

Dissertation Titled

“Depression Detection on Social Media using Machine Learning Technique”

Submitted in partial fulfillment of the requirements of the degree of Master of Technology

By

SUYASH DABHANE

(192050006)

M. Tech. (Computer Engineering)

2020-21

Under the guidance of

Mrs. P. M. CHAWAN



Department of Computer Science and Information Technology

Veermata Jijabai Technological Institute

(An Autonomous institute affiliated to University of Mumbai)

Mumbai-400019

2020-21

STATEMENT OF CANDIDATE

I state that work embodied in this Project entitled “Depression Detection on Social Media using Machine Learning Technique” forms my own contribution of work under the guidance of Prof. P. M. Chawan at the Department of Computer Engg. And Information Technology, Veermata Jijabai Technological Institute. The report reflects the work done during the period of candidature but may include related preliminary material provided that it has not contributed to an award of previous degree. No part of this work has been used by me for the requirement of another degree except where explicitly stated in the body of the text and the attached statement.

(Signature)

Suyash Dabhane

Roll No: 192050006

Date: _____

APPROVAL SHEET

The dissertation entitled, “Depression Detection on Social Media using Machine Learning Technique” by Suyash Dabhane (192050006) is found to be satisfactory and is approved as for the degree of Master of Technology.

Examiner

Prof. P. M. Chawan

Project Guide

Date: _____

Place: VJTI, MUMBAI

CERTIFICATE

This is to certify that Mr. Suyash Dabhane [192050006], a student of M. Tech. II (Computer Engineering), Veermata Jijabai Technological Institute (VJTI), Mumbai has successfully completed the Master of Technology Dissertation entitled “**Depression Detection on Social Media using Machine Learning Technique**” under the guidance of Mrs. P. M. Chawan.

Prof. P. M. Chawan

Project Guide

Examiner

Head, CS & IT department

ACKNOWLEDGEMENT

I would like to thank all those people whose support and cooperation has been an invaluable asset during the course of this Project. I would also like to thank my Guide Mrs. P. M. Chawan for guiding me throughout this project and giving it the present shape. It would have been impossible to complete the project without their support, valuable suggestions, criticism, encouragement and guidance.

I convey my gratitude also to Head of Department for his motivation and providing various facilities, which helped us greatly in the whole process of this project.

I am also grateful for all other teaching and non-teaching staff members of the Computer Technology for directly or indirectly helping us for the completion of this project and the resources provided.

Suyash Dabhane

SYNOPSIS

INTRODUCTION

Depression is a dysfunctional behavior that can influence anybody regardless of old enough, gender, status, and so forth. It extremely brunt a person's life affecting what they think about themselves, their sleeping cycle, eating cycle, etc. It is the worst state of a person's mind when they feel sad and loses interest in nearly doing every productive thing and they can't simply move from that state. Factors like Social, Biological and psychological factors are responsible for causing depression. Depression also causes other physical illnesses. Self-destruction is the subsequent reason for death in 15-29-year-olds due to depression. Using a machine learning approach to detect depression will surely help people and social media users for detecting and predicting depression risk. It additionally helps social media users to seek early help to overcome depression. A machine learning approach like supervised learning can analyse and build a model on social media posts like Twitter or Reddit posts. There are many factors like users' posts, tweets, replies, post time, emotions, etc. which can contribute to detecting depression. To classify the data or tweets that are depressive or not we will use machine learning algorithms and natural language processing. Before making a model, we will do exploratory data analysis on our dataset to thoroughly understand it.

PROBLEM STATEMENT

“To analyze the social media data of users like Twitter feed and detect depression by using various machine learning techniques”

Depression is a leading cause of mental ill-health, which has been found to increase the risk of early death. However, 70% of the patients would not consult doctors at a stage of depression. Meanwhile, people increasingly rely on social media for sharing emotions, and daily life activities thus helpful for detecting their mental health. We aim to apply Machine Learning Techniques on Social Data of a user like Twitter feeds for performing analysis focusing on depression detection.

Report Orientation

The report will compromise of four chapters with different content and scenarios providing the complete details about the project. The report is completed in such a way that it first provides the background Knowledge about the project and then gives the thorough details

about it. It also has methodology and the conclusion. The different chapters of the report are as follows.

Chapter 1: Background and Motivation

This chapter will provide introduction to the project and motivation for performing it. This chapter describes current scenario, previous attempts, approach, and Problem statement. This chapter will also introduce different machine learning algorithms that we are going to use in our project.

Chapter 2: Literature Review

This chapter will provide literature studied of the project. The focus would be on the technology going to be used. In this we give detailed explanation of evolution of the machine learning algorithms and their techniques.

Chapter 3: Proposed Methodology and System Design

This chapter is basically divided into Dataset Generation, Exploratory data analysis and System Design. In this chapter we have UML diagram like Use case, project flow diagram etc. In it we also focus on detail information of the data sets.

Chapter 4: Implementation – Model Training and Web Deployment

This chapter is basically divided into training our dataset using various machine learning algorithms as well as implementing ensemble learning techniques like voting classifier, stacking classifier etc. After model training, we have deployed our model on a web application to simplify end user interaction with system.

Chapter 5: Conclusion

This chapter gives the conclusion of the project and will also give a brief about future scope for this idea and methodology of the project.

ABSTRACT

Depression is a common but serious mental health disorder. Still, most people dealing with depression do not approach doctors for this problem. On the other hand, the use of Social Media Sites like Twitter is expanding extremely fast. Nowadays, people tend to rely on these social media platforms to share their emotions and feelings through their feed. Thus, this readily available content on social media has become helpful for us to analyse the mental health of such users. We can apply various machine learning techniques on this social media data to extract the mental health status of a user focusing on Depression. Detecting texts that express negativity in the data is one of the best ways to detect depression. In this project, we highlighted this problem of depression and discussed various techniques on how to detect it. we implemented a system that can detect if a person on social media is going through depression or not by analysing the user's data and activities by using various machine learning techniques.

Keywords: Depression Detection, Machine Learning, Natural Language Processing, Ensemble Learning, Twitter, social media

Arrangement of contents of project report

1. Title sheet
2. Statement of the Candidate
3. Approval Sheet
4. Certificate Sheet
5. Acknowledgement
6. Synopsis
7. Abstract
8. Table of contents
9. Content
10. References
11. Paper Published

Table of Content

1.	Introduction.....	1
1.1	Introduction.....	2
1.2	Conventional Machine Learning Model Building	3
1.2.1	Data Collection	4
1.2.2	Data Preparation.....	4
1.2.3	Training & Testing.....	4
1.3	Problem Statement	5
1.4	Machine Learning Algorithms.....	5
2.	Literature Review.....	10
3.	Proposed Methodology and System Design	14
3.1	Objectives	14
3.2	Significance.....	14
3.3	Proposed Method	17
3.4	Proposed System Architecture.....	19
3.4	Analysis of the System.....	20
3.4.1	Use case Diagram	21
3.4.2	System flow Diagram	21
3.4.3	Activity Diagram	22
4.	Implementation	24
4.1	Working Environment	25
4.2	Data Collection	25
4.3	Data Cleaning.....	27
4.4	Exploratory Data Analysis.....	28
4.4.1	Pie Chart.....	29
4.4.2	Bar Plot and Word Cloud.....	29

