GUI). An IDE typically consists of:

- **Source code editor**: A text editor that can assist in writing software code with features such as syntax highlighting with visual cues, providing language specific auto-completion, and checking for bugs as code is being written.
- **Local build automation**: Utilities that automate simple, repeatable tasks as part of creating a local build of the software for use by the developer, like compiling computer source code into binary code, packaging binary code, and running automated tests.
- **Debugger**: A program for testing other programs that can graphically display the location of a bug in the original code.

An IDE allows developers to start programming new applications quickly because multiple utilities don't need to be manually configured and integrated as part of the setup process. Developers also don't need to spend hours individually learning how to use different tools when every utility is represented in the same workbench. This can be especially useful for onboarding new developers who can rely on an IDE to get up to speed on a team's standard tools and workflows. In fact, most features of IDEs are meant to save time, like intelligent code completion and automated code generation, which removes the need to type out full character sequences.[8]

NOTE: During my project I used Visual Studio Code for twitter sentiment opinion mining.

Python

Python is an object-oriented, high-level programming language. The main feature of python is that it comes with dynamic semantics. When its high-level built in data structures combine with dynamic typing and dynamic binding, they make it perfect for Rapid Application Development. Also it can be used as a scripting or glue language to connect existing components together.

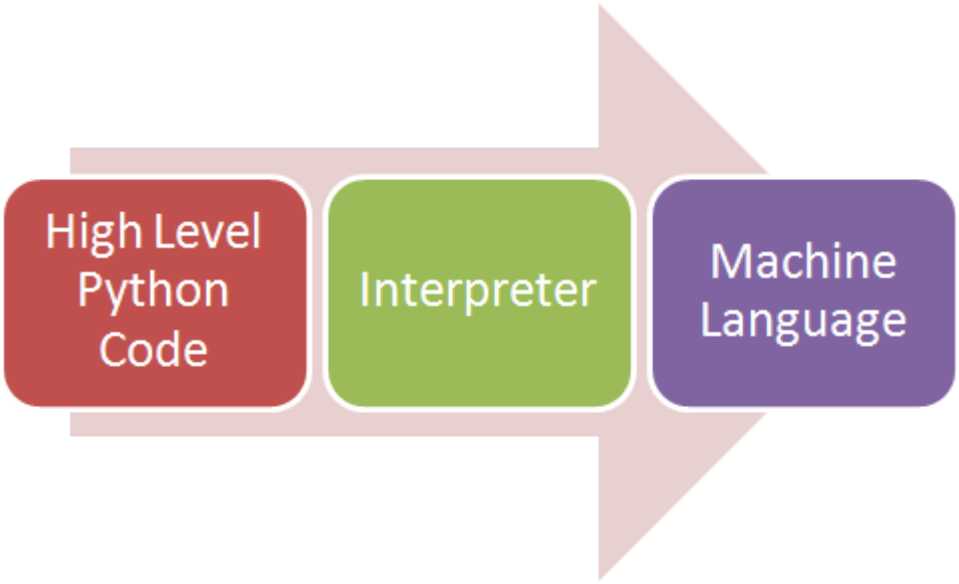
Python is an interpreted language. Because of the higher readability of its syntax, python is known as a very simple language compared to the other computer languages.

Why Python?

- Python is the most popular language due to the fact that it's easier to code and understand it.
- Python is an object-oriented programming language and can be used to write functional code too.
- It is a suitable language that bridges the gaps between business and developers.
- Subsequently, it takes less time to bring a Python program to market compared to other languages such as C#/Java.
- Additionally, there are a large number of python machine learning and analytical packages.
- A large number of communities and books are available to support Python developers.
- Nearly all types of applications, ranging from forecasting analytical to UI, can be implemented in Python.
- There is no need to declare variable types. Thus, it is quicker to implement a Python application.

How Does Python Work?

This image illustrates how python runs on our machines:



The key here is the Interpreter that is responsible for translating high-level Python language to low-level machine language.

The way Python works is as follows:

1. A Python virtual machine is created where the packages (libraries) are installed. Think of a virtual machine as a container.
2. The python code is then written in .py files
3. CPython compiles the Python code to bytecode. This bytecode is for the Python virtual machine.
4. When you want to execute the bytecode then the code will be interpreted at runtime. The code will then be translated from the bytecode into the machine code. The bytecode is not dependent on the machine on which you are running the code. This makes Python machine-independent.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

There are several toolkits that are available that extend python matplotlib functionality. Some of them are separate downloads, others can be shipped with the matplotlib source code but have external dependencies.

- **Basemap**: It is a map plotting toolkit with various map projections, coastlines, and political boundaries.
- **Cartopy**: It is a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities.
- **Excel tools**: Matplotlib provides utilities for exchanging data with Microsoft Excel.

- **Mplot3d**: It is used for 3-D plots.
- **Natgrid**: It is an interface to the natgrid library for irregular gridding of the spaced data.

There are various plots which can be created using python matplotlib. Some of them are listed below:

Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.

- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

NLTK

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).

