# Cyber Bullying Recognition System for Facebook, Twitter Using ML Algorithms for Sentiment Analysis

*by* 17bce2380 17bce2380

**Submission date:** 10-Jun-2021 10:54AM (UTC+0530)
**Submission ID:** 1603887127
**File name:** 17BCE2380_17BCE2389_17BCE2393_Capstone_1.pdf (2.96M)
**Word count:** 6657
**Character count:** 35901

**FINAL REVIEW**

# Cyber Bullying Recognition System for Facebook, Twitter Using ML Algorithms for Sentiment Analysis

*Submitted in partial fulfillment of the requirements for the degree of*

**Bachelor of Technology**

in

**Computer Science and Technology**

*by*


| ALOK SINHA | SUNNY AGRAWAL | BIBEK SINGH |
|---|---|---|
| 17BCE2380 | 17BCE2389 | 17BCE2393 |


**Under the guidance of**

**Dr. ANURADHA G**

**Scope**

**VIT, Vellore**

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

June, 2021

# DECLARATION

I hereby declare that the thesis entitled **"Cyber Bullying Recognition System for Facebook, Twitter Using ML Algorithms for Sentiment Analysis"** submitted by **Alok Sinha, Sunny Agrawal, Bibek Singh** for the award of the degree of *Bachelor of Technology in CSE* to VIT is a record of bonafide work carried out under the supervision of **Dr. Anuradha G.**

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place : Vellore

Date :

**Signature of the Candidate**

# CERTIFICATE

This is to certify that the thesis entitled **"Cyber Bullying Recognition System for Facebook, Twitter Using ML Algorithms for Sentiment Analysis"** submitted by **Alok Sinha (17BCE2380), Sunny Agrawal (17BCE2389), Bibek Singh (17BCE2393)**, VIT, for the award of the degree of *Bachelor of Technology in CSE*, is a record of bonafide work carried out by him / her under my supervision during the period, 2020 to 2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute orany other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :

**Signature of the Guide**

**Internal Examiner**

**External Examiner**

**Dr. S Vairamuthu**

**Head of the Department (HOD)**

**Department of Software Systems**

## Acknowledgements

It is great honor to thank Dr. Anuradha G for her invaluable contributions, capable leadership, support, fulsome collaboration, and insights during my Capstone Project. My relationship with her isn't only academic; it's a fantastic experience for me to collaborate with an academic and specialist in the area of Machine Learning.

I'd like to thank Dr. G. Vishwanathan (Chancellor), Mr. G.V Selvam (Vice President), Dr. Rambabu Kodali (Vice Chancellor), and Prof. S Narayanan (Pro Vice Chancellor) for providing me with a working atmosphere and inspiration during the course.

In a quavering voice, I say thanks to every teaching staff and members working for university with enthusiasm and timely messages of support for our team, that triggered the acquisition of the necessary knowledge to successfully complete my project. I'd want to express my gratitude to my parents for their unwavering support.

It gives me great pleasure to express my gratitude to my friends who convinced and supported me to take on and accomplish this project. Last but not least, I want to show my gratitude and thanks to everyone who has assisted me in the successful completion of this project, whether directly or indirectly.

# Executive Summary

Cyberbullying can cause huge mental pain and tension. With having similarities as the pre-existing person who survived the situation form of annoying, and bullying while they were teens to know-how nervousness, fear, unhappiness, and not having assurance feels like.

Society has developed from numerous points of view but then remained the same in numerous others. Just the techniques have changed. They likewise may meet mental signals, along with the fight academically. So, here the situation centers of social-harassment also experience about exceptional outcomes and negative sentiments. Facebook comments and Twitter tweets and statuses will be the main concern of this project. Facebook in contrast to Instagram and twitter is focused on this project due to its overall use by all kinds of users from across the world. The quantity of the users makes it easier to collect the corpus for our research. It is having greater amount in text words and also the amount for bullying cases in Facebook is way more than compared to other social Medias. In this project presenting sentiment analysis of Twitter and Facebook comments using Naive Bayes Classifier, logistic regression, Random forest, Decision tree and Support Vector Machine (SVM).

In this project, python and web technology are utilized for implementing cyberbullying recognition system. First it will search and find out the dataset and will download it for train the model. Then will preprocess the data and then transferred to Tf-Idf after downloading the dataset first. Now in the presence for the methods like Naïve Bayes Classifier, SVM, Choice Tree, LR, RF; It will train the dataset and generate model separately. After that will be developing a web application using FLASK framework. After that it will fetch the real time comments from Twitter and then applying generated model to these fetched comments and check the texts are cyberbullying or not. Here in this project, html, css, javascript is used as Fronted, python as backend.

## Table of Contents

## List of Figures

## List of Tables

## Abbreviations

| | |
|---|---|
| ML | Machine Learning |
| NLP | Natural Language Processing |
| SVM | Support Vector Machine |
| RF | Random Forest |
| POS | Part-of-speech |
| CNN | Convolutional Neural Network |
| RNN | Recurrent Neural Network |
| ANN | Artificial Neural Network |
| BPNN | Back Propagation Neural Network |

## Symbols and Notations

| | |
|---|---|
| % | Percentage |
| H/W | Hardware |
| S/W | Software |

## Abstract

Cyberbullying has been around for some time currently, however individuals have quite recently started understanding that difficulty. Society has developed from numerous points of view but then remained the same in numerous others. Just the techniques have changed. In this project, it will take a gander at a portion of the causes behind cyber-bullying and have prepared a statistical analysis of various algorithms used in the cyber-bullying detection systems. So many techniques contributing in cyber-bullying detection, mainly ML techniques along with NLP methods. Sentimentality examination has popularized due to the availability of abundant opinions that resides in social networks such as Twitter, Facebook. In this project presenting sentiment analysis of Twitter and Facebook comments using Naive Bayes Classifier, LR, RF, Decision tree and SVM. The essential then basic thought of this project is that, realizing how individuals feel certain Twitter tweets and Facebook comments can be utilized for classification.

# 1. Introduction

## 1.1 Theoretical Background

Cyberbullying can cause huge mental pain and tension. Which is almost as similar as some other survivor with troublesome, bullied children's experience nervousness, terror, sadness, and having no enthusiasm. The children likewise face different types of symptoms and it can be seen in the behavior of the children who experience about exceptional outcomes also non-positive sentiments. Facebook comments and Twitter tweets and statuses will be the main concern of this project. Facebook in contrast to Instagram and twitter is focused on this project due to its overall use by all kinds of users from across the world. The quantity of the users makes it easier to collect the corpus for our research. It is having more arguments and the different types of bullying cases in Facebook is way more than compared to other social Medias. The need for the research aroused from the fact that variety of algorithms were used to detect cyberbullying across social Medias. But the use of these algorithms used to differ across different situations. Thus, the main focus of this project would be to provide a statistical comparison of the efficiency and accuracy of ongoing algorithms that are currently used in Cyber-bullying detection systems. Consequently, various algorithms are compared in order of their efficiency. The algorithms that are mostly used were selected which are Naive Bayes Classifier, Logistic Regression, Random Forest, Decision Tree and SVM (Support Vector Machine)

I. Classification of Naive Bayes

This classifier has one of the differentiation method dependent on the theorem given by Bayes through the hypothesis of independence among medium. It accepts in existence with a specific idea which is disconnected in existence with some additional methods. The method can be managed erudition procedure that depends on his methods then utilized which is aimed at taking care of arrangement issues. The classifier is predominantly used for transcript organization which incorporates more types of information. The system is straightforward also best Sorting procedures that assistances for making the quick ML technique which will do speedy calculations.

II. Logistic regression

LR according to the regression analysis is good to the direction of the sensitive mutable is 0 and 1. That all of the regression model, the study, the LR model is as a predictive analysis. The logistic regression model was used to describe the data, and to refine the relationship between the result of the binary variables, and the minimum of the imaginary, the number, scope quite the independent variable. The logistic regression model is generally quick, with contradictions additional approaches of classification, such as the core computing lab, or a collaborative approach also, it has a similar problem as a straight, even, because of both of these methods are very easy to complex the relations among the variables. Finally, a LR model will be general - ineffective when the decision border is not lined.

III. Random forest

These are the technique used in organization, reversion and different methods which work with the making of the different types of algorithm during preparation with mean prediction of individual trees. These approaches will be considered as knowledge systems which build bunch of differentiators rather than single differentiators, and afterward characterize fresh information focuses captivating chance of finding. Normally utilized. Accidental timberland has sort in overseen AI procedure dependent on ensemble learning. Ensemble learning is a kind of realizing where you join various types of algorithms or a similar algorithm multiple occasions to frame an all the more powerful prediction model. The random woods algorithm joins multiple algorithms of a similar sort for example multiple decision trees, bringing about a woodland of trees, henceforth the name "Random Forest". The random backwoods algorithm can be utilized for both regression and classification tasks. RF classifier can be portrayed as the collection of tree-structured classifiers. It is an advanced rendition of bagging to such an extent that randomness is added to it. Rather than splitting every node utilizing the best split among all variables, Random Forests parts every node utilizing the most popular type used in indicators accidentally chooses the system.

IV. Decision tree

It's a supportive environment in which they use a different type of, choose a higher number of potential outcomes, including a random result. The tree can be "trained", the basis of a set of parameters, which depend on the evaluation of the tests, the value of the attribute. The procedure will be recurrent for the derived subsection with a recursive manner called recursive dividing. Recursion stops if the lower node has the same value of the time variable, or split to a further increase in the value of the predictions. To build a decision tree differentiator does not require the information on this topic, or to the settings, then it is advisable to search for and discover information. A decision tree can be work with a lot. For a general class of decision trees are very thorough in their work. A decision tree is a common approach to work, to study the details of the organization. The examples are grouped together on the basis of the origin node of the validation, the qualities are defined at this topic to fall down on the branches of a tree, with a connection to the attribute value. This process is then repeated for the subtree to the roots of the newest part. A choice tree, and I hold some of the everyday, depending on whether the game is appropriate for the game and return to the classifications that are associated with the specific sheet of paper.

V. SVM

It is called as the learned and trained ML method that uses sorting and reversion. Also the fact that regression issues also its best suited for classification. The goal of Support Vender Machine procedure will be finding a thing called hyperplane which is stated to be in N-dimensional part which definitely categorizes for information ideas. Here, measurement for hyperplane rest on the numerous functions. Assuming with amount for contribution type is 2, the hyperplane turns into 2-D style. This will get very tough to think when types is more than three. It is for the most part helpful in non-linear separation issues. Essentially put the kernel, this is very hard info to send to see if the features isolate the info on the basis of the outcome.

### 1.2 Motivation

Cyberbullying is the utilization of electronic communication to bully an individual by sending dangerous text utilizing social media, texting or through digital texts. Cyberbullying can be exceptionally harming to young age people. It can prompt anxiety, sorrow, and even suicide. Additionally, whenever things are circled on the Internet, they may never disappear, resurfacing at later occasions to recharge the torment of cyberbullying. So beat these issues detecting the cyberbullying is vital in now daily which will assist with stopping cyberbullying on social media networks.

### 1.3 Aim of the proposed Work

The motivation to complete this project was to detect the cyberbullying technique which will help to improve and monitoring the cyberbullying on social sites. In this project first fetching the tweets from twitter accounts as well as Facebook comments and preprocess the twits and images and applying generated model will detect the cyberbullying or not.

### 1.4 Objective(s) of the proposed work

- Collecting the dataset of cyberbullying words and preprocessing it and then applying different machine techniques.
- Generating different ML algorithms Model.
- Fetch the comment from Facebook account, tweets from Twitter account and preprocess it.
- Apply generated model on the fetched comment, tweets and get final output cyberbullying or not.

### 1.5 Key Challenges of Proposed Algorithms

The system called Naive Bayes is having too much assumption with the types to be on their own which is not obvious, all things considered, applications. It has Data scarcity and likewise has probabilities in waste of correctness. It requires no Occurrence for example in an event that the class for any uncompromising variable isn't understood for preparation information at that time prototype allocates no likelihood for it class then a forecast will not remain created.

It can't take care of other issues by LR meanwhile its choice superficial remains lined. LR can be additionally will not termed as the most effective methods as effortlessly outdid with additional multifaceted. Other difficulty can be called as an appropriate performance for information. That is the LR cannot be the best tool to use. Result is separate LR can simply forecast a definite result. This will likewise a Procedure which is recognized for its susceptibility to fit over the method.