4.5 Data Preparation.....	30
4.6 Model Training	30
4.6.1 Implementing base paper models.....	31
4.6.1.1 Naïve Bayes.....	31
4.6.1.2 Support Vector Machine.....	31
4.6.2 Implementing other Machine Learning Models	32
4.6.3 Implementing Ensemble Learning Techniques	32
4.6.3.1 Voting Classifier	33
4.6.3.2 Stacking Classifier	33
4.7 Program Code	33
4.7.1 Data Preprocessing and Model Training.....	33
4.7.2 Web Deployment.....	41
4.8 Web Application Snapshots.....	48
5. Conclusion	50
5.1 Conclusion	50
5.2 Future Scope	50
References.....	53
Appendix: Research Paper Published	57

List of Figures

Figure 1: Steps involved in building a machine learning model

Figure 2: One hidden layer MLP

Figure 3: Relationship between learning curve and model complexity

Figure 4: Base architecture of ensemble learning

Figure 5: The framework of ensemble classification

Figure 6: Existing system

Figure 7: Proposed system workflow for training a model

Figure 8: Use case Diagram

Figure 9: System flow Diagram

Figure 10: Activity Diagram for Performing Depression Detection

Figure 11: Time Frame

Lists of Snapshots

Snapshot 1: Twitter developer account approved

Snapshot 2: Twitter API Key

Snapshot 3: Tweets data creation: Depressive tweets

Snapshot 4: Tweets data creation: Non-Depressive tweets

Snapshot 5: Data Preprocessing

Snapshot 6: Pie chart code

Snapshot 7: Pie chart

Snapshot 8: Bar plot

Snapshot 9: Word cloud code

Snapshot 10: Word Cloud

Snapshot 11: Naïve Bayes Training

Snapshot 12: SVM Training

Snapshot 13: Voting Classifier Training

Snapshot 14: Stacking Classifier Training

Snapshot 15: Web application GUI

Snapshot 16: Twitter user prediction: Depressive

Snapshot 17: Twitter user prediction: Non-Depressive

Snapshot 18: prediction of single text: Non-Depressive

Snapshot 19: prediction of single text: Depressive

1. Introduction

Introduction

1.1 Introduction

Depression is a mental illness that can affect anyone regardless of gender, status, etc. It extremely impacts a person's life affecting what they think about themselves, their sleeping cycle, eating cycle, etc. It is a state of a person's mind when they feel sad and loses interest in nearly doing every productive thing and they can't simply move from that state. Social, Biological, and psychological factors are responsible for causing depression. Depression also causes other physical illnesses. Suicide is the second leading cause of death in 15-29-year-olds.

Using a machine learning approach to detect depression will help social media users for detecting and predicting depression risk. It also helps people to seek help to fight depression as soon as possible. A machine learning approach like supervised learning can analyse and build a model on social media posts like Twitter or Reddit posts. There are many factors like users' posts, tweets, replies, etc. which can detect depression. In this report, I have discussed different techniques that can be used to detect depression from social media data for e.g. Twitter data and how these techniques differ from each other and so on.

The social media of a user plays a key role in understanding many aspects of that person. The following points play an important role in depression: overthinking, pressure, rejection, etc.

➤ What Causes Depression?

Many factors are responsible for causing depression; it includes personal factors, recent events, and long-term events. Life events like unemployment, failure in exams, rejections etc. may lead to depression. Personal factors like the personality of a person, long term illness, family history, addiction etc. may also be responsible for depression in a person.

1.2 Conventional Machine Learning Model Building

Any machine learning algorithm involved the following steps to build a model. The main objective of the machine learning algorithm is to find the pattern in data and provide accurate predictions.

Following are seven important steps required to build a machine learning model:

- **Data collection:** This is the first step before building any machine learning model. Accuracy can affect if the data is less. For supervised machine learning, this data is labelled and for unsupervised machine learning models, the data is unlabeled.
- **Data Preparation:** The data we collect is present in its raw form. We need to clean the data with the help of data preprocessing techniques. After that, we perform exploratory data analysis to get a glimpse of patterns in data and to check if the data we collected is right or not. This step takes a lot of time. The cleaner our data the more positive impact it will make on the model.

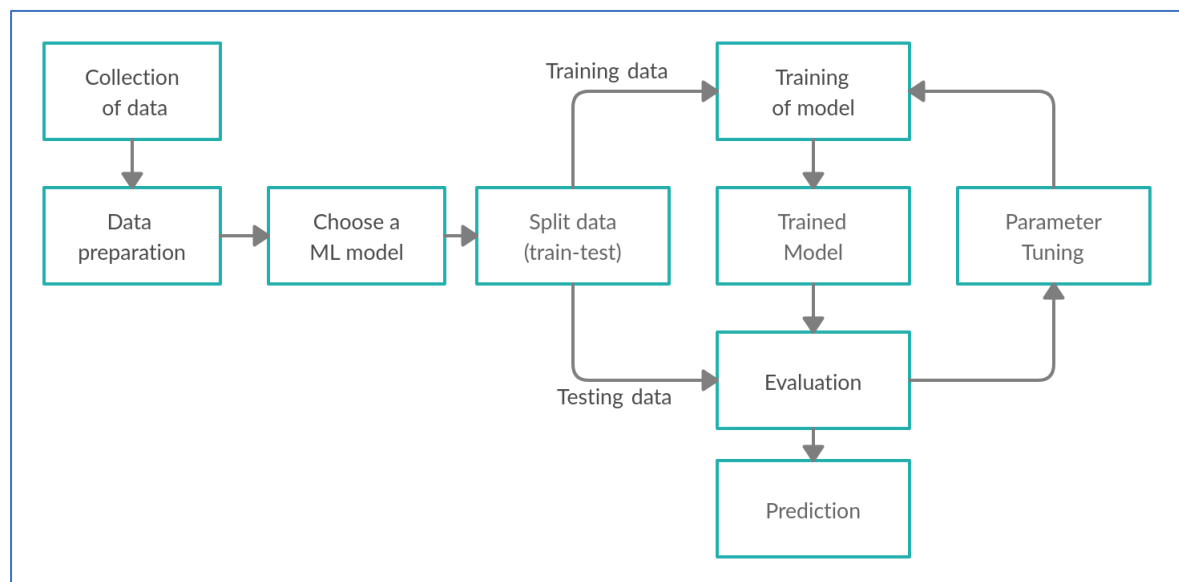


Fig 1. Steps involved in building a machine learning model

- **Choose a machine learning model:** The choice of a model depends on what type of data we have and what we want to predict. the right model should be selected. for example, If the data is labeled then we need to use a supervised machine learning model.
- **Splitting the data set:** Data should be split into training and testing sets. Training data is used for training a model and testing data will be used for making an evaluation.
- **Training:** Train a model on the training data. While the training machine learning model learns the pattern in data.

- **Evaluation:** The evaluation of the model is done on training data. The evaluation will give how well our model is performing on testing or validation data.
- **Parameter Tuning:** Tuning parameters involve changing the value of parameters in such a way that our model will give more accuracy and ultimately more accurate prediction.
- **Prediction:** Process of using a train machine learning model to predict the output of given value.