It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

The Natural Language Toolkit is an open source library for the Python programming language originally written by Steven Bird, Edward Loper and Ewan Klein for use in development and education. It comes with a hands-on guide that introduces topics in computational linguistics as well as programming fundamentals for Python which makes it suitable for linguists who have no deep knowledge in programming, engineers and researchers that need to delve into computational linguistics, students and educators.

Scikit-Learn

Scikit-Learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!

The functionality that scikit-learn provides include:

- **Regression**, including Linear and Logistic Regression
- **Classification**, including K-Nearest Neighbors
- **Clustering**, including K-Means and K-Means++
- **Model selection**
- **Preprocessing**, including Min-Max Normalization

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

Building twitter sentiment opinion mining

Step-1: Import the Necessary Dependencies

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

Step-2: Read and Load the Dataset

```
# Importing the dataset
DATASET_COLUMNS=['target','ids','date','flag','user','text']
DATASET_ENCODING = "ISO-8859-1"
df = pd.read_csv('Project_Data.csv', encoding=DATASET_ENCODING, names=DATASET_COLUMNS)
df.sample(5)
```

Output:

	target	ids	date	flag	user	text
305165	0	1999924339	Mon Jun 01 21:04:24 PDT 2009	NO_QUERY	twentyred25	man my b-day is coming up but i dont know what...
673186	0	2247413241	Fri Jun 19 19:03:37 PDT 2009	NO_QUERY	belenneleb	@officialuti_ i need they to come back here. ...
573387	0	2209824277	Wed Jun 17 10:50:29 PDT 2009	NO_QUERY	TeenieWahine	@krystyn13 Sorry to hear that
246882	0	1982346860	Sun May 31 11:01:36 PDT 2009	NO_QUERY	BrianWCollins	@TheJoeLynch I've only seen 3 (Leon, 5th Eleme...
669112	0	2246153162	Fri Jun 19 17:10:15 PDT 2009	NO_QUERY	heidioftheopera	kinda feels bad for missing out on the Solstic...

Step-3: Exploratory Data Analysis

3.1: Five top records of data

```
df.head()
```

Output:

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

3.2: Columns/features in data

```
df.columns
```

Output:

```
Index(['target', 'ids', 'date', 'flag', 'user', 'text'], dtype='object')
```

3.3: Length of the dataset

```
print('length of data is', len(df))
```

Output:

```
length of data is 1048576
```

3.4: Shape of data

```
df. shape
```

Output:

```
(1048576, 6)
```

3.5: Data information

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048576 entries, 0 to 1048575
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype 
---  ------  --------------    ----- 
 0   target  1048576 non-null  int64 
 1   ids     1048576 non-null  int64 
 2   date    1048576 non-null  object
 3   flag    1048576 non-null  object
 4   user    1048576 non-null  object
 5   text    1048576 non-null  object
dtypes: int64(2), object(4)
memory usage: 48.0+ MB
```

3.6: Datatypes of all columns

```
df.dtypes
```

Output:

```
target     int64
ids        int64
date      object
flag      object
user      object
text      object
dtype: object
```

3.7: Checking for null values

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

Output:

```
0
```

3.8: Rows and columns in the dataset

```
print('Count of columns in the data is:  ', len(df.columns))
print('Count of rows in the data is:  ', len(df))
```

Output:

```
Count of columns in the data is:   6
Count of rows in the data is:   1048576
```

3.9: Check unique target values

```
df['target'].unique()
```

Output:

```
array([0, 4], dtype=int64)
```

3.10: Check the number of target values

```
df['target'].nunique()
```

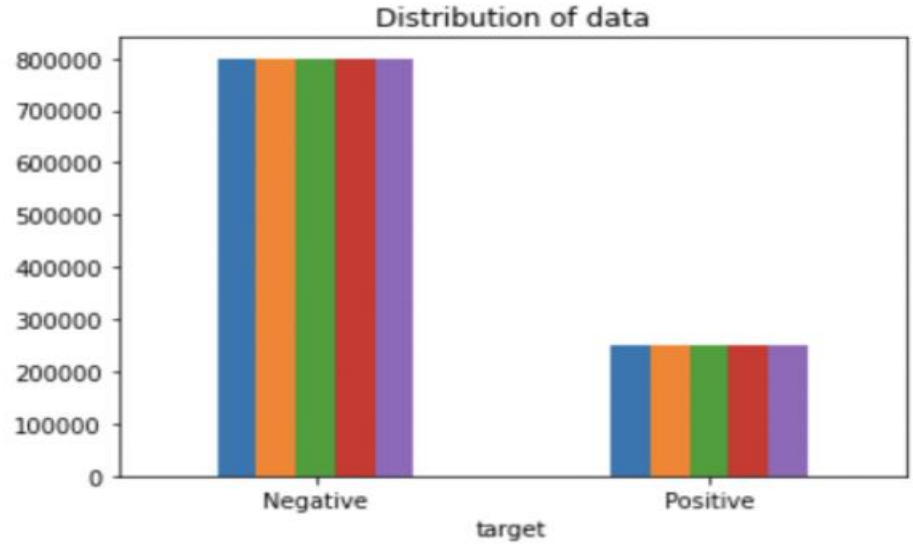
Output:

```
2
```

Step-4: Data Visualization of Target Variables

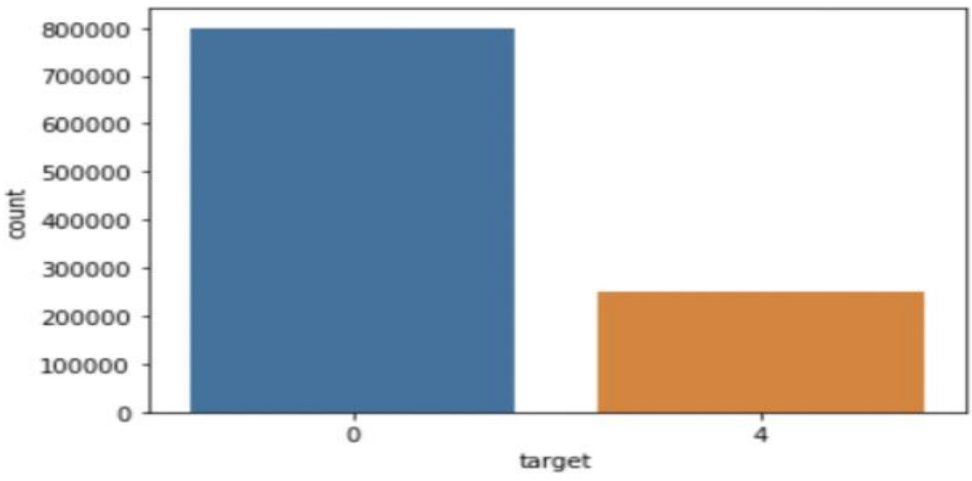
```
# Plotting the distribution for dataset.
ax = df.groupby('target').count().plot(kind='bar', title='Distribution of data',legend=False)
ax.set_xticklabels(['Negative','Positive'], rotation=0)
# Storing data in lists.
text, sentiment = list(df['text']), list(df['target'])
```

Output:




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

Output:



Step-5: Data Preprocessing

In the above-given problem statement, before training the model, we performed various pre-processing steps on the dataset that mainly dealt with removing stopwords, removing special characters like emojis, hashtags, etc. The text document is then converted into lowercase for better generalization.

Subsequently, the punctuations were cleaned and removed, thereby reducing the unnecessary noise from the dataset. After that, we also removed the repeating characters from the words along with removing the URLs as they do not have any significant importance.

At last, we then performed Stemming(reducing the words to their derived stems) and Lemmatization(reducing the derived words to their root form, known as lemma) for better results.[8]

5.1: Selecting the text and Target column for our further analysis

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

5.2: Replacing the values to ease understanding. (Assigning 1 to Positive sentiment 4)

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

5.3: Printing unique values of target variables

```
data['target'].unique()
```

Output:

```
array([0, 1], dtype=int64)
```

5.4: Separating positive and negative tweets

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

5.5: Taking one-fourth of the data so we can run it on our machine easily

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

5.6: Combining positive and negative tweets

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

5.7: Making statement text in lowercase

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

Output:

```
19995    not much time off this weekend, work trip to m...
19996                              one more day of holidays
19997    feeling so down right now .. i hate you damn h...
19998    geez,i hv to read the whole book of personalit...
19999    i threw my sign at donnie and he bent over to ...
Name: text, dtype: object
```

5.8: Defining set containing all stopwords in English.

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

5.9: Cleaning and removing the above stop words list from the tweet text

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

Output:

```
800000                love @health4uandpets u guys r best!!
800001    im meeting one besties tonight! cant wait!! - ...
800002    @darealsunisakim thanks twitter add, sunisa! g...
800003    sick really cheap hurts much eat real food plu...
800004                      @lovesbrooklyn2 effect everyone
Name: text, dtype: object
```

5.10: Cleaning and removing punctuations

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

Output:

```
19995    not much time off weekend work trip malmï¿½ fr...
19996                                  one day holidays
19997                    feeling right  hate damn humprey
19998    geezi hv read whole book personality types emb...
19999    threw sign donnie bent over get but thingee ma...
Name: text, dtype: object
```

5.11: Cleaning and removing repeating characters

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

Output:

```
19995    not much time of wekend work trip malmï¿½ fris...
19996                                      one day holidays
19997                          feling right hate damn humprey
19998    gezi hv read whole bok personality types embar...
19999    threw sign donie bent over get but thinge made...
Name: text, dtype: object
```

5.12: Cleaning and removing URLs

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

Output:

```
19995    not much time of wekend work trip malmï¿½ fris...
19996                                      one day holidays
19997                          feling right hate damn humprey
19998    gezi hv read whole bok personality types embar...
19999    threw sign donie bent over get but thinge made...
Name: text, dtype: object
```

5.13: Cleaning and removing numeric numbers

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

Output:

```
19995    not much time of wekend work trip malmï¿½ fris...
19996                                      one day holidays
19997                          feling right hate damn humprey
19998    gezi hv read whole bok personality types embar...
19999    threw sign donie bent over get but thinge made...
Name: text, dtype: object
```

5.14: Getting tokenization of tweet text

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

Output:

```
800000              [love, healthuandpets, u, guys, r, best]
800001    [im, meting, one, besties, tonight, cant, wait...
800002    [darealsunisakim, thanks, twiter, ad, sunisa, ...
800003    [sick, realy, cheap, hurts, much, eat, real, f...
800004                         [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