The most serious subjects in ML is called more accurate, though it is likely to happen thanks towards the random backwoods differentiator. In the event which around remain enough, this differentiator will not be good for the system. One of foremost restriction for accidental woodland can be said a great amount with parts will style this system excessively lethargic along with unsuccessful of ongoing guesses. Overall, the procedures will be quick toward learning, however very delayed for making forecasts after it remain trained. A more exact prediction requires more trees, which brings about a leisurelier classical. The most certifiable requests is quick sufficient however there can certainly be situations where run-time performance is important and different methodologies will used as chosen. Accidental backwoods are a predictive modeling tool with a non- expressive means, sense in case it is finding for some explanation for the relation for the info, different methodologies can be easy and nice to work.

In Decision Trees, a little change in the data can cause a large change in the construction of the decision tree causing instability. For a Decision tree sometimes, calculation can go undeniably more complex compared to different algorithms. Decision tree mostly involves very higher time to train the given model. Decision tree training is generally expensive as complexity and time taken is more. Decision Tree technique is sufficient for trying regression model and predicting continuous qualities.

SVM algorithm isn't appropriate for large data sets. SVM does not perform well indeed, when the data set has more noise for example target classes are overlapping. In one of the situations where number of features for each data point is more than or exceeds the number of training data sample, then SVM will underperform. As the support vector classifier

works by putting data points, above and underneath the classifying hyper plane there is no probabilistic explanation for the classification

‘

# 2. Literature Survey

## 2.1 Survey of the Existing Models/Work

Scores after the internet shop website called flipkart.com will be examined, in light of parts for item the score will be categorized in +ve, okay or -ve. Following planned effort is examined in utilizing ML model named RF and reenacted through utilizing SPYDER as a methods. The system's correctness, exactness and memory will be determined in the two methods called RF with SVM procedure and afterward correctness contrast will be made for the 2 procedures. Crowds for client part for the response regarding the use in communal platform, it assists the supplier with welling the users to find out with the emotions regarding with item.

It focuses on utilizing ML methods with the use of sentiment analysis of the posts on the micro blogging site - Twitter. Four significant machine learning techniques have been utilized, specifically Decision Tree Classifier, Logistic Regression, Support Vector Machine and Neural Network. The outcomes are analyzed and the upsides of different techniques in comparison to others have been talked about. In choice tree classifier, parameters max_depth and min_sample_split have been fluctuated. The maximum accuracy accomplished utilizing choice tree classifier is 0.555. This variety of accuracy with parameters in the event of choice tree classifiers is depicted.

Cyberbullying is one of the major issue plaguing social networking websites in a few different ways especially developed between adolescents. Along these lines, this investigation will investigate the secret patterns of MomoChallenge an occasion and form of cyberbullying particularly developed on Twitter. Social network analysis(SNA) will be utilized to examine the Twitter network of MomoChallenge. Information is gathered then examined using NodeXL

This Scheme intended for forestalling internet harrasment occurrences, so the noticing along with hesitant how they work. This utilizes NLP for classify and handle other language arguments. At that point Machine Learning techniques are utilized to classify bullying substance. They tell themselves the best way to arrange the information subject to

similarities and contrasts between information. At the point the overseen along with unverified teachings will be merged composed through means of marked along with the non- labeled information.

(Viktor Golem, Laden Karan and Jan ˇSnajder,2018) assemble a framework which will check that the provided information is hostile, or not hostile. They can handle the assignment with the help of the algorithms termed as LR and SVM algorithms like old CNNs along with LSTMs. To perform the one which is good among the two, it performed exercising in the use of mixture of both narrow along with profound algorithms.

(Ivan Habernal, Tomas ptacek and Josef Steinberger,2013) aim to form document-level sentimentality examination based for all kinds of techniques in the ML. One of the datasets is created using social media has 100k status. Social media is analyzed focusing on Twitter and Facebook. Studying twitter and Facebook with actual relaxed linguistic welfaresin the connecting original skills, like emoji character n-grams, POS and POS ratio, or word shape etc.

(Befoul Haidari, Maroon Shamoun and Serhrchni, 2018) calculated aimed at stopping internet based harassments and assaults, with the help of noticing along with discontinuing the work. Then Machine Learning methods can cast-off to categorize harassment content. They show them-selves how to order the information dependent on likenesses and contrasts between information. At the point all the overseen and unverified teachings remain consolidated composed of utilizing marked then not-labeled information.

(Dublin, Ireland, 2014) suggested the CNN scheme which abuses after words with the condemn equal data in order for the achieve of notion examination of short messages. The proposed organize, called Char-SCNN, and utilizes 2 practised deposits or the separate significant highlights after arguments with verdicts. Proposed system be able to without much of a stretch investigate the wealth of word embeddings delivered by solo pre-preparing. They perform tests that show the viability of CharSCNN for supposition

examination of writings from two areas: film audit sentences; and Twitter messages. CharSCNN accomplishes best in class results for the two areas.

(V Kharde, 2016) utilize different component extraction strategy. They utilized the structure where the pre-processor will be used in the not surely made lines that will be the fitting toward comprehend. Additionally, dissimilar ML strategies prepare all the information that include paths and afterward examination proposals an enormous arrangement for equivalents then comparability that gives extremity for substance. They give an overview and near investigation of existing procedures for conclusion mining including AI and dictionary-based methodologies, along with cross area and cross-lingual strategies and some assessment measurements.

(F Del Vigna12, 2017) presented the principal of hate speech classifier for Italian writings. Thinking about a binary grouping, the classifier accomplished outcomes similar with those got in generally researched supposition examination for Italian language. Empowered by such encouraging result, they leave for future work the refinement of the classifier results while thinking about differentiation among hate levels and among different types of hate speech. They are developing the explanation process, both to build the corpus size and to gather more comments for a single comment. They are trying new explanation techniques, assessing the between annotator understanding or approving the explanation on the various degrees of hate speech.

(Liew Choong Hon, Kasturi Dewi Varathan,2015) make a web-based application, for example the Cyberbullying Detection Framework on Twitter. The usage of the Cyberbullying Detection System on Twitter depends on PHP and HTML with the MySQL and Twitter API. This framework will identify cyberbullying related tweets that have matching keyword from the database. This framework tells overall the general procedure on how the cyberbullying-related tweets can be recognized and alarms the NGOs, and hence action taken by police headquarters through email reports, likewise alert the cyberbullying client's or on the other hand the system administration in checking their

cyberbullying exercises. At first, the client needs to login into the framework and then all other steps are taken by the system.

(Mohammed Nazrul Islam Arif, Sarwar Hussain Paplu, Prof. Dr. Karsten Berns, 2019) recognize the emotions present in the communication or the emotions of the involved users to make those interactions more humanly. This study summarizes different approaches that can be used to analyze the emotion from the text and recognize the emotion between Text-Based Communications. Approaches like finding subjective feeling through introspection, Word categorization, Kinetic Typography to convey emotion, Meta-analysis on mood induction, Emotion expression through text-based communication, Sentimental analysis are discussed.

(Pinkesh Badjatiya, Shashank Gupta, 2017) define the job for finding out that the posts in the social sites will be considered as harassing or not. It is difficult for usual linguistic concepts will make it works really tough. They do the harsh scientific methods along with the use of the methods to read and find out the harassing terms which cannot be handled Their analyses for the standard information of 16 thousand commented on posts uses the convolution methods and strategy for beat cutting edge burn parts of the sentence.

## 2.2 Summary/Gaps identified in the Survey

| S. No | Algorithms and Techniques Used | Cons | Pros |
|---|---|---|---|
| [1] | An unsupervised approach, growing hierarchical self-organizing map, GHSOM network, Clustering algorithms, random subsampling technique, syntactic and semantic analysis | Might get startling result if the dataset isn't truely trained. | Detect probable cybercrime, construct substantial checking applications to alleviate the hefty social issue of cyber bullying, opportunity to get better on these techniques to accomplish better results. |
| [2] | Lexicon-based methods, hybrid approached, Emoticon Analysis and Sentiment | cyber-crime can caused heartbreaking | Utilization such given model need to reached out in different online |

| | | | |
|---|---|---|---|
| | Analysis, Vernacular languages | outcomes so need to recognize crime | networking like fb since they didn't adhere specific no. of letterings unlike twitter |
| [3] | Bagof-Words (BoW), Lexical Syntactic Feature (LSF) [2], Bi-Gram tokenizer, NB, LR | Utilization of the uneven data comprising a greater percent of anti-harrasment tweets than harassing tweets. | If indeed the data obtained using collaborative method were similar to those obtained using step-wise method of analysis, it is worthwhile to use the approaches that combine because they are efficient in terms of time. As well as parallel computing seemed to have the better retrieval outcome, parallel computing seemed to have the greater correctness. Whenever the amount of incorrect negatives is minimal, causing the recall improve. |
| [4] | S V M, R F Algorithm, Sentiment Analysis. | A user finds it difficult in determining the assessment of the certain feature for device which intend to buy. Similarly, it seem to be a mix of favorable and unfavorable assessments.. | Users express their opinions online and in large numbers, which helps the provider keep track of user opinion concerning the brand. |
| [5] | N B, Arabic Social Networks, Sentiment Analysis. | The menace of online harassment have received a lot attention in Arab nations, especially with the large variety of online networking sites like fb | It outlines the strategy for detecting bullying on online platforms targeting Arabian responses, with a N B accuracy of 0.959 mostly on excluded YT and Twit data sets. |

| | | | |
|---|---|---|---|
| | | etc platforms available to children. | |
| [6] | A N N, DNN,, Sentiment Analysis, RNN, Maximum Entropy, Naive Bayes , CNN classifier, CNN , and RNN classifier | Researchers must deal with the constant evolution of the terminology used online in user material. | consolidate emotions and text for sentiment analysis. gives the solid correctness i.e 83.6 % with sensitiveness 87.1%, specification 79.3%. |
| [7] | Sentiment Analysis, Twitter, Neural Networks, Logistic regression, Decision Tree Classifier, a skip-gram neural network | Decision tree can sometimes prompt confounded trees that don't sum up without any problem. | Huge data generated by Twitter can be analyzed and utilized for extracting insights on shopper conduct, deciding marketing strategies and improve customer administration and conceivably significantly more. Sentiment Analysis offers a quick and effective approach to complete such an analysis. |
| [8] | social medium; data preprocessing; support vector machine | absence of recognizable criteria that identify whatever posting as a bullying occasion. Even subsequent to distinguishing bullying, making a decision about the seriousness of the example is a test as it tends to be straightforward ridiculing prompting social rejection. | We will get a better result if we use S V M to distinguish abusive trails from pro ones. Create a response rating scale that would categories cyberbullying, properly manage any incident if needed. |
| [9] | NB Classifier, SVM | The Naive Bayes has a positive exactness of 16.04 percent and a 6.77 percent review. As a result, | Utilized two supervised ML models to shape an examination and figure out which among the two gives out the most elevated precision with |

| | | | |
|---|---|---|---|
| | | there isn't much that can be done. | the end goal for us to conclude how to identify cyberbullying movement on the Internet. |
| [10] | Deep learning, NB method | Deep learning methodical models have to have a massive volumes of data and training. | Using picture analysis, iconography, and examining the psychological aspect of a victim and offender, improvements to harassment categorization will be achievable. Twitter is commonly used platform for researchers. Researchers may easily collect data thanks to the Twitter API's openness. |
| [11] | harassing tweets; swearwords; S V M; reliability workers | For Indonesia POS Tagger, there are incorrect labels that need be corrected. | Using SVM computation, effectively detected abuse tweets has had F1-score of 67 percent. |
| [12] | community info amenities, ST, SA | Data for social data management is growing at an exponential rate. Furthermore, effective assessment of large data is directly connect to their implementation. | Its spatial-temporal features underlying users' attitudes were employed to differentiate illness occurrences' regions. |