1.3 Problem Statement

“To analyse the social media data of users like Twitter feed and detect depression by using various machine learning techniques”

Depression is a leading cause of mental ill-health, which has been found to increase the risk of early death. However, 70% of the patients would not consult doctors at a stage of depression. Meanwhile, people increasingly rely on social media for sharing emotions, and daily life activities thus helpful for detecting their mental health. We aim to apply Machine Learning Techniques on Social Data of a user like Twitter feeds for performing analysis focusing on depression detection.

1.4 Machine Learning Algorithms and Methodologies

Mainly machine learning algorithms are of two types supervised machine learning algorithms and unsupervised machine learning algorithms.

- **Supervised machine learning:**

This type of machine learning model gets its training on the labelled dataset. The dataset which has input, as well as output/ label, is called a labelled dataset.

- **Unsupervised machine learning:**

This type of machine learning model gets trained only on input data. The dataset available for training doesn't contain its respective output/label.

➤ **Types of supervised machine learning algorithms:**

There are a total of two types of supervised machine learning algorithms classification and regression. As the dataset for this project is a labelled dataset, we will review various types of classification algorithms:

- Naive Bayes
- K Nearest Neighbor
- Support Vector Machines (SVM)
- Decision Trees
- Logistic Regression
- Multi-layer Perceptron

A. Naive Bayes

This algorithm is based on the Bayes theorem. This model is easy to build. It can be applied to a large dataset. For making real-time predictions we can use the Naive Bayes algorithm as it is fast. Naive Bayes assumes that the presence of one feature is independent of other features. There are three variations of naive Bayes classifier Bernoulli, Multinomial, and gaussian.

Bayes' Theorem is stated as:

$$P(A|B) = (P(B|A) * P(A)) / P(B)$$

$P(A)$: The prior probability of A. The probability of hypothesis A is always true (in any case of data).

$P(B)$: is the probability of the data (in any case of the hypothesis).

$P(A|B)$: it is a Posterior probability. the probability of hypothesis A as data B is given.

$P(B|A)$: is the probability of data B given that the hypothesis h was true.

B. K Nearest Neighbor

KNN is mostly used for classification problems. KNN stores all provided cases and then it classifies new input based on which cases are majorly near to new input. The class of new input is decided based on the most common K classes nearest to the new input. The distance between the new input and its neighbors is calculated using different distance functions like Manhattan, Hamming distance, etc. KNN doesn't learn any model. For KNN with the help of normalization, we can rescale our data. With the help of rescaling, normalizing and low-dimensional data KNN can perform well.

C. Support Vector Machines (SVM)

In this algorithm data points available are plotted on 2D space. SVM required less memory. SVM gives better results in high dimensional space. SVMs are good at finding the best linear separator. For nonlinear separation, we need to use a kernel. This kernel makes SV learn non-learning. We need to choose the proper kernel value. The hyperplane is drawn which separates different classes of datasets. The data points which are present nearer to the hyperplane are considered or called a support vector. With the help of a maximum marginal classifier, we can easily explain the support vector machine. Linear kernel, polynomial kernel, and radial kernel are 3 types of kernels used in SVM.

D. Decision Trees

A decision tree looks identical to a flowchart. Decision trees get results on input. Internal nodes of the decision trees perform decisions. Missing values give no effect. Training of the model requires much time. The goal of the decision tree model is to predict the target value by learning the rules of the decision tree from the available data. The main feature of the decision tree is it can handle both categorical and numerical data. If the dataset has some biased classes then the decision tree creates a biased tree is one of the disadvantages of the decision tree.

E. Logistic Regression

Logistic regression is a classification algorithm that is based on probability. Logistic regression uses a function called the logit function. Output value is in between 0 and 1.

logistic regression example,

$$y = e^{(b_0 + b_1 * x)} / (1 + e^{(b_0 + b_1 * x)}) \dots\dots\dots(1)$$

x = input value

y = output value

b_0 = bias

b_1 = coefficient for input value

F. Multi-layer Perceptron

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot): \mathbb{R}^m \rightarrow \mathbb{R}^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X = x_1, x_2, \dots, x_m$ and a target y , it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 1 shows a one hidden layer MLP with scalar output.

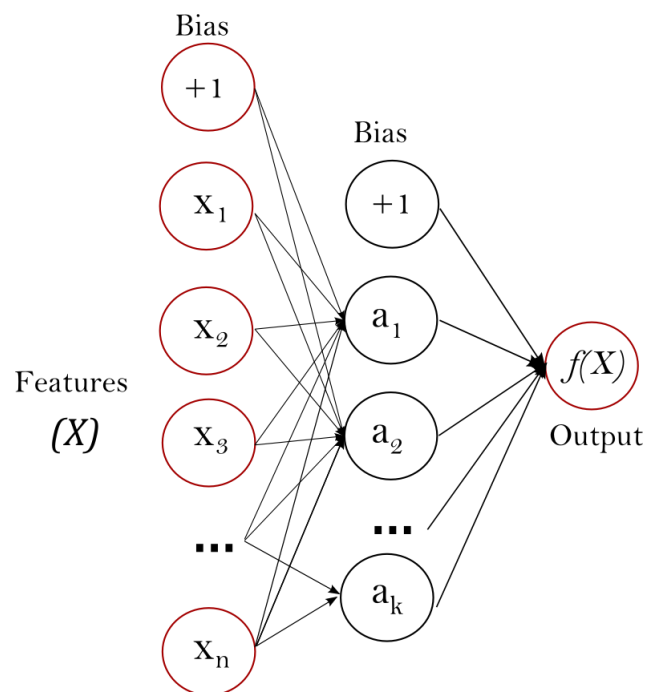


Fig. 2 One hidden layer MLP

The leftmost layer, known as the input layer, consists of a set of neurons $\{x_i | x_1, x_2, \dots, x_m\}$ representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + \dots + w_mx_m$, followed by a non-linear activation function $g(\cdot): \mathbb{R} \rightarrow \mathbb{R}$ - like the hyperbolic tan function.

The output layer receives the values from the last hidden layer and transforms them into output values.

The module contains the public attributes `coefs_` and `intercepts_`. `coefs_` is a list of weight matrices, where weight matrix at index `i` represents the weights between layer `i` and layer `i+1`. `intercepts_` is a list of bias vectors, where the vector at index `i` represents the bias values added to layer `i+1`.

The advantages of Multi-layer Perceptron are:

- Capability to learn non-linear models.
- Capability to learn models in real-time (on-line learning) using `partial_fit`.

The disadvantages of Multi-layer Perceptron (MLP) include:

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore, different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.

2. Literature Review

Literature Review

In machine learning, there are supervised machine learning classification algorithms for example Support Vector Machine, K Nearest Neighbor, Naive Bayes, Decision Tree, etc. Based on the different tasks and available data we can use those algorithms. The accuracy of the model can be improved using ensemble learning methods. There are different methods in machine learning which can be used to detect depression from posts of users. But each of them has its pros and cons which may affect a system in terms of accuracy and efficiency. There are simple ensemble methods like max voting, averaging, weighted averaging. but here we will use advanced ensemble techniques. We will first experiment with different ensemble learning models like bagging boosting and tagging.

Each classification algorithm with its pros and cons. For depression classification or detection, no method can be considered perfect. Depending on data in some cases Support Vector Machine is giving better accuracy and in other cases Multinomial Naive Bayes giving better accuracy. The supervised classification algorithms don't perform at the human level hence give a limited performance. A voting model with several classifiers can be used. With the help of this voting model, we can select features that are giving the maximum number of votes. But this method is not effective. For the model to be perfect we should have bias and variance in balance. One way to increase the performance of machine learning models is to ensemble them.