5.15: Applying stemming

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

Output:

```
800000              [love, healthuandpets, u, guys, r, best]
800001    [im, meting, one, besties, tonight, cant, wait...
800002    [darealsunisakim, thanks, twiter, ad, sunisa, ...
800003    [sick, realy, cheap, hurts, much, eat, real, f...
800004                         [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

5.16: Applying lemmatizer

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

Output:

```
800000              [love, healthuandpets, u, guys, r, best]
800001    [im, meting, one, besties, tonight, cant, wait...
800002    [darealsunisakim, thanks, twiter, ad, sunisa, ...
800003    [sick, realy, cheap, hurts, much, eat, real, f...
800004                         [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

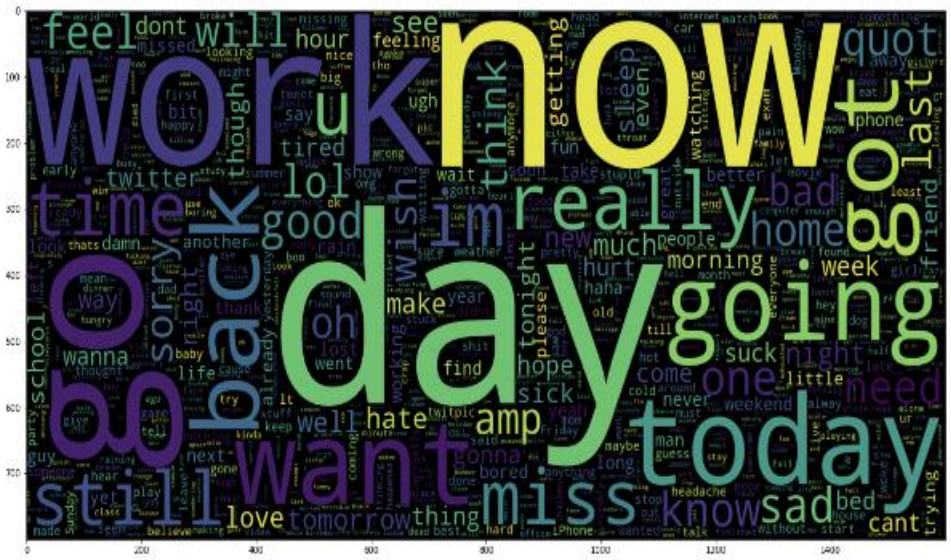
5.17: Separating input feature and label

```
X=data.text
y=data.target
```

5.18: Plot a cloud of words for negative tweets

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

Output:



5.19: Plot a cloud of words for positive tweets

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

Output:

Step-6: Splitting Our Data Into Train and Test Subsets

```
# Separating the 95% data for training data and 5% for testing data
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.05, random_state =26105111)
```

Step-7: Transforming the Dataset Using TF-IDF Vectorizer

7.1: Fit the TF-IDF Vectorizer

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

Output:

```
No. of feature_words:  500000
```

7.2: Transform the data using TF-IDF Vectorizer

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

Step-8: Function for Model Evaluation

After training the model, we then apply the evaluation measures to check how the model is performing. Accordingly, we use the following evaluation parameters to check the performance of the models respectively:

- Accuracy Score
- Confusion Matrix with Plot
- ROC-AUC Curve

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

Step-9: Model Building

In the problem statement, we have used three different models respectively:

- Bernoulli Naive Bayes Classifier
- SVM (Support Vector Machine)
- Logistic Regression

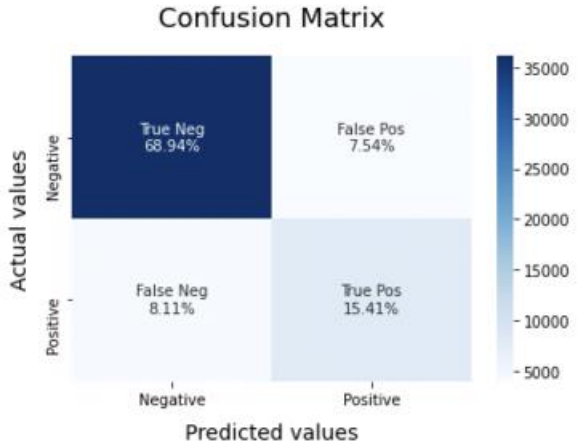
The idea behind choosing these models is that we want to try all the classifiers on the dataset ranging from simple ones to complex models, and then try to find out the one which gives the best performance among them.