# 3. Overview of Proposed Work

## 3.1 Architecture or Module for the Proposed System

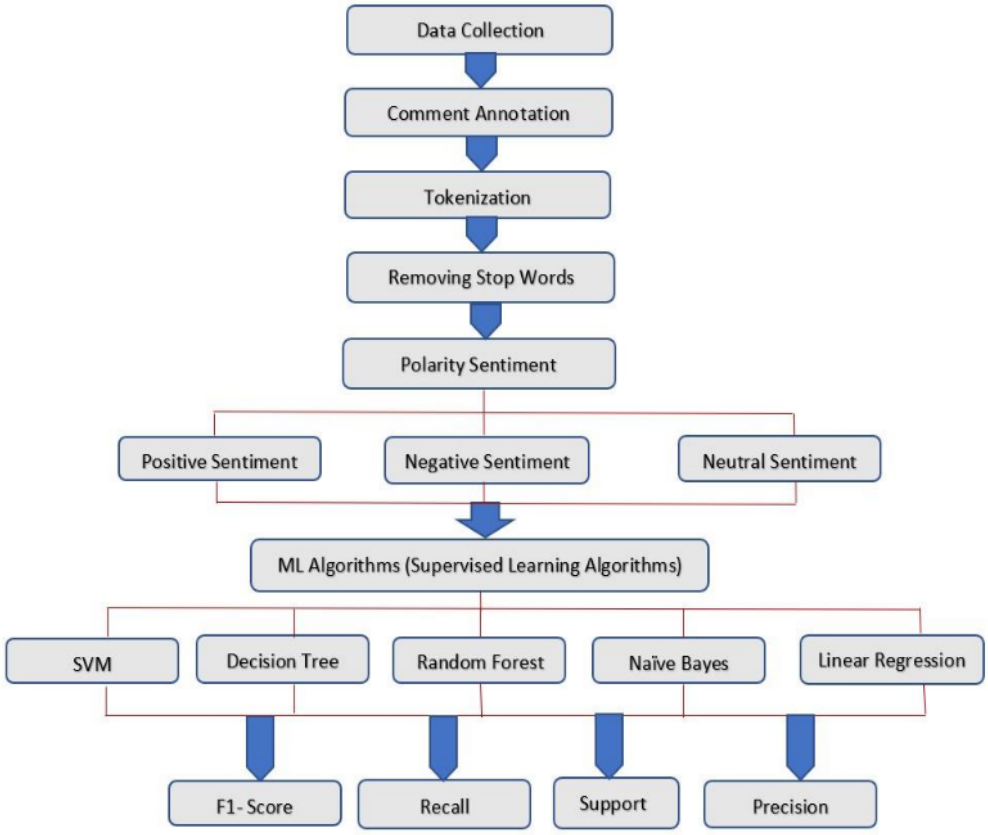



**Figure 1: Architecture Diagram**

In this project, python and web technology are utilized for implementing cyberbullying recognition system. First it will search and find out the dataset and will download it for train the model. Then will preprocess the data and then transferred to Tf-Idf after downloading the dataset first. Now using NB, SVM, DT, LR, RF; It will train dataset and generate model separately. After that will be developing a web application using FLASK framework. After that it will fetch the real time comments from Twitter and then applying generated model to these fetched comments and check the texts are cyberbullying or not. Here in this project, html, css, javascript is used as Fronted, python as backend.

### 3.2 Proposed System Model

Usecase Diagram

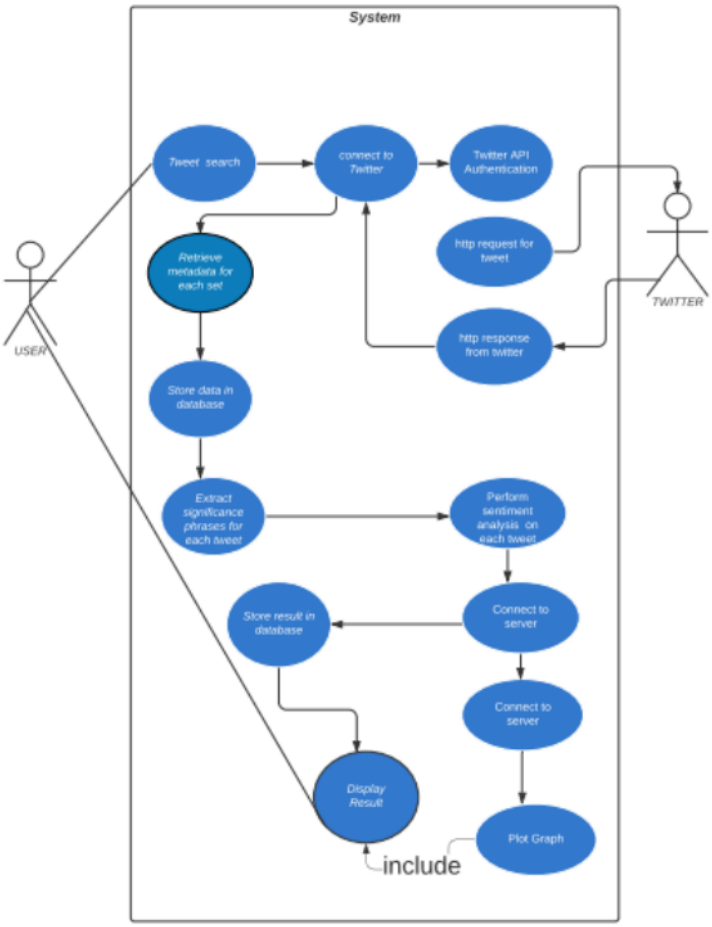



**Figure 2: Usecase Diagram**

ER Diagram

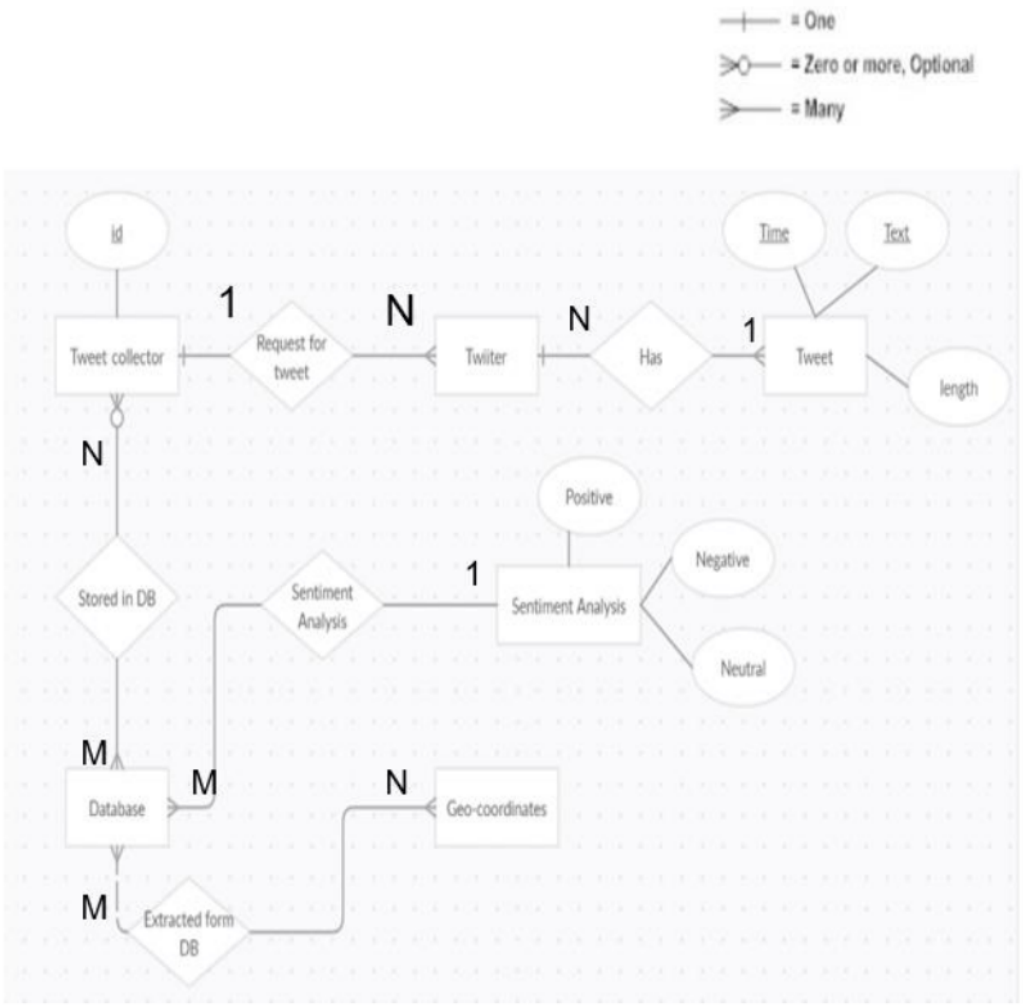



**Figure 3: ER Diagram**

# 4. Proposed System Analysis and Design

## 4.1 Introduction

## 4.2 Requirement Analysis

### 4.2.1 Functional Requirements

#### 4.2.1.1 Product features

Product going to take input in the form of Twitter. In any case, just a single subject was investigated on a given period, defining mind-set in case of tweets people as well as for Facebook comments.

#### 4.2.1.2 User characteristics

It will have characterized by way of systematizing removal such as arrogances, thoughts, visions, feelings in the form of script, dialogue, twitters, file bases over NLP.

#### 4.2.1.3 Assumption & Dependencies

It will be likely to correctly assume sentimentality of one-hundred- forty-character thread for various language script. It is reliant on typical Machine OS structures working appropriately. Social media required on behalf of each study period working appropriately.

#### 4.2.1.4 Domain Requirements

- Windows 10 or any other operating System
- Visual studio
- Twitter api login credentials

#### 4.2.1.5 User Requirements

- Windows 10 or any other operating System

### 4.2.2 Non Functional Requirements

#### 4.2.2.1 Product Requirements

##### 4.2.2.1.1 Efficiency

There are show desires in order to produce numerous situations, public them, clarifying reasoning, in order to promoting inventors with understanding those committed, then settle on appropriate project selections.

#### 4.2.2.1.2 Reliability

System determined to see the entirety for primary needs with no startling conduct. The device final outcome show inaccurate data deprived of informing worker toward expected mistakes in no period.

#### 4.2.2.1.3 Portability

It considered in such a way it can track in every AOS form.

### 4.2.2.2 Operational Requirements

- Political

  The solitary political contribution would be the tweets in regards to politics so comparatively few civil required in organization.

- Ethical

  Here will be not at all moral necessities in the scheme. There was an exceptionally elementary organization, very few moral harms were associated with structure & application.

- Health and Safety

  It will measure on the basis of fitness sector i.e not in physical health straightforwardly but rather in mental health which may assist with improving the physical health.

- Sustainability

  For a drawn out sustainability, it is necessary to check the latest tweets from the twitter. As these tweets are always showing signs of change so need to access the tweets in customary basis.

- Legality

  There is a twitter programming interface credentials which is used in this project. The permission for using twitter programming interface is given by twitter.

- Inspect ability

  The System and the working of visual studio will be inspecting previously whole organization ensuring legitimate working.