Amna Noureen, Usman Qamar, Mubashir Ali, et al. ^[1], in this paper the author has enlightened methods of classification for classifying psychotic behaviour and also compared different methods of classification. Also summarized analysis of each classification algorithm with its pros and cons. The author has to conclude that for psychotic behaviour classification no method can be considered perfect.

Priyanka Arora et al. ^[2] proposed supervised machine learning algorithms like Support Vector Machine and Multinomial Naive Bayes for analyzing tweets related to depression and anxiety. The dataset used for the training model consists of 3754 tweets. Proposed classifiers Multinomial Naive Bayes and Support Vector Machine model are giving an

accuracy of 78% and 79.7 % respectively. Need to improve the model for more classes and more accuracy.

Mandar Deshpande et al. ^[3], the author has used Naive Bayes and support vector machine classifiers for the prediction of class. Performance metrics like accuracy, confusion matrix, and F1-score are used to evaluate the performance of proposed classifiers. a 10,000 tweets dataset is used to train the model. Multinomial Naive Bayes classifier is giving an accuracy of 83% and SVM is giving an accuracy of 79%. Because of the limited performance of supervised classification, they don't perform at the human level. To make this system better False-positives need to be reduced.

Priyanka Gupta et al. ^[4], has presented and provided a review of different data analysis techniques to identify suicidal cases through social media data. The author has also summed up data collection and preprocessing techniques which are initial and essential parts of the machine learning pipeline. Mentioned list of words connected to suicide which helps to collect correct data.

Guangyao Shen and Jia Jia and Liqiang Nie et al. ^[5], proposed a model that performs better than many baselines. Created their own dataset of larger size and found 6 different feature groups related to depression. Some of these feature groups are user profile features, topic-level features, emotional features. Multimodal Depressive Dictionary Learning is a proposed method which is giving 85 % F1-measure as performance. This paper has also given a statistical analysis from depression data.

Nafiz Al Asad, Md. Appel Mahmud Pranto et al. ^[6] In this paper the author has proposed a model that takes the username as input and analyses posts of that particular user to check if posts are stating depression or not. There are a total of six categories (Normal, Mild, Moderate, etc.) which are considered to analyze depression. Facebook and Twitter's post dataset is used to train a model. The proposed Naive Bayes model of this paper gives an accuracy of 74%. The model can give the result of any individual going through depression or not with help of their username as input.

Swati Jain, Suraj Prakash Narayan, et al. ^[7], the author has trained a machine learning model on data with five different categories. The real-time dataset is collected by asking

questions to students and parents. These questions were similar to PHQ-9. This custom dataset is also responsible for giving high accuracy of the model. Before training the model Principal Component analysis (PCA) technique is used to reduce feature sets. The logistic regression model is giving more accuracy which is 86.45 %.

Anees Ul Hassan, Jamil Hussain, et al. ^[8], has illustrated a method of finding emotions from social networking data with help of machine learning and natural language processing. To identify depression an analysis of user posts can be done. A voting model is used which consists of 3 classifiers. With the help of the voting model, we will select features with maximum votes.

Purude Vaishali Narayanrao, P. Lalitha Surya Kumari, et al. ^[9], have analyzed different algorithms like classification algorithms and deep learning algorithms. Dataset is collected in different ways. So to train the proposed technique, a multisource dataset is used. The model build can classify data into two categories: depressive or positive.

Chiara Zucco, Barbara Calabrese, et al. ^[10], the author has discussed sentiment analysis and affective computing methods along with their applications. the author has proposed a multimodal system consisting of sentiment analysis and affective computing. The main challenges faced during the design and implementation of this multimodal system are also presented in this paper.

3. Proposed Methodology and System Design

Proposed Methodology and System Design

3.1 Objectives

- Perform depression analysis on Twitter data
- To make more accurate and efficient depression detection system
- To find the Twitter user is depressed or not from the previous 7-8 day.

3.2 Significance

- Study and improve classification and prediction techniques for depression detection.
- A system can be made where close ones of the person (like family members or friends) can search the username of a person and they get to know how depressive a person is in the last 7-8 days.
- Data analysis can be made by taking the data of Twitter in lockdown to know what is the depression rate before and after the covid-19 lockdown.

3.3 Proposed Method

There are different methods in machine learning which can be used to detect depression from social media posts of users. But each of them has its pros and cons which may affect a system in terms of accuracy and efficiency. Machine learning supervised algorithms like naive Bayes, support vector machine, k nearest neighbor, decision tree and logistic regression, etc. As studied, we observed that naive Bayes and support vector machine algorithms are giving better performance than other models. but the accuracy which we are getting is not good. The accuracy which was obtained is stuck between 85 to 88 %. So, to improve the performance of depression detection systems we need to use other powerful algorithms or we need to make changes in the same algorithms with hyperparameter tuning. One way to increase the performance of machine learning models is ensemble learning. The traditional machine learning algorithm follows steps like data collection, preprocessing, model selection, model training, evaluation, parameter tuning, and prediction as we have discussed in the introduction section.

With the help of ensemble learning, we combined different individual models to get the improved and powerful final model. In machine learning, we select different models based

on what data we have, what is the volume of data and what hypothesis we have to do. Any machine learning model gives errors. This error can be broken down into 3 parts: bias, variance, and irreducible error. The trend of bias and variance are opposite. For the model to be perfect we should have bias and variance in balance.

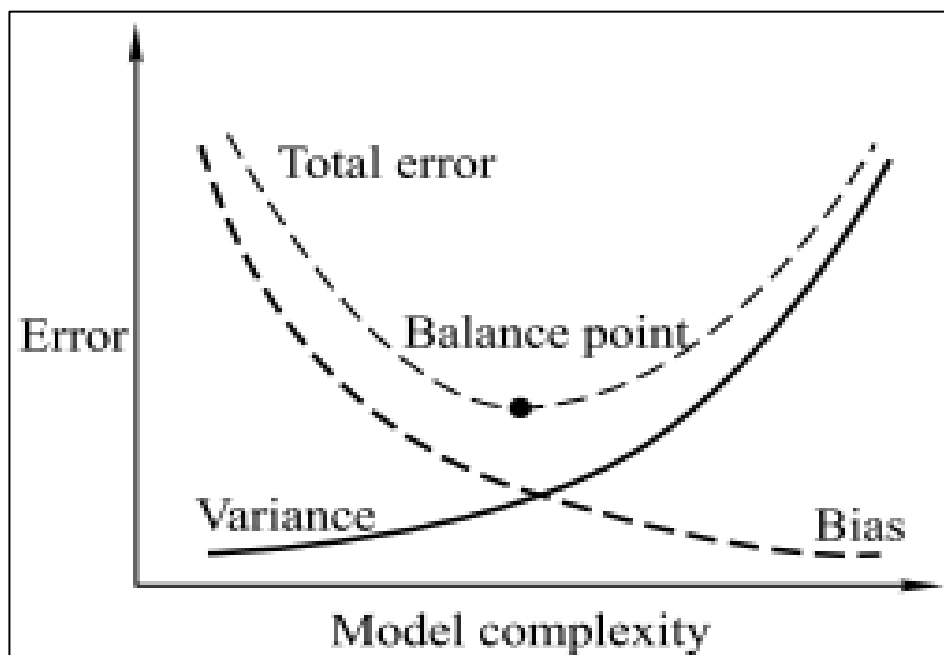
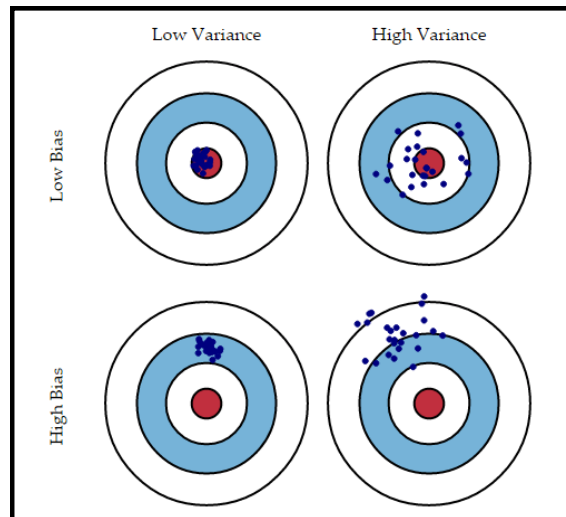