9.1: Model-1

```
BNBmodel = BernoulliNB()
BNBmodel.fit(X_train, y_train)
model_Evaluate(BNBmodel)
y_pred1 = BNBmodel.predict(X_test)
```

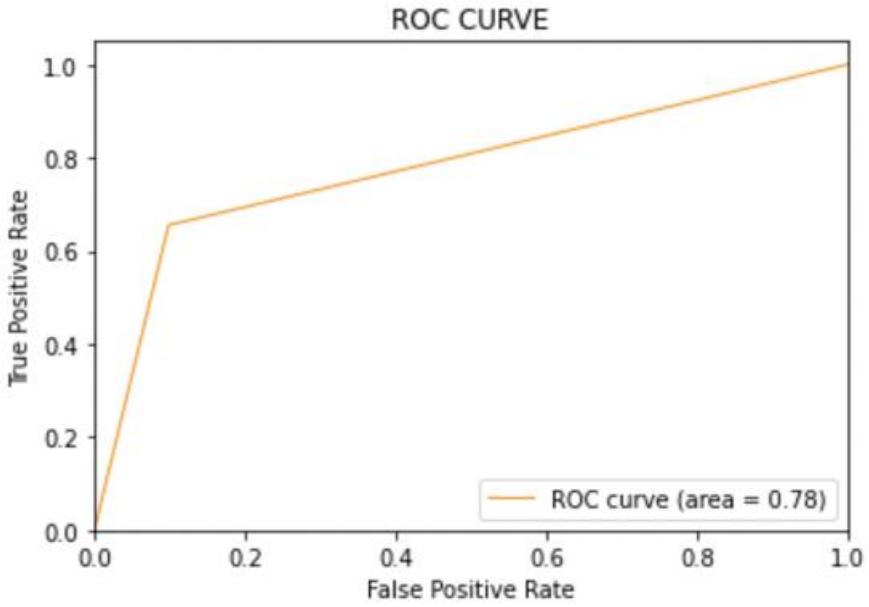
Output:

	precision	recall	f1-score	support
0	0.89	0.90	0.90	40097
1	0.67	0.66	0.66	12332
accuracy			0.84	52429
macro avg	0.78	0.78	0.78	52429
weighted avg	0.84	0.84	0.84	52429



9.2: Plot the ROC-AUC Curve for model-1

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred1)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

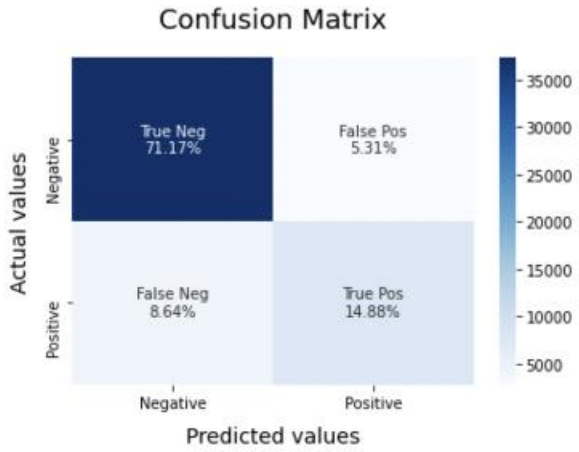


9.3: Model-2:

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

Output:

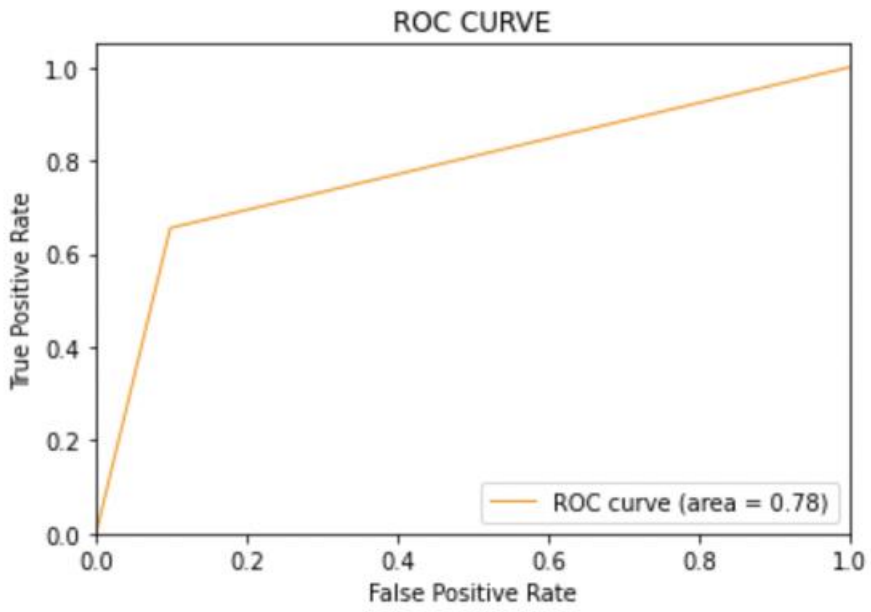
	precision	recall	f1-score	support
0	0.89	0.93	0.91	40097
1	0.74	0.63	0.68	12332
accuracy			0.86	52429
macro avg	0.81	0.78	0.80	52429
weighted avg	0.86	0.86	0.86	52429



9.4: Plot the ROC-AUC Curve for model-2

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

Output:

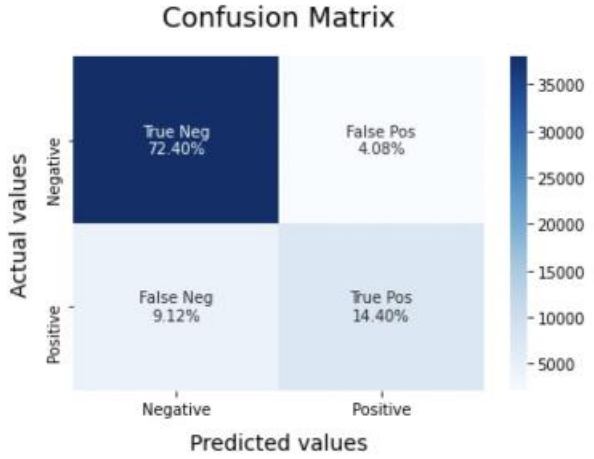


9.5: Model-3

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

Output:

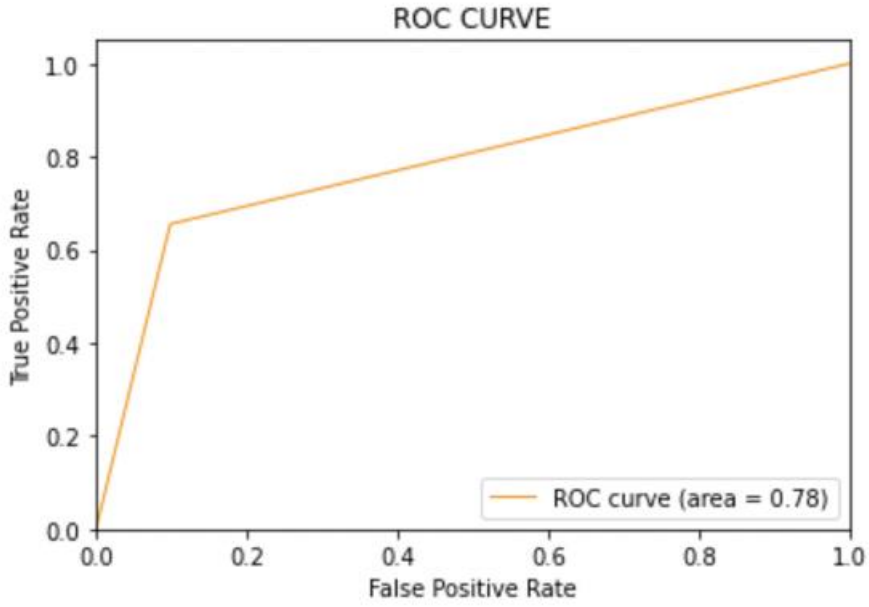
	precision	recall	f1-score	support
0	0.89	0.95	0.92	40097
1	0.78	0.61	0.69	12332
accuracy			0.87	52429
macro avg	0.83	0.78	0.80	52429
weighted avg	0.86	0.87	0.86	52429



9.6: Plot the ROC-AUC Curve for model-3

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

Output:



Step-10: Model Evaluation

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

Accuracy: As far as the accuracy of the model is concerned, Logistic Regression performs better than SVM, which in turn performs better than Bernoulli Naive Bayes.

F1-score: The F1 Scores for class 0 and class 1 are:
(a) For class 0: Bernoulli Naive Bayes (accuracy = 0.90) < SVM (accuracy =0.91) < Logistic Regression (accuracy = 0.92)
(b) For class 1: Bernoulli Naive Bayes (accuracy = 0.66) < SVM (accuracy = 0.68) < Logistic Regression (accuracy = 0.69)

AUC Score: All three models have the same ROC-AUC score.

We, therefore, conclude that the Logistic Regression is the best model for the above-given dataset.

In our problem statement, **Logistic Regression** follows the principle of **Occam's Razor,** which defines that for a particular problem statement, if the data has no assumption, then the simplest model works the best. Since our dataset does not have any assumptions and Logistic Regression is a simple model. Therefore, the concept holds true for the above-mentioned dataset.

Testing

Testing is the process of exercising software with the intent of finding errors and ultimately correcting them. The following testing techniques have been used to make this project free of errors.

Content Review

The whole content of the project has been reviewed thoroughly to uncover typographical errors, grammatical error and ambiguous sentences.

Navigation Errors

Different users were allowed to navigate through the project to uncover the navigation errors. The views of the user regarding the navigation flexibility and user friendliness were taken into account and implemented in the project.

Unit Testing

Focuses on individual software units, groups of related units.

- Unit – smallest testable piece of software.
- A unit can be compiled /assembled / linked/loaded; and put under a test harness.
- Unit testing done to show that the unit does not satisfy the application and /or its implemented software does not match the intended designed structure.

Integration Testing

Focuses on combining units to evaluate the interaction among them

- Integration is the process of aggregating components to create larger components.

• Integration testing done to show that even though components were individually satisfactory, the combination is incorrect and inconsistent.

System testing

Focuses on a complete integrated system to evaluate compliance with specified requirements (test characteristics that are only present when entire system is run)

• A system is a big component.

• System testing is aimed at revealing bugs that cannot be attributed to a component as such, to inconsistencies between components or planned interactions between components.

• Concern: issues, behaviours that can only be exposed by testing the entire integrated system (e.g., performance, security, recovery)each form encapsulates (labels, texts, grid etc.). Hence in case of project in V.B. form are the basic units. Each form is tested thoroughly in term of calculation, display etc.