### 4.2.3 System Requirements

#### 4.2.3.1 H/W Requirements

- Windows 10
- I5 8th gen

#### 4.2.3.2 S/W Requirements

- Visual Studio
- Twitter API
- Python

# 5. Codes and Outputs

## 5.1 Codes

**#Facebook**

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from collections import defaultdict
from nltk.corpus import wordnet as wn


# In[2]:


import warnings
warnings.filterwarnings("ignore")


# In[3]:


from textblob import *


# In[4]:


trump=pd.read_csv('./trump_comm.csv',encoding='ISO-8859-1',index_col=0)
trump.head()


# In[5]:


import re
trump['Comment'] = [re.sub('(\\W|\\d)', " ",str(text)) for text in
trump['Comment']]


# In[6]:


def get_tweet_sentiment(text):

    analysis = TextBlob(text)
```

```
    # set sentiment
    if analysis.sentiment.polarity > 0:
        return('positive')
    elif analysis.sentiment.polarity == 0:
        return ('neutral')
    else:
        return ('negative')

# In[7]:

trump['Sentiment'] = list(map(get_tweet_sentiment,trump['Comment']))

# In[8]:

trump.Sentiment.value_counts()

# In[9]:

trump.head()

# In[10]:

trump = trump[['Comment','Sentiment']]

# In[11]:

trump.shape

# In[12]:

trump.isnull().sum()

# In[13]:

trump.dropna(inplace=True)

# In[14]:

trump.dtypes
```

```
# In[15]:


trump['Sentiment']=trump['Sentiment'].astype("category")


# In[16]:


from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm


# In[17]:


train_X, test_X, train_y, test_y =
train_test_split(trump['Comment'],trump['Sentiment'],test_size=0.3,rand
om_state=523)


# In[18]:


nltk.download('stopwords')
stop words = stopwords.words('english')

Tfidf_vect =
TfidfVectorizer(min_df=5,max_df=0.8,sublinear_tf=True,use_idf=True)
Tfidf_vect.fit(train_X)
train_X_Tfidf = Tfidf_vect.transform(train_X)
test_X_Tfidf = Tfidf_vect.transform(test_X)


# In[19]:


print(train_X_Tfidf)


# In[20]:


Dense mat = train X Tfidf.todense()
Tfidf Mat = pd.DataFrame(Dense mat,
columns=Tfidf_vect.get_feature_names())
# Tfidf_Mat.shape
Tfidf_Mat.head()


# In[21]:


from sklearn.metrics import classification report,accuracy score,
recall score, precision score
```

```
# In[22]:


acc dict train = {}
acc dict test = {}


# In[23]:


 # Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(kernel='linear')
SVM.fit(train X Tfidf,train y)

# predict the labels on train dataset
pred_train1 = SVM.predict(train_X_Tfidf)

# predict the labels on validation dataset
pred_test1 = SVM.predict(test_X_Tfidf)

# Use accuracy score function to get the accuracy
train_rmse1 = accuracy_score(train_y, pred_train1)*100
test rmse1 = accuracy score(test y, pred test1)*100
print("SVM Accuracy Score on Train set -> ", train rmse1)
print("SVM Accuracy Score on Validation set -> ", test rmse1)

acc_dict_test['SVM'] = accuracy_score(test_y, pred_test1)*100
acc_dict_train['SVM'] = accuracy_score(train_y, pred_train1)*100


# In[24]:


print(classification report(test y,pred test1))


# In[25]:


from sklearn.metrics import precision_score,recall_score,f1_score


# In[26]:


metric = defaultdict(list)
metric['SVM'].append(precision_score(train_y,
pred_train1,average='macro'))
metric['SVM'].append(recall_score(train_y,
pred_train1,average='macro'))
metric['SVM'].append(f1 score(train y, pred train1,average='macro'))


# In[27]:
```

```
# Classifier - Algorithm - Random Forest Algorithm

#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier
Rforest = RandomForestClassifier(n estimators=100)

# fit the training dataset on the classifier
Rforest.fit(train_X_Tfidf,train_y)

# predict the labels on train dataset
pred_train1 = Rforest.predict(train_X_Tfidf)

# predict the labels on validation dataset
pred test1 = Rforest.predict(test X Tfidf)

train rmse2=accuracy score(train y, pred train1)*100
test_rmse2 =accuracy_score(test_y, pred_test1)*100


# Use accuracy score function to get the accuracy
print("RandomForest Accuracy Score on Train set -> ",train_rmse2 )
print("RandomForest Accuracy Score on Validation set -> ",test_rmse2)

acc dict test['RF'] = accuracy score(test y, pred test1)*100
acc dict train['RF'] = accuracy score(train y, pred train1)*100


# In[28]:


print(classification_report(test_y,pred_test1))


# In[29]:


metric['RF'].append(precision_score(train_y,
pred_train1,average='macro'))
metric['RF'].append(recall_score(train_y, pred_train1,average='macro'))
metric['RF'].append(f1_score(train_y, pred_train1,average='macro'))


# In[30]:


# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(train_X_Tfidf,train_y)

# predict the labels on train dataset
pred_train = Naive.predict(train_X_Tfidf)

# predict the labels on validation dataset
pred test = Naive.predict(test X Tfidf)

train_rmse3=accuracy_score(train_y, pred_train)*100
```

```
test_rmse3=accuracy_score(test_y, pred_test)*100


# Use accuracy score function to get the accuracy
print("Naive Bayes Accuracy Score on Train set -> ", train rmse3)
print("Naive Bayes Accuracy Score on Validation set -> ",test rmse3)

 acc_dict_test['NB'] = accuracy_score(test_y, pred_test)*100
acc_dict_train['NB'] = accuracy_score(train_y, pred_train)*100


# In[31]:


print(classification report(test y,pred test1))


# In[32]:


metric['NB'].append(precision_score(train_y,
pred_train,average='macro'))
metric['NB'].append(recall_score(train_y, pred_train,average='macro'))
metric['NB'].append(f1 score(train y, pred train,average='macro'))


# In[33]:


# Logistic Regression Algorithm
# fit the training dataset on the classifier
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random state=0).fit(train X Tfidf,train y)

clf.score(train X Tfidf,train y)

# predict the labels on train dataset
pred_train1 = clf.predict(train_X_Tfidf)

# predict the labels on validation dataset
pred test1 = clf.predict(test X Tfidf)

train rmse4=accuracy score(train y, pred train1)*100
test rmse4=accuracy score(test y, pred test1)*100

# Use accuracy_score function to get the accuracy
print("Logistic Regression Accuracy Score on Train set -> ",train_rmse4
)
print("Logistic Regression Accuracy Score on Validation set ->
",test_rmse4)

acc dict test['LG'] = accuracy score(test y, pred test1)*100
acc dict train['LG'] = accuracy score(train y, pred train1)*100
```

```
# In[34]:


print(classification_report(test_y,pred_test1))


# In[35]:


metric['LG'].append(precision_score(train_y,
pred_train1,average='macro'))
metric['LG'].append(recall_score(train_y, pred_train1,average='macro'))
metric['LG'].append(f1_score(train_y, pred_train1,average='macro'))


# In[36]:


 # Classifier - Algorithm - Decision Tree
# fit the training dataset on the classifier
from sklearn.tree import DecisionTreeClassifier
Dtree = DecisionTreeClassifier(random_state=0)
Dtree.fit(train_X_Tfidf,train_y)

# predict the labels on train dataset
pred train1 = Dtree.predict(train X Tfidf)

# predict the labels on validation dataset
pred_test1 = Dtree.predict(test_X_Tfidf)

train_rmse5=accuracy_score(train_y, pred_train1)*100
test_rmse5=accuracy_score(test_y, pred_test1)*100

# Use accuracy score function to get the accuracy

print("Decision tree Accuracy Score on Train set -> ",train rmse5 )
print("Decision tree Accuracy Score on Validation set -> ",test rmse5)

acc_dict_test['DT'] = accuracy_score(test_y, pred_test1)*100
acc_dict_train['DT'] = accuracy_score(train_y, pred_train1)*100


# In[37]:


print(classification report(test y,pred test1))


# In[38]:


metric['DT'].append(precision_score(train_y,
pred train1,average='macro'))
metric['DT'].append(recall score(train y, pred train1,average='macro'))
metric['DT'].append(f1 score(train y, pred train1,average='macro'))
```

```
# In[39]:


tmp = pd.DataFrame(dict(metric),index=['precision','recall','f1'])


# In[40]:


import matplotlib.pyplot as plt


# In[41]:


## Visualization and comparison of metrics for all the models or
algorithms
ax=tmp.plot(kind='bar',figsize=(16,6),width=0.8)
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height*100:.3}%', (x + width/2, y + height*1.02),
ha='center')
plt.show()


# In[42]:


## RMSE evaluation
from sklearn.metrics import mean_squared_error
var1 =(mean_squared_error([train_rmse1],[test_rmse1]))
var2 =(mean squared error([train rmse2],[test rmse2]))
var3 =(mean squared error([train rmse3],[test rmse3]))
var4 = (mean squared error([train rmse4],[test rmse4]))
var5 = (mean squared error([train rmse4],[test rmse4]))
var1=np.sqrt(var1)
var2=np.sqrt(var2)
var3=np.sqrt(var3)
var4=np.sqrt(var4)
var5=np.sqrt(var5)
print("RMSE of SVM",var1)
print("RMSE of Random Forest",var2)
print("RMSE of Naive Bayes",var3)
print("RMSE of Logistic Regression",var4)
print("RMSE of Decision Tree",var4)


# In[43]:


## Bar plot of accuracy of different models trained above.
x = np.arange(5)  # the label locations
width = 0.35
fig,ax = plt.subplots(figsize=(14,8))
```

```
#ax = plt.subplot(121)

br = ax.bar(x-width/2,list(acc_dict_train.values()),width,
label='train')
br2 = ax.bar(x+width/2,list(acc dict test.values()),width,
label='test')
ax.set xticks((0,1,2,3,4))
ax.set xticklabels(['SVM','RF','NB','LG','DT'])
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.4}%', (x + width/2, y + height*1.02),
ha='center')
ax.legend(loc='upper right')
#fig.tight layout()
plt.show()


# In[ ]:
```

**TWITTER**

```
Filename: get_tweet_data_nb
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import nltk
nltk.download('sentiwordnet')

from nltk.corpus import wordnet as wn
from nltk.corpus import sentiwordnet as swn
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# Basic Packages
import random
import pandas as pd

# Text Preprocessing Packages
import re
import nltk
# from nltk.tokenize import word tokenize
import tweepy
from tweepy.auth import OAuthHandler
from textblob import *

import warnings
warnings.filterwarnings("ignore")
```

```
# In[2]:


def get data(name):
    #name='gandhi'
    def initialize():

        # keys and tokens from the Twitter Dev Console
        consumer_key = 'snseusIoIioTvEpDaBcPjUryw'
        consumer_secret =
'cDhGVySW9xRUQSbc2o8yKMHfAxBrnIBvE1wSaWozlPIBXspFTm'
        access_token = '2856915038-
5xVKLBpmMhX314uHBZfmdmIRtXGtlQ0K8yUrexR'
        access token secret =
'uiHGPQZuxr9z0bsnaPMPgdMemk8oXOwUYjSLJfsT2VmCM'

        # attempt authentication
        try:
            # create OAuthHandler object
            auth = OAuthHandler(consumer_key, consumer_secret)
            # set access token and secret
            auth.set_access_token(access_token, access_token_secret)
            # create tweepy API object to fetch tweets
            api = tweepy.API(auth,wait on rate limit=True)
            print('Authentication Success')
            return(api)

        except Exception as e:
            print("Error: Authentication Failed")
            print(e)



    def get_tweets(api, query, count = 100):
            # empty list to store parsed tweets
            tweets = []
            sinceId = None
            max_id = -1
            tweetCount = 0
            tweetsPerQry = 100

            while tweetCount < count:
                try:
                    if (max_id <= 0):
                        if (not sinceId):
                            new_tweets = api.search(q=query,
count=tweetsPerQry,lang='en')
                        else:
                            new_tweets = api.search(q=query,
count=tweetsPerQry,

since_id=sinceId,lang='en')
                    else:
                        if (not sinceId):
                            new_tweets = api.search(q=query,
count=tweetsPerQry,
```

```
                                                    max_id=str(max_id -
1),lang='en')
                        else:
                            new_tweets = api.search(q=query,
count=tweetsPerQry,
                                                    max_id=str(max_id -
1),
since_id=sinceId,lang='en')
                        if not new_tweets:
                            print("No more tweets found")
                            break

                        for tweet in new_tweets:
                            parsed_tweet = {}
                            parsed_tweet['tweets'] = tweet.text
                            parsed_tweet['date'] = tweet.created_at

                            # saving sentiment of tweet

parsed_tweet['cleaned_tweets'],parsed_tweet['sentiment_score'],parsed_t
weet['sentiment'] = get_sentiment(tweet.text)
                            #parsed_tweet['sentiments'] =
[tag_sentiment(tweet.text)]
                            # appending parsed tweet to tweets list
                            if tweet.retweet_count > 0:
                                # if tweet has retweets, ensure that it is
appended only once
                                if parsed_tweet not in tweets:
                                    tweets.append(parsed_tweet)
                            else:
                                tweets.append(parsed_tweet)

                        tweetCount += len(new_tweets)
                        #print("Downloaded {0} tweets".format(tweetCount))
                        max_id = new_tweets[-1].id
                       # print(max_id)
                       # print(new_tweets[-1])
                        return tweets
                    except tweepy.TweepError as e:
                        print("Tweepy error : " + str(e))

    api_initialization = initialize()

    def clean_tweet(tweets):
        '''