Fig. 3 Relationship between learning curve and model complexity

Ensemble learning balances this bias-variance trade-off to get perfect model complexity.

Performance improvement:

In the proposed system we will use ensemble learning methods to improve the accuracy of the existing system. There are simple ensemble methods like max voting, averaging, weighted averaging. but here we will use advanced ensemble techniques. We will first experiment with different ensemble learning models like bagging boosting and tagging. Fig 4. shows the base architecture of ensemble learning.

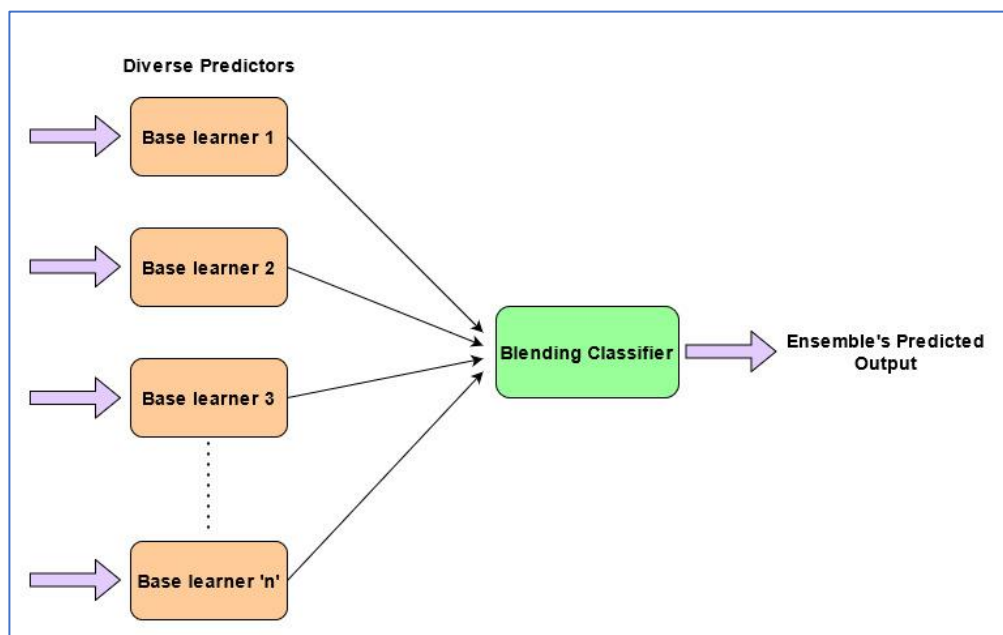


Fig. 4 Base architecture of ensemble learning

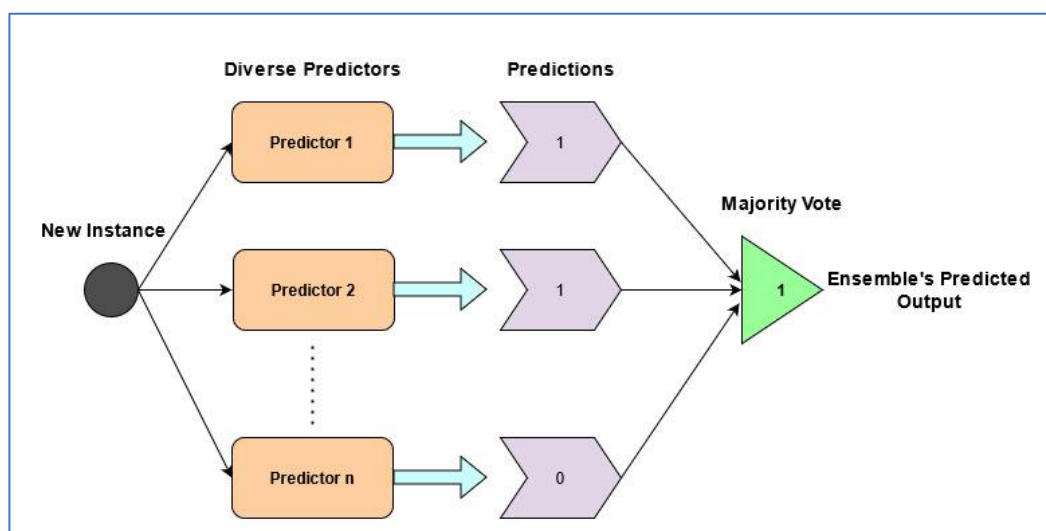


Fig. 5 The framework of ensemble classification

Stacking: In stacking, we train each base learner with a complete training dataset. We use meta learners to combine the results of all base learners. This meta learner model is trained on the output of base learner models. We can use different machine learning algorithms as base learners.

Bagging: In bagging, we divide a dataset into a number of base learners. We give this sub-dataset to each base learner and then we take the mean of all predictions obtained from base learners. Bagging helps in reducing variance.

Boosting: In boosting, there are strong learners and weak learners. With the help of boosting we can decrease bias error. Boosting converts weak learners to strong learners. It is an iterative process of building models. In boosting, base learners are trained using an iterative process. The disadvantage of boosting includes it sometimes overfits the data.

3.4 Proposed System Architecture:

In an existing system given in this paper [3], only an individual classifier is used as given in the training diagram. A multinomial naive Bayes classifier is used and gives 83 % accuracy.

We will use different individual classifiers and then we will apply ensemble methods with some modification to get accurate results.