Regression Testing

Each time a new form is added to the project the whole project is tested thoroughly to rectify any side effects. That might have occurred due to the addition of the new form. Thus regression testing has been performed.

White-Box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and

determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

API testing (application programming interface) – testing of the application using public and private APIs.

Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once).

Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies.

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for:

Function coverage, which reports on functions executed Statement coverage, which reports on the number of lines executed to complete the test 100% statement coverage ensures that all code paths, or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly.

Black-box testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The tester is only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

Alpha Testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

Beta Testing

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Learning's and Value Addition

A project is a learning experience of its own kind. Neither is it spoon-fed school learning, nor pressure filled workload. It is in between; I not only learn the basics of work life but also the skills required for a brighter professional career.

During my project I had learned lots of thing and few of them I'm mentioning here:

Teamwork: It is important because it enables your team to share ideas and responsibilities, which helps reduce stress on everyone, allowing them to be meticulous and thorough when completing tasks.

Problem Solving Skills: It allow you to find candidates who are cognitively equipped to handle anything their jobs throw at them. Problem solvers can observe, judge, and act quickly when difficulties arise when they inevitably do.

Work Ethics: Workplace ethics ensures positive ambience at the workplace. Workplace ethics leads to happy and satisfied employees who enjoy coming to work rather than treating it as a mere source of burden.

Adaptability Skills: It expands your capacity to handle change, no matter how serious it might be. Instead of throwing away your energy trying to change your circumstance, you will change yourself right

from within, thus making you thrive in whatever situation you find yourself.

Communication Skills: It is fundamental to the existence and survival of humans as well as to an organization. It is a process of creating and sharing ideas, information, views, facts, feelings, etc. among the people to reach a common understanding.

Responsibility: Each step we take towards being responsible and productive helps to raise our self-esteem and our relationships with friends, family and co-workers improve ten-fold.

Time Management: It helps you prioritize your tasks so that you ensure you have enough time available to complete every project. The quality of your work increases when you're not rushing to complete it ahead of a fast-approaching deadline.

Theoretical v/s Practical Knowledge

Practical knowledge is knowledge that is acquired by day-to-day hands-on experiences. In other words, practical knowledge is gained through doing things; it is very much based on real-life endeavours and tasks. On the other hand, theoretical knowledge teaches the reasoning, techniques, and theory of knowledge. While practical knowledge is gained by doing things, theoretical knowledge is gained, for example, by reading a manual.

While theoretical knowledge may guarantee that you understand the fundamental concepts and have know-how about how something works and its mechanism, it will only get you so far, as, without practice, one is not able to perform the activity as well as he could.

During this project period, I had only theoretical knowledge about some programming language and never build any thing like this before. But during this project period mentor assigned me a task for Twitter Sentiment Opinion Mining and then I try to make that first I made mock-up and yes, it is my first mock up I had made ever. Then some research is important to work practically on a project. So, I did that and share that with our team members and get some review. We told each other some of our mistakes and then we correct them accordingly because this detector is used by many people, so some feedback is important, and we really get some positive feedback as well as some negative from our mentors too and all that are honest. So, on the negative feedback, we research again for that and end up with good user experience. By doing this we got lots of practical knowledge with theoretical knowledge.

Theoretical and practical knowledge are interconnected and complement each other — if one knows exactly HOW to do something, one must be able to apply these skills and therefore succeed in practical knowledge.

LIMITATIONS

1. TWITTER USER DEMOGRAPHICS ARE SKEWED

This is probably not a surprise to anyone, but it is something often brushed over when people present their conclusions from Twitter data. First of all, only about 22% of Americans use Twitter.[9] So we can hardly claim to encompass the population. Still, it could work as a representative sample, right?

Sadly, this isn't really the case. Twitter's users do not represent the general population of the world, or even the population of their particular regions. They tend to be younger, more left-leaning, and more affluent than the overall population. Just think about it when was the last time your grandfather used Twitter? This means that if you're doing research on a possible customer base or political issue, focusing on Twitter alone can lead you to somewhat erroneous conclusions.

2. RETWEETS

A major part of Twitter is the ability to 'retweet' a specific tweet, often with your own commentary added to it. Retweets are often understood as endorsements of a position, but it is not rare to see a tweet retweeted along with a criticism of it.

So if we are using Twitter data, we have to decide how to treat retweets do we count retweets as agreement, so that multiple retweets of an idea make that idea weigh more heavily? Or should we only count a tweet once, no matter how often it is retweeted? The first option will fail to fully account for those retweets that actually function as criticism, and the second will flatten the conversation so that the influential retweeted statements are kept on the same level as random comments from the periphery. Either choice leaves something to be desired, but a choice has to be made.[12]

3. TEXTUAL ODDITIES

Twitter requires users to stick to a character limit. It is also used overwhelmingly on mobile devices, where users have access to an autocorrect that will fix their spelling for them. Both of these can create oddities in the way tweets are written that would not be present in normal language. So, for example, if your GloVe vectors are trained on newspapers, it's not clear that they'll work as well on tweets.

The character limit on tweets often means users abbreviate words. These words tend to be stop words, so this may not be a particularly problematic issue. But when conversations focus on specific topics, people often invent acronyms on the fly or omit words whose presence can be inferred by humans. These acronyms and omissions could possibly be detected by a machine learning technique, but that's a whole other layer that complicates the matter.