        Utility function to clean tweet text by removing links, special
characters
        using simple regex statements.
        '''
        #print(tweets)
        return(' '.join(re.sub("([,\.():;!$%^&*\d])|([^0-9A-Za-z \t])",
" ", tweets).split()))

    def penn_to_wn(tag):
        if tag.startswith('J'):
            return wn.ADJ
        elif tag.startswith('N'):
            return wn.NOUN
        elif tag.startswith('R'):
            return wn.ADV
```

```
        elif tag.startswith('V'):
            return wn.VERB
        return None

    from nltk.stem import WordNetLemmatizer
    lemmatizer = WordNetLemmatizer()

    def get sentiment(text):
        """ returns list of pos neg and objective score. But returns
empty list if not present in senti wordnet. """
        cleaned_tweets = preprocess_tweet(text)
        tagged = nltk.pos_tag(word_tokenize(cleaned_tweets))
        sentiment_score = 0.0
        tokens_count = 0
        sentiment = []
        for word, tag in tagged:
            wn tag = penn to wn(tag)
            if wn_tag not in (wn.NOUN, wn.ADJ, wn.ADV):
                continue

            lemma = lemmatizer.lemmatize(word, pos=wn_tag)
            if not lemma:
                continue

            synsets = wn.synsets(word, pos=wn tag)
            if not synsets:
                continue

        # Take the first sense, the most common
            synset = synsets[0]
    #         print(synset)
    #         print(synset.name())
            swn_synset = swn.senti_synset(synset.name())
    #         print(swn synset)
            sentiment score += swn synset.pos score() -
swn synset.neg score()

            tokens_count += 1
        if not tokens_count:
            sentiment.append('neutral')

    # sum greater than 0 => positive sentiment
        if sentiment score >= 0:
            sentiment.append('positive')
        else:
            sentiment.append('negative')
        # negative sentiment


        return cleaned_tweets,sentiment_score, sentiment.pop()

    contractions = {
"ain't": "is not",
"aren't": "are not",
"can't": "cannot",
"can't've": "cannot have",
"'cause": "because",
```

```
"could've": "could have",
"couldn't": "could not",
"couldn't've": "could not have",
"didn't": "did not",
"doesn't": "does not",
"don't": "do not",
"hadn't": "had not",
"hadn't've": "had not have",
"hasn't": "has not",
"haven't": "have not",
"he'd": "he would",
"he'd've": "he would have",
"he'll": "he will",
"he'll've": "he he will have",
"he's": "he is",
"how'd": "how did",
"how'd'y": "how do you",
"how'll": "how will",
"how's": "how is",
"I'd": "I would",
"I'd've": "I would have",
"I'll": "I will",
"I'll've": "I will have",
"I'm": "I am",
"I've": "I have",
"i'd": "i would",
"i'd've": "i would have",
"i'll": "i will",
"i'll've": "i will have",
"i'm": "i am",
"i've": "i have",
"isn't": "is not",
"it'd": "it would",
"it'd've": "it would have",
"it'll": "it will",
"it'll've": "it will have",
"it's": "it is",
"let's": "let us",
"ma'am": "madam",
"mayn't": "may not",
"might've": "might have",
"mightn't": "might not",
"mightn't've": "might not have",
"must've": "must have",
"mustn't": "must not",
"mustn't've": "must not have",
"needn't": "need not",
"needn't've": "need not have",
"o'clock": "of the clock",
"oughtn't": "ought not",
"oughtn't've": "ought not have",
"shan't": "shall not",
"sha'n't": "shall not",
"shan't've": "shall not have",
"she'd": "she would",
"she'd've": "she would have",
"she'll": "she will",
```

```
"she'll've": "she will have",
"she's": "she is",
"should've": "should have",
"shouldn't": "should not",
"shouldn't've": "should not have",
"so've": "so have",
"so's": "so as",
"that'd": "that would",
"that'd've": "that would have",
"that's": "that is",
"there'd": "there would",
"there'd've": "there would have",
"there's": "there is",
"they'd": "they would",
"they'd've": "they would have",
"they'll": "they will",
"they'll've": "they will have",
"they're": "they are",
"they've": "they have",
"to've": "to have",
"wasn't": "was not",
"we'd": "we would",
"we'd've": "we would have",
"we'll": "we will",
"we'll've": "we will have",
"we're": "we are",
"we've": "we have",
"weren't": "were not",
"what'll": "what will",
"what'll've": "what will have",
"what're": "what are",
"what's": "what is",
"what've": "what have",
"when's": "when is",
"when've": "when have",
"where'd": "where did",
"where's": "where is",
"where've": "where have",
"who'll": "who will",
"who'll've": "who will have",
"who's": "who is",
"who've": "who have",
"why's": "why is",
"why've": "why have",
"will've": "will have",
"won't": "will not",
"won't've": "will not have",
"would've": "would have",
"wouldn't": "would not",
"wouldn't've": "would not have",
"y'all": "you all",
"y'all'd": "you all would",
"y'all'd've": "you all would have",
"y'all're": "you all are",
"y'all've": "you all have",
"you'd": "you would",
"you'd've": "you would have",
```

```
"you'll": "you will",
"you'll've": "you will have",
"you're": "you are",
"you've": "you have"
}

    def expand_contractions(text):
        for word in text.split():
            if word.lower() in contractions:
                text = text.replace(word, contractions[word.lower()])
        return text

    def preprocess_word(word):
        # Remove punctuation
        word = word.strip('"?!,.():;')
        # Convert more than 2 letter repetitions to 2 letter
        # funnnnny --> funny
        word = re.sub(r'(.)\1+', r'\1\1', word)
        # Remove - & '
        word = re.sub(r'(-)', '', word)
        return word

    def is_valid_word(word):
        # Check if word begins with an alphabet
        return (re.search(r'^[a-zA-Z][a-z0-9A-Z\. ]*$', word) is not
None)

    def handle_emojis(tweet):
        # Smile -- :), : ), :-), (:, ( :, (-:, :')
        tweet = re.sub(r'(:\s?\)|:-\)|\(\s?:|\(-:|:\'\))', ' EMO_POS ',
tweet)
        # Laugh -- :D, : D, :-D, xD, x-D, XD, X-D
        tweet = re.sub(r'(:\s?D|:-D|x-?D|X-?D)', ' EMO_POS ', tweet)
        # Love -- <3, :*
        tweet = re.sub(r'(<3|:\*)', ' EMO POS ', tweet)
        # Wink -- ;-), ;), ;-D, ;D, (;,  (-;
        tweet = re.sub(r'(;-?\)|;-?D|\(-?;)', ' EMO POS ', tweet)
        # Sad -- :-(, : (, :(, ):, )-:
        tweet = re.sub(r'(:\s?\(|:-\(|\)\s?:|\)-:)', ' EMO_NEG ',
tweet)
        # Cry -- :,(, :'(, :"(
        tweet = re.sub(r'(:,\(|:\'\(|:"\()', ' EMO_NEG ', tweet)
        return tweet
    from nltk import WordNetLemmatizer

    def preprocess_tweet(tweet):
        processed_tweet = []
        # Convert to lower case
        tweet = tweet.lower()
        tweet = expand_contractions(re.sub('’', "'", tweet))
        # Replaces URLs with the word URL
        tweet = re.sub(r'((www\.[\S]+)|(https?://[\S]+))', ' URL ',
tweet)
```

```
            # Replace @handle with the word USER_MENTION
            tweet = re.sub(r'@[\S]+', r' ', tweet)
            # Replaces #hashtag with hashtag
            tweet = re.sub(r'#(\S+)', r' \1 ', tweet)
            # Remove RT (retweet)
            tweet = re.sub(r'\brt\b', '', tweet)
            # Replace 2+ dots with space
            tweet = re.sub(r'\.{2,}', ' ', tweet)
            # Strip space, " and ' from tweet
            #tweet = tweet.strip(' "\'')
            # Replace emojis with either EMO_POS or EMO_NEG
            tweet = handle_emojis(tweet)
            # Replace multiple spaces with a single space
            tweet = re.sub(r'\s+', ' ', tweet)
            tweet = re.sub(r'(\[|\])',' ', tweet)

            words = tweet.split()

            for word in words:
                word = preprocess_word(word)
                if is_valid_word(word):
                    #if use_stemmer:
                    word = str(WordNetLemmatizer().lemmatize(word))
                    processed_tweet.append(word)

            return ' '.join(processed_tweet)
    global retreived_tweets
    retreived_tweets = get_tweets(api=api_initialization,query=name,
count = 100)


    return retreived_tweets


# In[5]:


if __name__ == "__main__":
    get_data('OnePlus 8')


# In[6]:


pd.DataFrame(retreived_tweets)


# In[ ]:
```

```
Filename: tweets_training
#!/usr/bin/env python
# coding: utf-8

# In[5]:


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from nltk.corpus import stopwords
from sklearn.metrics import classification report,accuracy score,
recall score, precision score,f1 score
import collections
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error
#from alok assignment get tweets flask import get data
#from get tweet data import get data
import warnings
warnings.filterwarnings("ignore")


# In[6]:


def train_tweets(df):
    df = pd.DataFrame(retreived tweets)
    df.head()
    display(df.head())
    df = df[['tweets','sentiment']]

    df['sentiment']=df['sentiment'].astype("category")

    train_X, test_X, train_y, test_y =
train_test_split(df['tweets'],df['sentiment'],test_size=0.3,random_stat
e=523)

    #nltk.download('stopwords')

    stop_words = stopwords.words('english')

    Tfidf_vect =
TfidfVectorizer(min_df=5,max_df=0.8,sublinear_tf=True,use_idf=True)
    Tfidf_vect.fit(train_X)
    train_X_Tfidf = Tfidf_vect.transform(train_X)
    test_X_Tfidf = Tfidf_vect.transform(test_X)

    acc_dict_train = {}
    acc_dict_test = {}
```

```
    # Classifier - Algorithm - SVM
    # fit the training dataset on the classifier
    SVM = svm.SVC(kernel='linear')
    SVM.fit(train_X_Tfidf,train_y)

    # predict the labels on train dataset
    pred train1 = SVM.predict(train X Tfidf)

    # predict the labels on validation dataset
    pred_test1 = SVM.predict(test_X_Tfidf)

    # Use accuracy_score function to get the accuracy
    train_rmse1 = accuracy_score(train_y, pred_train1)*100
    test_rmse1 = accuracy_score(test_y, pred_test1)*100
    print("SVM Accuracy Score on Train set -> ", train rmse1)
    print("SVM Accuracy Score on Validation set -> ", test rmse1)

    acc_dict_test['SVM'] = accuracy_score(test_y, pred_test1)*100
    acc_dict_train['SVM'] = accuracy_score(train_y, pred_train1)*100


    print(classification_report(test_y,pred_test1))



    metric = collections.defaultdict(list)
    metric['SVM'].append(precision score(train y,
pred_train1,average='macro'))
    metric['SVM'].append(recall_score(train_y,
pred_train1,average='macro'))
    metric['SVM'].append(f1_score(train_y,
pred_train1,average='macro'))

    # Classifier - Algorithm - Random Forest Algorithm

    #Import Random Forest Model

    Rforest = RandomForestClassifier(n_estimators=100)

    # fit the training dataset on the classifier
    Rforest.fit(train_X_Tfidf,train_y)

    # predict the labels on train dataset
    pred train1 = Rforest.predict(train X Tfidf)

    # predict the labels on validation dataset
    pred_test1 = Rforest.predict(test_X_Tfidf)

    train_rmse2=accuracy_score(train_y, pred_train1)*100
    test_rmse2 =accuracy_score(test_y, pred_test1)*100


    # Use accuracy score function to get the accuracy
    print("RandomForest Accuracy Score on Train set -> ",train rmse2 )
    print("RandomForest Accuracy Score on Validation set ->
",test rmse2)
```

```
    acc_dict_test['RF'] = accuracy_score(test_y, pred_test1)*100
    acc_dict_train['RF'] = accuracy_score(train_y, pred_train1)*100

    print(classification_report(test_y,pred_test1))

    metric['RF'].append(precision score(train y,
pred train1,average='macro'))
    metric['RF'].append(recall score(train y,
pred_train1,average='macro'))
    metric['RF'].append(f1_score(train_y, pred_train1,average='macro'))