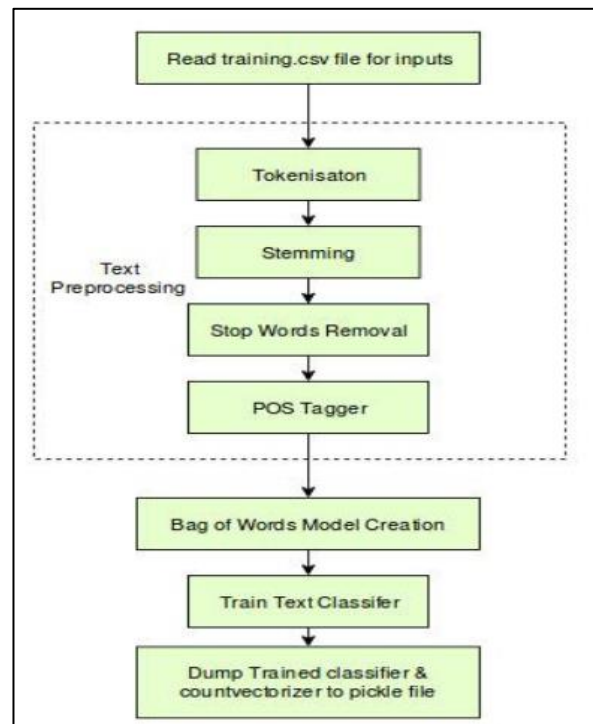


Fig. 6 existing system

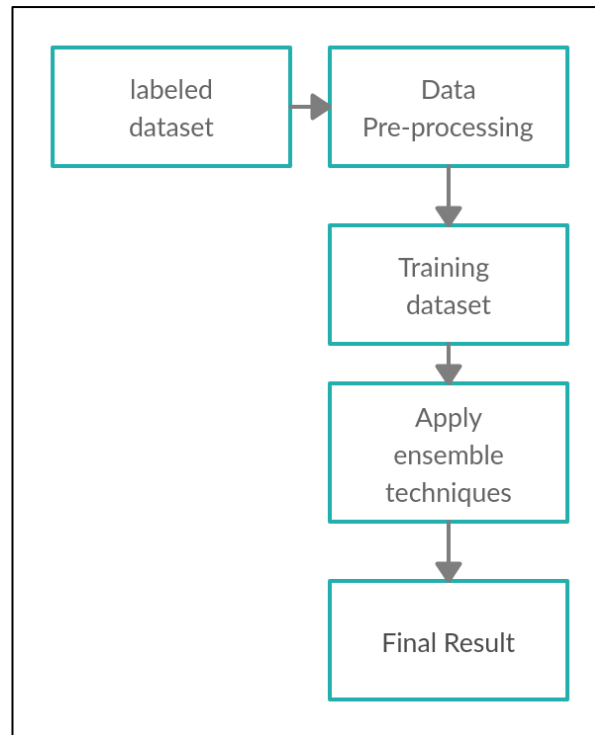


Fig. 7 proposed system workflow for training a model

As shown in the above block diagram a labeled social media post dataset is taken as input to our system. A dataset which we are going to collect is from Twitter. The preprocessing techniques we have already discussed. After preprocessing we will get the vector of inputs. We will use this training dataset to train individual classifiers or depending on which ensemble learning we are going to apply on data. We will apply ensemble techniques like stacking, bagging, and boosting. and using those techniques we will compare the performance of our proposed model with existing techniques used in the base paper. With the help of ensemble learning, we will successfully improve the accuracy of existing systems.

Following is the workflow of our project:

- I. **Data Collection:** Before training any model, we require data. We will collect the data of Twitter posts using Twitter API. The data is in the form of text. It contains users' tweets. The dataset is in its raw form. The collected dataset is in the raw form which contains information which is of no use like links, emoticons, images, GIF, videos, etc. we need to remove those things after removing them we will get the text data. The dataset is in the JSON format. We want the dataset in CSV file for that we will convert the JSON file

into CSV file and we will take only the tweets data which is the main goal. As our dataset needs a label we will add them manually.

II. Data Preprocessing: Data preprocessing involves steps like tokenization, stemming, lemmatization, stop word removal, PoS tagging, and finally we convert the text data in vector format using TF-IDF or Bag of Words.

III. Training: Before doing training, we will split our dataset into training and testing sets. As we are going to use ensemble learning techniques, we will use different classification algorithms as base learners with three types of ensemble methods. In the end, we will select the best method which will give better accuracy.

IV. Testing: We will test our model on testing data and we will evaluate the performance of our model using evaluation metrics like accuracy, precision, recall, F1-score, etc.

3.5 Analysis of the System

I. Use case Diagram

A use case Diagram is a set of scenarios that describe an interaction between the actor and the use case. It describes all the function and the processes. Use case diagram focuses on requirement functionality or object behavior.