In addition, Twitter data is going to have some errors due to autocorrect (and normal spelling errors too, of course). Unlike spelling errors, autocorrect errors are harder to detect since the problematic word is a real word, just used in the wrong way. Again, there are certainly tools that can correct for this, but using them adds an additional layer of complexity to any Twitter-based research project.

4. SARCASM & JOKES

Detecting sarcasm and humor accurately is something of a holy grail for NLP researchers. Humans on their own aren't amazingly good at detecting sarcasm in particular, especially when the only clues are a few lines of text. But sarcasm is a common part of conversations, especially online conversations. If you take Twitter data and don't account for sarcasm, you are accepting a certain level of noise that can adversely affect your results. Jokes, too are common throughout Twitter, and if we don't take them to account, they will be another source of noise in our data.

CONCLUSION

Twitter Sentimental Opinion Mining helps us preprocess the data (tweets) using different methods and feed it into ML models to give the best accuracy.

Key Takeaways

- Twitter Sentimental Opinion Mining is used to identify as well as classify the sentiments that are expressed in the text source.
- Logistic Regression, SVM, and Naive Bayes are some of the ML algorithms that can be used for Twitter Sentimental Opinion Mining.

FUTURE SCOPE

- **Multi-lingual support:** Due to the lack of multi-lingual lexical dictionary, it is currently not feasible to develop a multi-language based sentiment analyser.
  Further research can be carried out in making the classifiers language independent. The authors have proposed a sentiment analysis system with support vector machines, similar approach can be applied for our system to make it language independent.

- **Interpreting Sarcasm:** The proposed approach is currently incapable of interpreting sarcasm. In general sarcasm is the use of irony to mock or convey contempt, in the context of current work sarcasm transforms the polarity of an apparently positive or negative utterance into its opposite.[3]
  The main goal of this approach is to empirically identify lexical and pragmatic factors that distinguish sarcastic, positive and negative usage of words.

- Analysing sentiments on emoji/smileys.

REFERENCES

[1] A.Pak and P. Paroubek. „Twitter as a Corpus for Sentiment Analysis and Opinion Mining". In Proceedings of the Seventh Conference on International Language Resources and Evaluation, 2010, pp.1320-1326.

[2] R. Parikh and M. Movassate, “Sentiment Analysis of User-GeneratedTwitter Updates using Various Classi_cation Techniques",CS224N Final Report, 2009.

[3] Go, R. Bhayani, L.Huang. “Twitter Sentiment ClassificationUsing Distant Supervision". Stanford University, Technical Paper,2009.

[4] L. Barbosa, J. Feng. “Robust Sentiment Detection on Twitterfrom Biased and Noisy Data". COLING 2010: Poster Volume,pp. 36-44.

[5] Bifet and E. Frank, "Sentiment Knowledge Discovery inTwitter Streaming Data", In Proceedings of the 13th InternationalConference on Discovery Science, Berlin, Germany: Springer,2010, pp. 1-15.

[6] Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, “Sentiment Analysis of Twitter Data", In Proceedings of the ACL 2011Workshop on Languages in Social Media,2011 , pp. 30-38.

[7] Dmitry Davidov, Ari Rappoport." Enhanced Sentiment Learning Using Twitter Hashtags and Smileys". Coling 2010: Poster Volumepages 241{249, Beijing, August 2010.

[8] Po-Wei Liang, Bi-Ru Dai, “Opinion Mining on Social MediaData", IEEE 14th International Conference on Mobile Data Management,Milan, Italy, June 3 - 6, 2013, pp 91-96, ISBN: 978-1-494673-6068-5.

[9] Pablo Gamallo, Marcos Garcia, “Citius: A Naive-Bayes Strategyfor Sentiment Analysis on English Tweets", 8th InternationalWorkshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland,Aug 23-24 2014, pp 171-175.

[10] Neethu M,S and Rajashree R,” Sentiment Analysis in Twitter using Machine Learning Techniques” 4th ICCCNT 2013,at Tiruchengode, India. IEEE – 31661.

[11] P. D. Turney, “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews,” in Proceedings of the 40th annual meeting on association for computational linguistics, pp. 417–424, Association for Computational Linguistics, 2002.

[12] J. Kamps, M. Marx, R. J. Mokken, and M. De Rijke, “Using wordnet to measure semantic orientations of adjectives,” 2004.

[13] R. Xia, C. Zong, and S. Li, “Ensemble of feature sets and classification algorithms for sentiment classification,” Information Sciences: an International Journal, vol. 181, no. 6, pp. 1138–1152, 2011.

[14] ZhunchenLuo, Miles Osborne, TingWang, An effective approachto tweets opinion retrieval", Springer Journal onWorldWideWeb,Dec 2013, DOI: 10.1007/s11280-013- 0268-7.

[15] Liu, S., Li, F., Li, F., Cheng, X., &Shen, H.. Adaptive cotraining SVM for sentiment classification on tweets. In Proceedings of the 22nd ACMinternational conference on Conference on information & knowledgemanagement (pp. 2079-2088). ACM,2013.