    # fit the training dataset on the NB classifier
    Naive = naive_bayes.MultinomialNB()
    Naive.fit(train_X_Tfidf,train_y)

    # predict the labels on train dataset
    pred train = Naive.predict(train X Tfidf)

    # predict the labels on validation dataset
    pred_test = Naive.predict(test_X_Tfidf)

    train_rmse3=accuracy_score(train_y, pred_train)*100
    test_rmse3=accuracy_score(test_y, pred_test)*100



    # Use accuracy score function to get the accuracy
    print("Naive Bayes Accuracy Score on Train set -> ", train_rmse3)
    print("Naive Bayes Accuracy Score on Validation set ->
",test_rmse3)

    acc_dict_test['NB'] = accuracy_score(test_y, pred_test)*100
    acc_dict_train['NB'] = accuracy_score(train_y, pred_train)*100

    print(classification report(test y,pred test1))

    metric['NB'].append(precision score(train y,
pred_train,average='macro'))
    metric['NB'].append(recall_score(train_y,
pred_train,average='macro'))
    metric['NB'].append(f1_score(train_y, pred_train,average='macro'))

    # Logistic Regression Algorithm
    # fit the training dataset on the classifier


    clf = LogisticRegression(random_state=0).fit(train_X_Tfidf,train_y)

    clf.score(train_X_Tfidf,train_y)

    # predict the labels on train dataset
    pred_train1 = clf.predict(train_X_Tfidf)

    # predict the labels on validation dataset
    pred test1 = clf.predict(test X Tfidf)

    train_rmse4=accuracy_score(train_y, pred_train1)*100
```

```
    test_rmse4=accuracy_score(test_y, pred_test1)*100

    # Use accuracy score function to get the accuracy
    print("Logistic Regression Accuracy Score on Train set ->
",train rmse4 )
    print("Logistic Regression Accuracy Score on Validation set ->
",test rmse4)

    acc_dict_test['LG'] = accuracy_score(test_y, pred_test1)*100
    acc_dict_train['LG'] = accuracy_score(train_y, pred_train1)*100


    print(classification_report(test_y,pred_test1))

    metric['LG'].append(precision score(train y,
pred train1,average='macro'))
    metric['LG'].append(recall score(train y,
pred_train1,average='macro'))
    metric['LG'].append(f1_score(train_y, pred_train1,average='macro'))

    # Classifier - Algorithm - Decision Tree
    # fit the training dataset on the classifier

    Dtree = DecisionTreeClassifier(random_state=0)
    Dtree.fit(train X Tfidf,train y)

    # predict the labels on train dataset
    pred_train1 = Dtree.predict(train_X_Tfidf)

    # predict the labels on validation dataset
    pred_test1 = Dtree.predict(test_X_Tfidf)

    train_rmse5=accuracy_score(train_y, pred_train1)*100
    test rmse5=accuracy score(test y, pred test1)*100

    # Use accuracy score function to get the accuracy
    print("Decision tree Accuracy Score on Train set -> ",train rmse5 )
    print("Decision tree Accuracy Score on Validation set ->
",test_rmse5)

    acc_dict_test['DT'] = accuracy_score(test_y, pred_test1)*100
    acc_dict_train['DT'] = accuracy_score(train_y, pred_train1)*100

    print(classification report(test y,pred test1))

    metric['DT'].append(precision score(train y,
pred_train1,average='macro'))
    metric['DT'].append(recall_score(train_y,
pred_train1,average='macro'))
    metric['DT'].append(f1_score(train_y, pred_train1,average='macro'))

    tmp = pd.DataFrame(dict(metric),index=['precision','recall','f1'])
    display(tmp)

tmp.plot.pie(subplots=True,figsize=(16,12),autopct='%1.1f%%',labels=Non
e)
    plt.legend(labels=tmp.index,loc='best')
```

```
    ax=tmp.plot(kind='bar',figsize=(16,6),width=0.8)
    for p in ax.patches:
        width = p.get_width()
        height = p.get_height()
        x, y = p.get_xy()
        ax.annotate(f'{height*100:.3}%', (x + width/2, y +
height*1.02), ha='center')
    plt.show()


    var1 =(mean_squared_error([train_rmse1],[test_rmse1]))
    var2 =(mean_squared_error([train_rmse2],[test_rmse2]))
    var3 =(mean_squared_error([train_rmse3],[test_rmse3]))
    var4 = (mean_squared_error([train_rmse4],[test_rmse4]))
    var5 = (mean_squared_error([train_rmse4],[test_rmse4]))
    var1=np.sqrt(var1)
    var2=np.sqrt(var2)
    var3=np.sqrt(var3)
    var4=np.sqrt(var4)
    var5=np.sqrt(var5)
    print("RMSE of SVM",var1)
    print("RMSE of Random Forest",var2)
    print("RMSE of Naive Bayes",var3)
    print("RMSE of Logistic Regression",var4)
    print("RMSE of Decision Tree",var4)

    x = np.arange(5)  # the label locations
    width = 0.35

    fig,ax = plt.subplots(figsize=(14,8))

    #ax = plt.subplot(121)

    br = ax.bar(x-width/2,list(acc_dict_train.values()),width,
label='train')
    br2 = ax.bar(x+width/2,list(acc_dict_test.values()),width,
label='test')
    ax.set_xticks((0,1,2,3,4))
    ax.set_xticklabels(['SVM','RF','NB','LG','DT'])
    for p in ax.patches:
        width = p.get_width()
        height = p.get_height()
        x, y = p.get_xy()
        ax.annotate(f'{height:.4}%', (x + width/2, y + height*1.02),
ha='center')
    ax.legend(loc='upper right')
    #fig.tight_layout()
    plt.show()

    return


# In[57]:


if __name__ == "__main__":
```

```
    retreived_tweets = get_data('OnePlus 8')
    train_tweets(retreived_tweets)


# In[ ]:




# In[ ]:



#Filename:tmp.py

from flask import Flask, render_template, request, redirect, url_for
import jinja2
from joblib import load
from get_tweet_data import get_data
import json
import tweepy
from tweepy.auth import OAuthHandler
app = Flask(  name   ,template folder="template")

@app.route('/')
def base():
    # ENV =
jinja2.Environment(loader=jinja2.FileSystemLoader('twitter sentiment ap
i demo/template'))
    # template = ENV.get template('base.html')

    return render template("tweets.html")

@app.route('/',methods=['POST','GET'])
def get_tweets():
    success=False
    if request.method == 'POST':
        topic = request.form['search']
        print(type(topic))
        try:
            global tweets
            tweets = get data(topic)
            print(tweets)
            if len(tweets)>0:
                success = True
                return render_template('tweets.html',status =
'successfull',show=False)
            else:
                return render_template('tweets.html',status =
'unsuccessfull')
        except:
            return render template('tweets.html',status = 'something
went wrong')
```

```
@app.route('/show',methods = ["POST","GET"])
def show():
    print(tweets[0])
    if request.method=='GET':
        return render template('tweets.html',status =
'successfull',show=True,tweets=tweets)

@app.route('/pipeline')
def pipeline():
    import datetime
    print(tweets)
    def myconverter(o):
        if isinstance(o, datetime.datetime):
            return o.__str__()

    return render template('pipeline.html',tweets=json.dumps(tweets,
default = myconverter))

if __name__ == "__main__":
    app.run(debug=True)
```

```
#Filename: get_tweet_data_copy

import nltk
#nltk.download('sentiwordnet')

from nltk.corpus import wordnet as wn
from nltk.corpus import sentiwordnet as swn
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# Basic Packages
import random
import pandas as pd

# Text Preprocessing Packages
import re
import nltk
# from nltk.tokenize import word tokenize
import tweepy
from tweepy.auth import OAuthHandler
from textblob import *

import warnings
warnings.filterwarnings("ignore")

def get_data(name):
    def initialize():

        # keys and tokens from the Twitter Dev Console
        consumer key = 'snseusIoIioTvEpDaBcPjUryw'
        consumer secret =
'cDhGVySW9xRUQSbc2o8yKMHfAxBrnIBvE1wSaWoz1PIBXspFTm'
```

```
        access_token = '2856915038-
5xVKLBpmMhX3l4uHBZfmdmIRtXGt1Q0K8yUrexR'
        access_token_secret =
'uiHGPQZuxr9z0bsnaPMPgdMemk8oXOwUYjSLJfsT2VmCM'

        # attempt authentication
        try:
            # create OAuthHandler object
            auth = OAuthHandler(consumer_key, consumer_secret)
            # set access token and secret
            auth.set_access_token(access_token, access_token_secret)
            # create tweepy API object to fetch tweets
            api = tweepy.API(auth)
            print('Authentication Success')
            return(api)

        except Exception as e:
            print("Error: Authentication Failed")
            print(e)


    def get_tweets(api, query, count = 100):
            # empty list to store parsed tweets
            tweets = []
            sinceId = None
            max_id = -1
            tweetCount = 0
            tweetsPerQry = 100

            while tweetCount < count:
                try:
                    if (max_id <= 0):
                        if (not sinceId):
                            new_tweets = api.search(q=query,
count=tweetsPerQry,lang='en')
                        else:
                            new_tweets = api.search(q=query,
count=tweetsPerQry,

since_id=sinceId,lang='en')
                    else:
                        if (not sinceId):
                            new_tweets = api.search(q=query,
count=tweetsPerQry,
                                                    max_id=str(max_id -
1),lang='en')
                        else:
                            new_tweets = api.search(q=query,
count=tweetsPerQry,
                                                    max_id=str(max_id -
1),

since_id=sinceId,lang='en')
                    if not new_tweets:
                        print("No more tweets found")
                        break
```

```
                    for tweet in new_tweets:
                        parsed_tweet = {}
                        parsed_tweet['tweets'] = tweet.text
                        parsed_tweet['date'] = tweet.created_at

                        # saving sentiment of tweet

#parsed tweet['sentiment score'],parsed tweet['sentiment'] =
get_sentiment(tweet.text)
                        #parsed_tweet['sentiments'] =
[tag_sentiment(tweet.text)]
                        # appending parsed tweet to tweets list
                        if tweet.retweet_count > 0:
                            # if tweet has retweets, ensure that it is
appended only once
                            if parsed tweet not in tweets:
                                tweets.append(parsed tweet)
                        else:
                            tweets.append(parsed_tweet)

                    tweetCount += len(new_tweets)
                    #print("Downloaded {0} tweets".format(tweetCount))
                    max_id = new_tweets[-1].id
                   # print(max id)
                   # print(new tweets[-1])
                    return tweets
                except tweepy.TweepError as e:
                    print("Tweepy error : " + str(e))

    api_initialization = initialize()
    retreived_tweets = get_tweets(api=api_initialization,query=name,
count = 100)

    return retreived tweets
```

### 5.2 Outputs

```
SVM Accuracy Score on Train set ->  93.43783209351754
SVM Accuracy Score on Validation set ->  85.65230864580106
```

```
1 print(classification_report(test_y,pred_test1))
```

```
              precision    recall  f1-score   support

    negative       0.84      0.63      0.72       678
     neutral       0.86      0.97      0.91      1421
    positive       0.85      0.85      0.85      1128

    accuracy                           0.86      3227
   macro avg       0.85      0.82      0.83      3227
weighted avg       0.86      0.86      0.85      3227
```

```
RandomForest Accuracy Score on Train set ->  99.80074388947928
RandomForest Accuracy Score on Validation set ->  75.92190889370933
```

```
1 print(classification_report(test_y,pred_test1))
```

```
              precision    recall  f1-score   support

    negative       0.93      0.27      0.42       678
     neutral       0.78      0.92      0.85      1421
    positive       0.71      0.85      0.77      1128

    accuracy                           0.76      3227
   macro avg       0.81      0.68      0.68      3227
weighted avg       0.79      0.76      0.73      3227
```

```
Naive Bayes Accuracy Score on Train set ->  83.0499468650372
Naive Bayes Accuracy Score on Validation set ->  75.27114967462039
```

```
1 print(classification_report(test_y,pred_test1))
```

```
              precision    recall  f1-score   support

    negative       0.93      0.27      0.42       678
     neutral       0.78      0.92      0.85      1421
    positive       0.71      0.85      0.77      1128

    accuracy                           0.76      3227
   macro avg       0.81      0.68      0.68      3227
weighted avg       0.79      0.76      0.73      3227
```