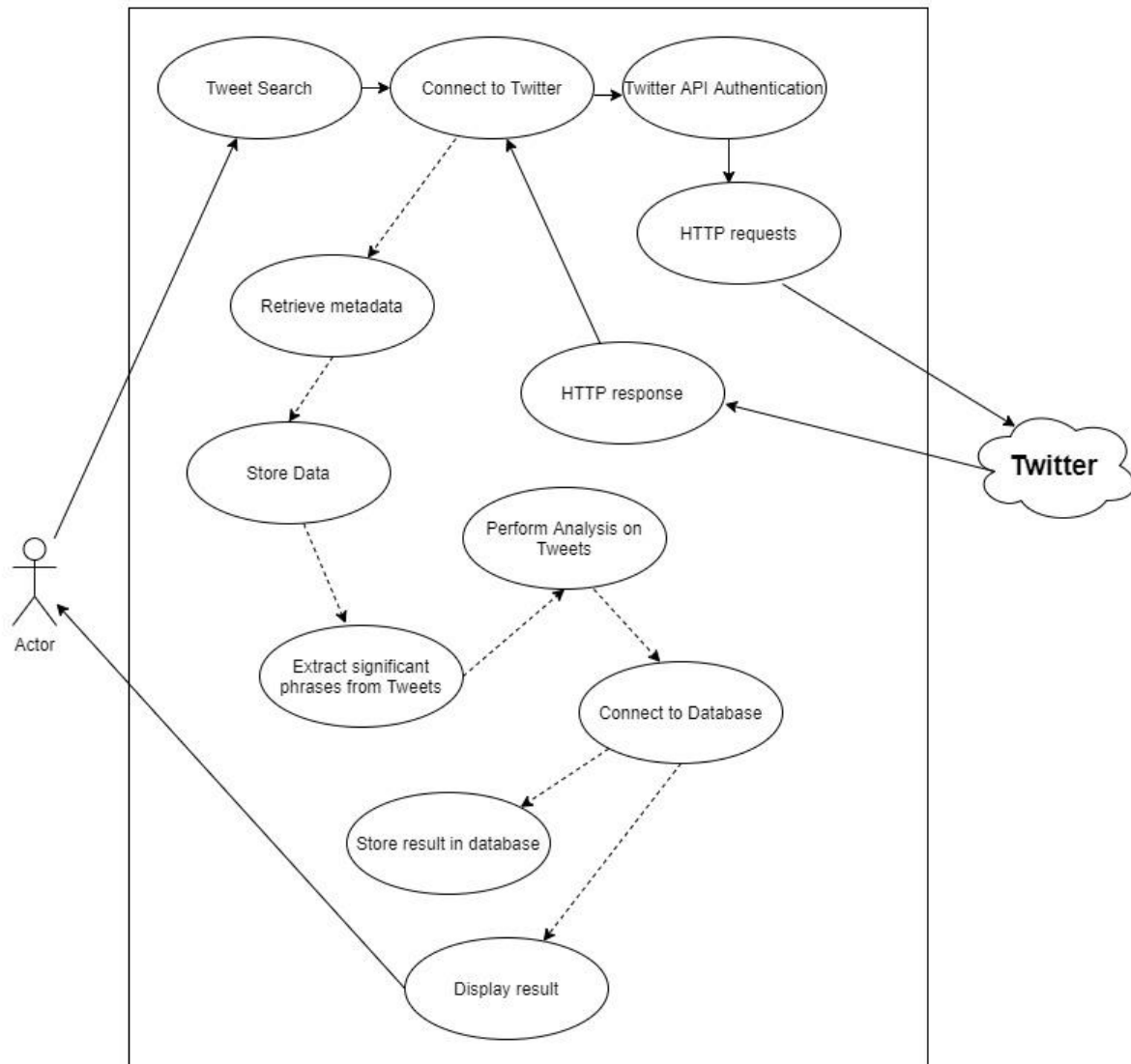


Fig. 8 Use case Diagram

System flow Diagram

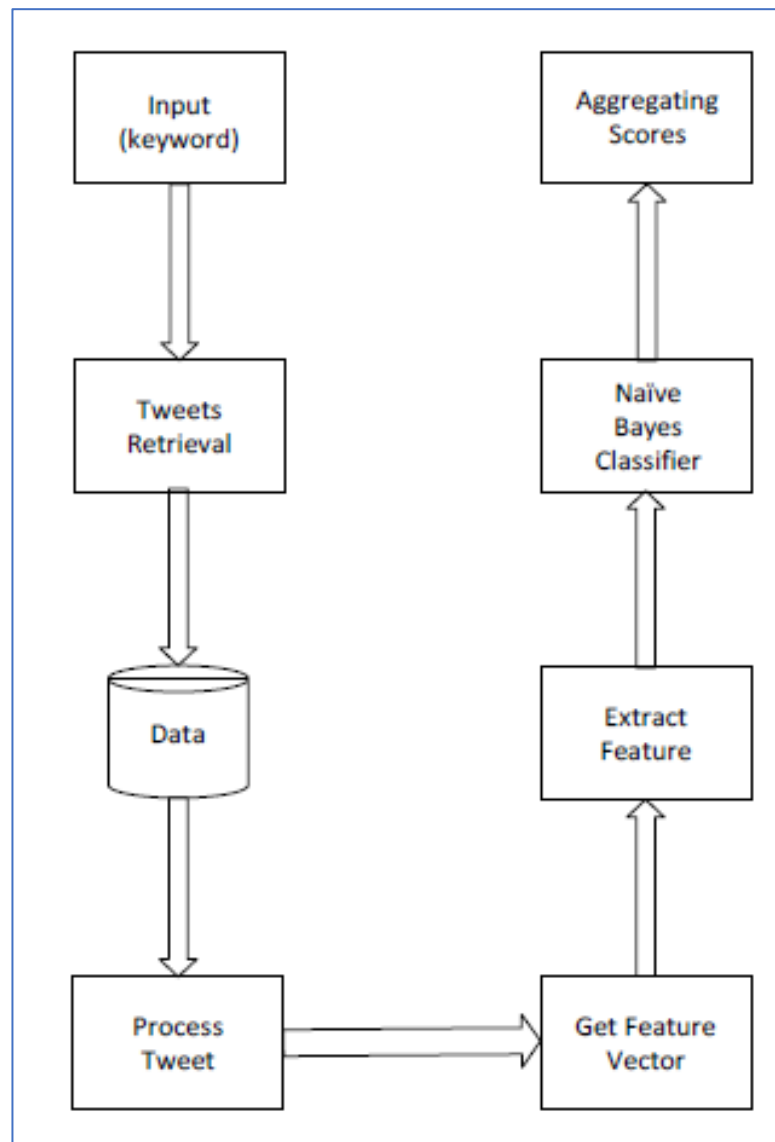


Fig. 9 System flow Diagram

Activity Diagram

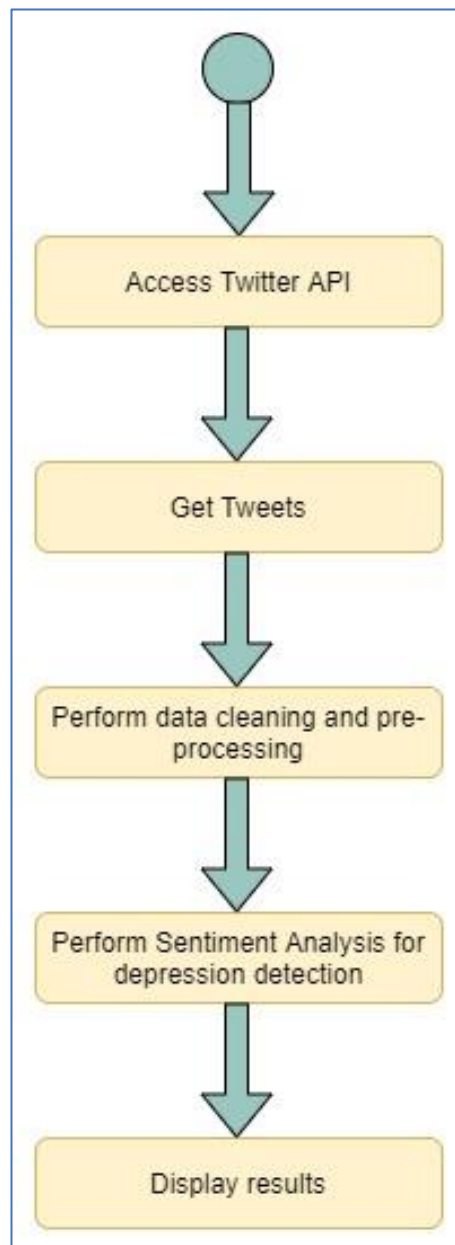


Fig. 10 Activity Diagram for Performing Depression Detection

Activity diagram is used for business process modeling, for modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule. Activity diagram is a dynamic diagram which shows the activity and the event that causes the object to be in the particular state. The easiest way to visualize activity diagram is to think of flowchart.

4. Implementation

Implementation

4.1 Working Environment

➤ Software Requirement

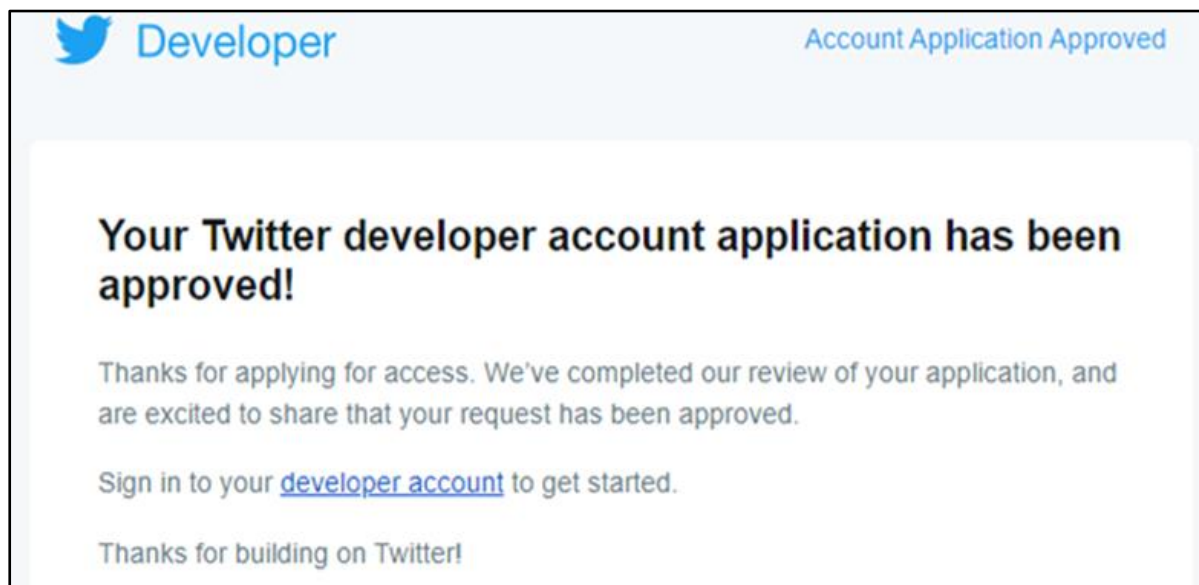
1. Operating System: Linux: Ubuntu 17.04 / Windows 10
2. Documentation: MS Word, MS PowerPoint.
3. Tool: Smart draw, StarUML, VSCode
4. Language: Python3.7
5. Libraries: numpy, sklearn, nltk, matplotlib, pandas, tweepy etc.