```
Logistic Regression Accuracy Score on Train set ->  90.52869287991498
Logistic Regression Accuracy Score on Validation set ->  82.18159281066005
```

```
1 print(classification_report(test_y,pred_test1))
```

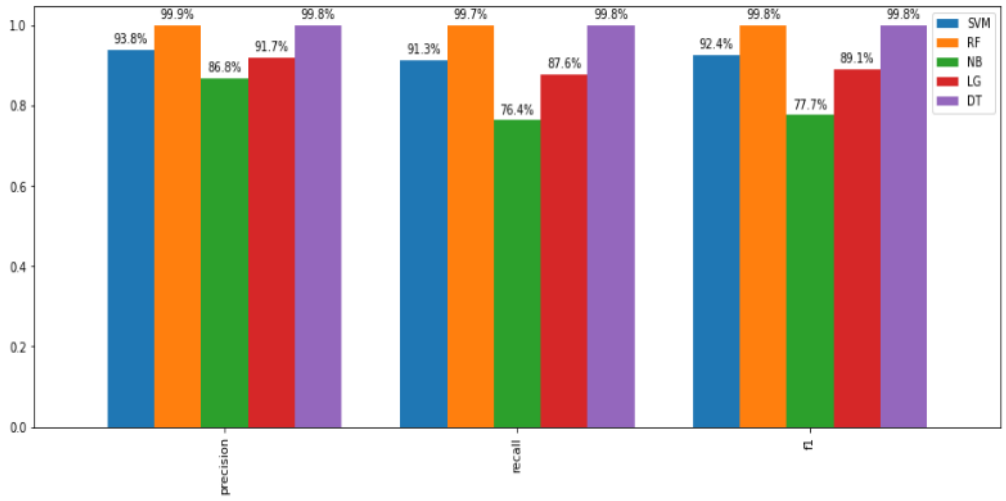
```
              precision    recall  f1-score   support

    negative       0.87      0.54      0.67       678
     neutral       0.82      0.94      0.87      1421
    positive       0.81      0.84      0.83      1128

    accuracy                           0.82      3227
   macro avg       0.83      0.77      0.79      3227
weighted avg       0.83      0.82      0.81      3227
```

```
Decision tree Accuracy Score on Train set ->  99.80074388947928
Decision tree Accuracy Score on Validation set ->  69.72420204524326
```

```
1 print(classification_report(test_y,pred_test1))
```

```
              precision    recall  f1-score   support

    negative       0.49      0.44      0.46       678
     neutral       0.77      0.84      0.80      1421
    positive       0.71      0.67      0.69      1128

    accuracy                           0.70      3227
   macro avg       0.66      0.65      0.65      3227
weighted avg       0.69      0.70      0.69      3227
```

**Figure 4: Accuracy score of train and validation of Facebook of various ML**

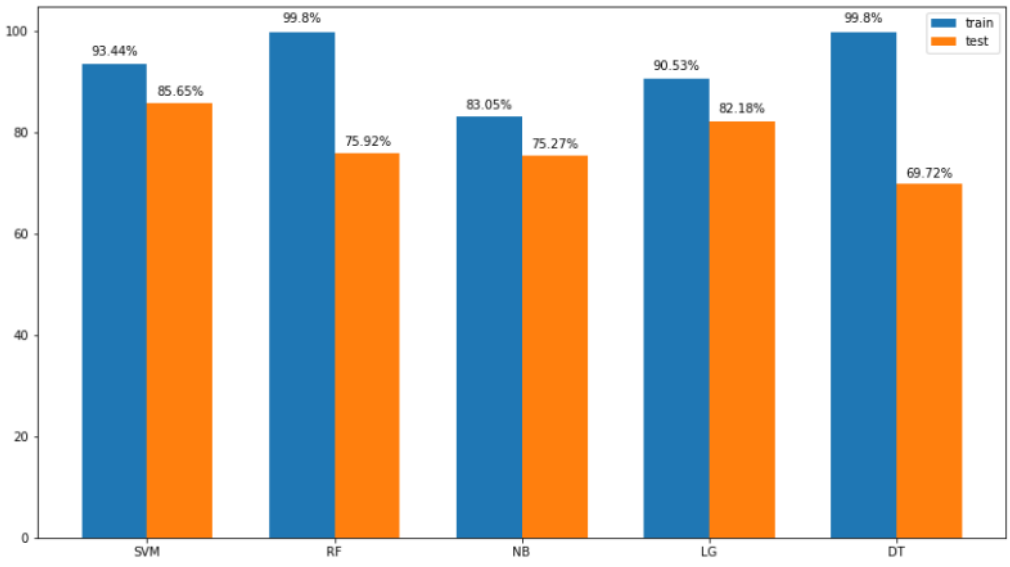



**X= Metrics**
**Y= 1 Unit=10%**

**Figure 5: Bar graph of various ML model using Facebook**

```
RMSE of SVM 7.785523447716486
RMSE of Random Forest 23.87883499576995
RMSE of Naive Bayes 7.77879719041681
RMSE of Logistic Regression 8.347100069254935
RMSE of Decision Tree 8.347100069254935
```

**Figure 6: RMSE value of various Ml Models using Facebook**




**X= Metrics**
**Y= 1 Unit = 1%**

**Figure 7: Bar graph train and test data of Facebook**

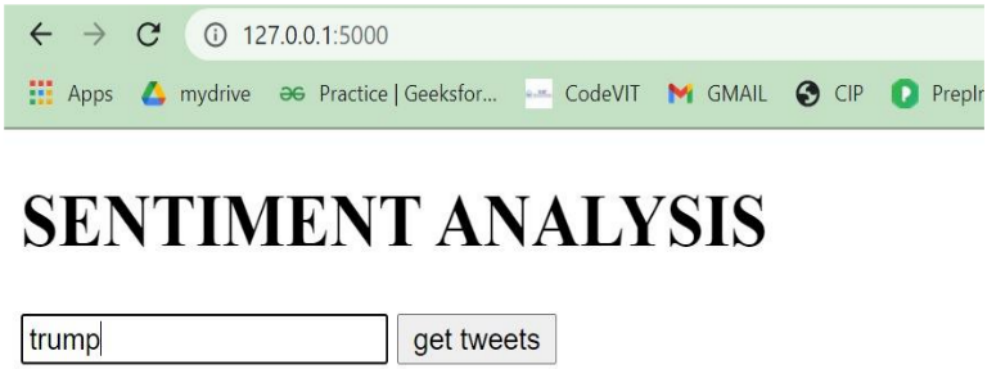



**Figure 8: Sentiment Analysis using twitter Page**

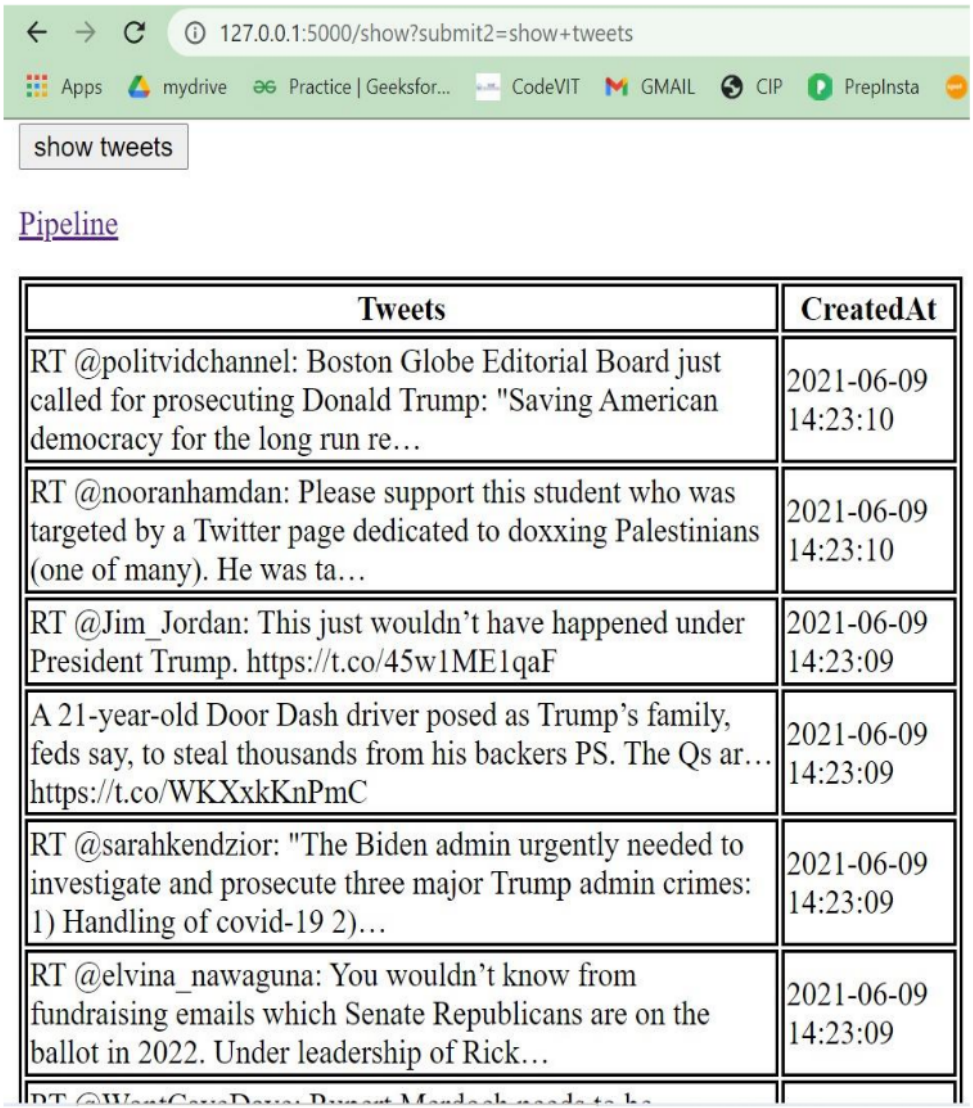
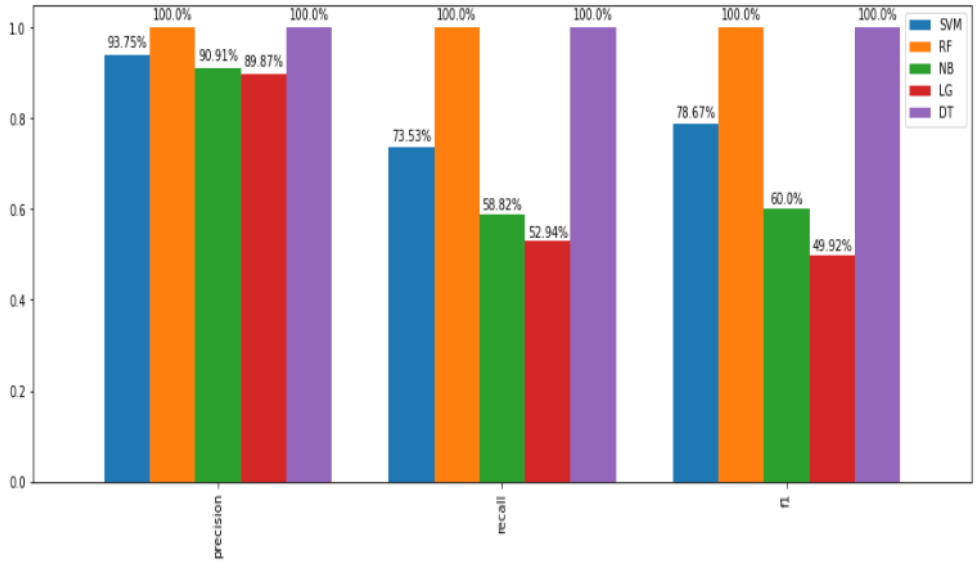



**Figure 9: Getting tweets using sentiment analysis**

```
SVM Accuracy Score on Train set ->  88.75
SVM Accuracy Score on Validation set ->  90.0
              precision    recall  f1-score   support

    negative       0.90      1.00      0.95        18
    positive       0.00      0.00      0.00         2

    accuracy                           0.90        20
   macro avg       0.45      0.50      0.47        20
weighted avg       0.81      0.90      0.85        20

RandomForest Accuracy Score on Train set ->  100.0
RandomForest Accuracy Score on Validation set ->  90.0
              precision    recall  f1-score   support

    negative       0.90      1.00      0.95        18
    positive       0.00      0.00      0.00         2

    accuracy                           0.90        20
   macro avg       0.45      0.50      0.47        20
weighted avg       0.81      0.90      0.85        20

Naive Bayes Accuracy Score on Train set ->  82.5
Naive Bayes Accuracy Score on Validation set ->  90.0
              precision    recall  f1-score   support

    negative       0.90      1.00      0.95        18
    positive       0.00      0.00      0.00         2

    accuracy                           0.90        20
   macro avg       0.45      0.50      0.47        20
weighted avg       0.81      0.90      0.85        20
```

```
Logistic Regression Accuracy Score on Train set ->  80.0
Logistic Regression Accuracy Score on Validation set ->  90.0
              precision    recall  f1-score   support

    negative       0.90      1.00      0.95        18
    positive       0.00      0.00      0.00         2

    accuracy                           0.90        20
   macro avg       0.45      0.50      0.47        20
weighted avg       0.81      0.90      0.85        20

Decision tree Accuracy Score on Train set ->  100.0
Decision tree Accuracy Score on Validation set ->  85.0
              precision    recall  f1-score   support

    negative       0.94      0.89      0.91        18
    positive       0.33      0.50      0.40         2

    accuracy                           0.85        20
   macro avg       0.64      0.69      0.66        20
weighted avg       0.88      0.85      0.86        20
```

**Figure 10: Accuracy score of train and validation of twitter of various ML**

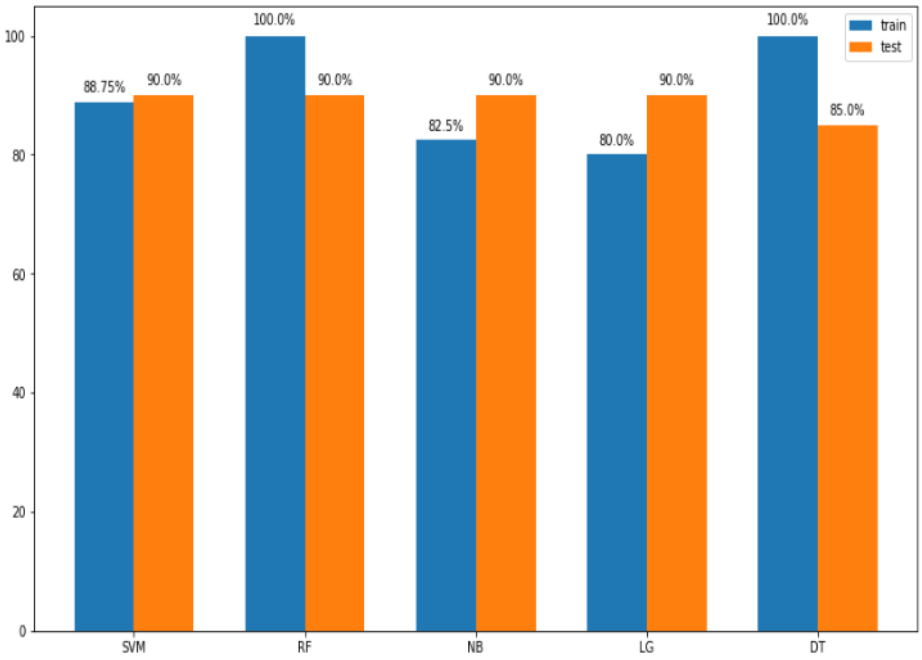



**X= Metrics**
**Y= 1 unit = 100%**

**Figure 11: Bar graph of various ML models using Sentiment Analysis on Twitter**

```
RMSE of SVM 1.25
RMSE of Random Forest 10.0
RMSE of Naive Bayes 7.5
RMSE of Logistic Regression 10.0
RMSE of Decision Tree 10.0
```

**Figure 12: RMSE value of various ML models**




**X= Metrics**

**Y= 1 Unit = 1%**

**Figure 13: Bar graph of Train and test data Twitter sentiment analysis**

## 6. Comparison Table

| Algorithms | RMSE Value of Facebook | RMSE Value of Twitter |
|---|---|---|
| SVM | 7.78 | 1.25 |
| Random Forest | 23.87 | 10.0 |
| Naïve Bayes | 7.77 | 7.5 |
| Logistic Regression | 8.34 | 10.0 |
| Decision Tree | 8.34 | 10.0 |

**Table 1: RMSE value of Twitter and Facebook**

| Algorithms | Facebook Accuracy on train set | Twitter Accuracy on train set | Facebook Accuracy on test set | Twitter Accuracy on test set |
|---|---|---|---|---|
| SVM | 93.43 | 88.75 | 85.65 | 90.0 |
| Random Forest | 99.80 | 100.0 | 75.92 | 90.0 |
| Naïve Bayes | 83.04 | 82.5 | 75.27 | 90.0 |
| Logistic Regression | 90.52 | 80.0 | 82.18 | 90.0 |
| Decision Tree | 99.80 | 100.0 | 69.72 | 85.0 |

**Table 2: Accuracy on train and test value of Twitter and Facebook**

| Algorithms | Precision (%) | Recall (%) | F1-score (%) |
|---|---|---|---|
| SVM | 93.8 | 91.3 | 92.4 |
| Random Forest | 99.9 | 99.7 | 99.8 |
| Naïve Bayes | 86.8 | 76.4 | 77.7 |
| Logistic Regression | 91.7 | 87.6 | 89.1 |
| Decision Tree | 99.8 | 99.8 | 99.8 |

**Table 3: Precision, recall, F1-score of Facebook Using ML models**

| Algorithms | Precision (%) | Recall (%) | F1-score (%) |
|---|---|---|---|
| SVM | 93.75 | 73.53 | 78.67 |
| Random Forest | 100 | 100.0 | 100.0 |
| Naïve Bayes | 90.91 | 58.82 | 60.0 |
| Logistic Regression | 89.87 | 52.94 | 49.92 |
| Decision Tree | 100.0 | 100.0 | 100.0 |

**Table 4: Precision, recall, F1-score of Twitter Using ML models**

## 7. Conclusion

In the project, it has generalized five different Machine Learning Algorithms for sentiment analysis of the Facebook data and the Twitter tweets. First it will search and find out the dataset and will download it for train the model. Then will preprocess the data and then transferred to Tf-Idf after downloading the dataset first. Now using NB, SVM, DT, LR, RF; It will train dataset and generate model separately which is done successfully The presentation of these algorithm will make a good understanding for making the performance better. Any individual can predict which algorithm will work far better for making sentiment analysis. The post and status are posted in large number where the best technique will prove to be the best out of it.

## 8. Future Modification

In future it is expected to improve the system created by utilize more accurate dataset and to detect the cyberbullying or not. It likewise applies other machine learning algorithm and check the accuracy of models. Higher accuracy model will assist with detecting more accurate cybercrime. It is especially fascinating in order to checking purposes. At the point when applied in a various ML model, the system could discover serious detection of cyberbullying with high precision.

## 9. References

[1] Di Capua, M., Di Nardo, E., & Petrosino, A. (2016, December). Unsupervised cyber bullying detection in social networks. In *2016 23rd International conference on pattern recognition (ICPR)* (pp. 432-437). IEEE.

[2] Mody, A., Shah, S., Pimple, R., & Shekokar, N. (2018, December). Identification of Potential Cyber Bullying Tweets using Hybrid Approach in Sentiment Analysis. In *2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)* (pp. 878-881). IEEE.

[3] Mangaonkar, A., Hayrapetian, A., & Raje, R. (2015, May). Collaborative detection of cyberbullying behavior in Twitter data. In *2015 IEEE international conference on electro/information technology (EIT)* (pp. 611-616). IEEE.

[4] Karthika, P., Murugeswari, R., & Manoranjithem, R. (2019, April). Sentiment Analysis of Social Media Network Using Random Forest Algorithm. In *2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)* (pp. 1-5). IEEE.

[5] Mouheb, D., Albarghash, R., Mowakeh, M. F., Al Aghbari, Z., & Kamel, I. (2019, November). Detection of Arabic Cyberbullying on Social Networks using Machine Learning. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)* (pp. 1-5). IEEE.

[6] Abd El-Jawad, M. H., Hodhod, R., & Omar, Y. M. (2018, December). Sentiment analysis of social media networks using machine learning. In *2018 14th international computer engineering conference (ICENCO)* (pp. 174-176). IEEE.

[7] Gupta, A., Singh, A., Pandita, I., & Parashar, H. (2019, March). Sentiment Analysis of Twitter Posts using Machine Learning Algorithms. In *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 980-983). IEEE.

[8] Uddin, A. H., Bapery, D., & Arif, A. S. M. (2019, July). Depression Analysis from Social Media Data in Bangla Language using Long Short Term Memory (LSTM) Recurrent Neural Network Technique. In *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)* (pp. 1-4). IEEE.

[9] Kalaivani, P., & Dinesh, D. (2020, September). Machine Learning Approach to Analyze Classification Result for Twitter Sentiment. In *2020 International Conference on Smart Electronics and Communication (ICOSEC)* (pp. 107-112). IEEE.

[10] Sugandhi, R., Pande, A., Chawla, S., Agrawal, A., & Bhagat, H. (2015, December). Methods for detection of cyberbullying: A survey. In *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)* (pp. 173-177). IEEE.

[11] Sintaha, M., & Mostakim, M. (2018, December). An empirical study and analysis of the machine learning algorithms used in detecting cyberbullying in social media. In *2018 21st International Conference of Computer and Information Technology (ICCIT)* (pp. 1-6). IEEE.

[12] Purba, K. R., Asirvatham, D., & Murugesan, R. K. (2018, October). A Study on the Methods to Identify and Classify Cyberbullying in Social Media. In *2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA)* (pp. 1-6). IEEE.

[13] Singh, S., Thapar, V., & Bagga, S. (2020). Exploring the hidden patterns of cyberbullying on social media. *Procedia Computer Science*, *167*, 1636-1647.

[14] Nurrahmi, H., & Nurjanah, D. (2018, March). Indonesian twitter cyberbullying detection using text classification and user credibility. In *2018 International Conference on Information and Communications Technology (ICOIACT)* (pp. 543-548). IEEE.

[15] Ali, K., Dong, H., Bouguettaya, A., Erradi, A., & Hadjidj, R. (2017, June). Sentiment analysis as a service: a social media based sentiment analysis framework. In *2017 IEEE International Conference on Web Services (ICWS)* (pp. 660-667). IEEE.

# Cyber Bullying Recognition System for Facebook, Twitter Using ML Algorithms for Sentiment Analysis

ORIGINALITY REPORT

| 13% | 10% | 2% | 12% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | Submitted to VIT University<br>Student Paper | 4% |
| 2 | Submitted to Monash University<br>Student Paper | 1% |
| 3 | www.coursehero.com<br>Internet Source | 1% |
| 4 | Submitted to Indian Institute of Technology, Madras<br>Student Paper | 1% |
| 5 | medium.com<br>Internet Source | 1% |
| 6 | Submitted to Yogi Vemana University, Vemanapuram<br>Student Paper | 1% |
| 7 | Submitted to Marmara University<br>Student Paper | 1% |
| 8 | Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK)<br>Student Paper | 1% |

| | | |
|---|---|---|
| 9 | dhirajkumarblog.medium.com<br>Internet Source | <1% |
| 10 | Submitted to Napier University<br>Student Paper | <1% |
| 11 | Huan-Kai Peng. "Retweet Modeling Using Conditional Random Fields", 2011 IEEE 11th International Conference on Data Mining Workshops, 12/2011<br>Publication | <1% |
| 12 | www.edureka.co<br>Internet Source | <1% |
| 13 | Submitted to University of Westminster<br>Student Paper | <1% |
| 14 | scholars.lib.ntu.edu.tw<br>Internet Source | <1% |
| 15 | ethesis.nitrkl.ac.in<br>Internet Source | <1% |
| 16 | Submitted to Universiti Malaysia Pahang<br>Student Paper | <1% |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 10 words |
| Exclude bibliography | On | | |