➤ Hardware Requirement

1. Processor: An Intel minimum 500 MHz processor recommended.
2. RAM: 4 GB.
3. HDD: 40 GB.

4.2 Data Collection

Applied for twitter API for collecting available tweets of users on which we will be performing depression detection analysis.



Snapshot 1: Twitter developer account approved

Activating the API key-

```
# input your credentials here
consumer_key = 'EcwAltC0olBlwQlesRg2Ux5A8'
consumer_secret = 'tMltHbelqqDAWJhoTSPa2TDrB4LWL4FK9B40F7qGsU7wUEmN9J'
access_token = '1331612828281683973-9SYgUjN2pYpoTVpONPRh3SCsMsMbb1'
access_token_secret = 'n901ArACONHH1EEExcCEn8BMApnGfjejHM6V0Cz3wtS5Mq'

#accessing data using twitter api via tweepy
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)
```

Snapshot 2: Twitter API Key

We needed twitter data such as tweets that are already available on twitter for training our model for depression detection. So, we have written a script in python which collects data from twitter and saves them locally into a .csv file.

In this script we specifically fetched tweets that contains words like 'depression', 'anxiety', 'mental health', 'suicide', 'stress', 'sad' for the dataset that contains all depressed tweets and on the other hand we also fetched tweets that contains words like 'joyful', 'cheerful', 'merry', 'happy', 'blissful', 'satisfied' and saved them into another .csv file which contain all the positive and non-depressive tweets.

```
#a csv file for storing depressive tweets
csvFile = open('tweet-dataset-depressive.csv', 'a')

# Use csv Writer
csvWriter = csv.writer(csvFile)
query = ['depression', 'anxiety', 'mental health', 'suicide', 'stress', 'sad']

csvWriter.writerow(["id", "timestamp", "tweet_text"])
for qu in query:
    print(qu)
    for tweet in tqdm(tweepy.Cursor(api.search, q=qu, tweet_mode="extended", \
                                   lang="en", since="2020-04-01").items(2000)):
        if (not tweet.retweeted() and ('RT @' not in tweet.full_text)):
            csvWriter.writerow([tweet.id, tweet.created_at, tweet.full_text.encode('utf-8')])

    print("Processing for the word {} finished".format(qu))
```

Snapshot 3: Tweets data creation: Depressive tweets

```
#csv file for storing normal tweets
csvFile = open('tweet-dataset-normal.csv', 'a')

# Use csv Writer
csvWriter = csv.writer(csvFile)
query = ['joyful', 'cheerful', 'merry', 'happy', 'blissful', 'satisfied']

csvWriter.writerow(["id", "timestamp", "tweet_text"])
for qu in query:
    print(qu)
    for tweet in tqdm(tweepy.Cursor(api.search, q=qu, tweet_mode="extended", \
                                   |lang="en", since="2020-01-01").items(1000)):
        if (not tweet.retweeted) and ('RT @' not in tweet.full_text):
            csvWriter.writerow([tweet.id, tweet.created_at, tweet.full_text.encode('utf-8')])

    print("Processing for {} finished".format(qu))
```

Snapshot 4: Tweets data creation: Non-Depressive tweets

4.3 Data Cleaning

After fetching the raw tweets and saving them locally. We have performed all possible time of data cleaning techniques to remove all unwanted data from our dataset.

Some of the highlights are-

1. Removing Emojis from the tweets
2. Removing digits, symbols and irrelevant characters.
3. Removing username and various kinds of tagging in the tweets
4. Removing hashtags from the tweets if any.
5. Removing URLs
6. Deleting entries with NULL values
7. Deleting duplicate entries

```

# function for replacing urls(starting from http) with an empty string.
def remove_urls(text):
    return re.sub(r'http\S+', '', text)

# function for replacing usernames(starting with @__) with an empty string
def remove_usernames(text):
    return re.sub('@[\^s]+', '', text)

def remove_hashtags(text):
    return re.sub('![@_%^*#$=-]', '', text)

def remove_emojis(text):
    allchars = [j for j in text]
    emoji_list = [c for c in allchars if c in emoji.UNICODE_EMOJI]
    clean_text = ' '.join([k for k in text.split() if not any(i in k for i in emoji_list)])
    return clean_text

def remove_digits(text):
    return re.sub(r'\b\d+\b', '', text)

def remove_digits_str(text):
    return re.sub(r'\d+', '', text)

```

Snapshot 5: Data Preprocessing

So, at the end of performing all above operations we will get clean texts in our dataset such that it will be easy to perform further operations like model training testing etc.

4.4 Exploratory Data Analysis

After creating the dataset for the model, we have performed exploratory data analysis on the dataset.

4.4.1 Pie chart for balanced dataset -

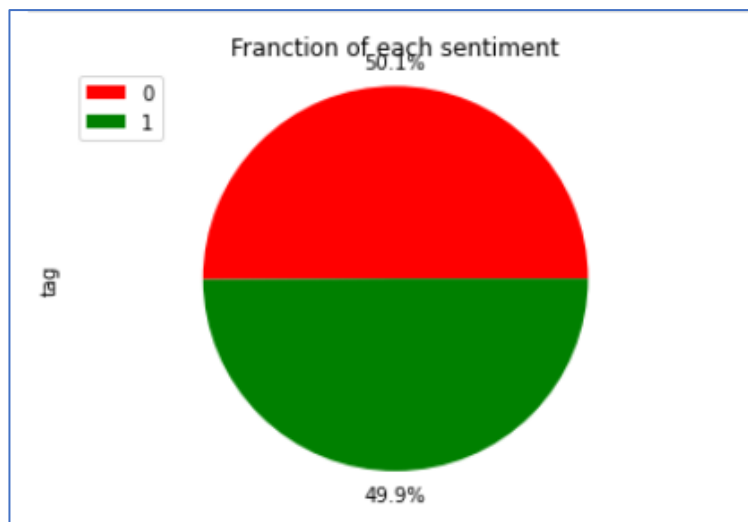
```

df_final.tag.value_counts().plot(kind="pie",
                                autopct='%1.1f%%',
                                labels=None,
                                pctdistance=1.12,
                                colors=["red", "green"])

plt.axis('equal')
plt.title("Fraction of each sentiment")
plt.legend(labels=df_final.tag.value_counts().index, loc="upper left")
plt.show()

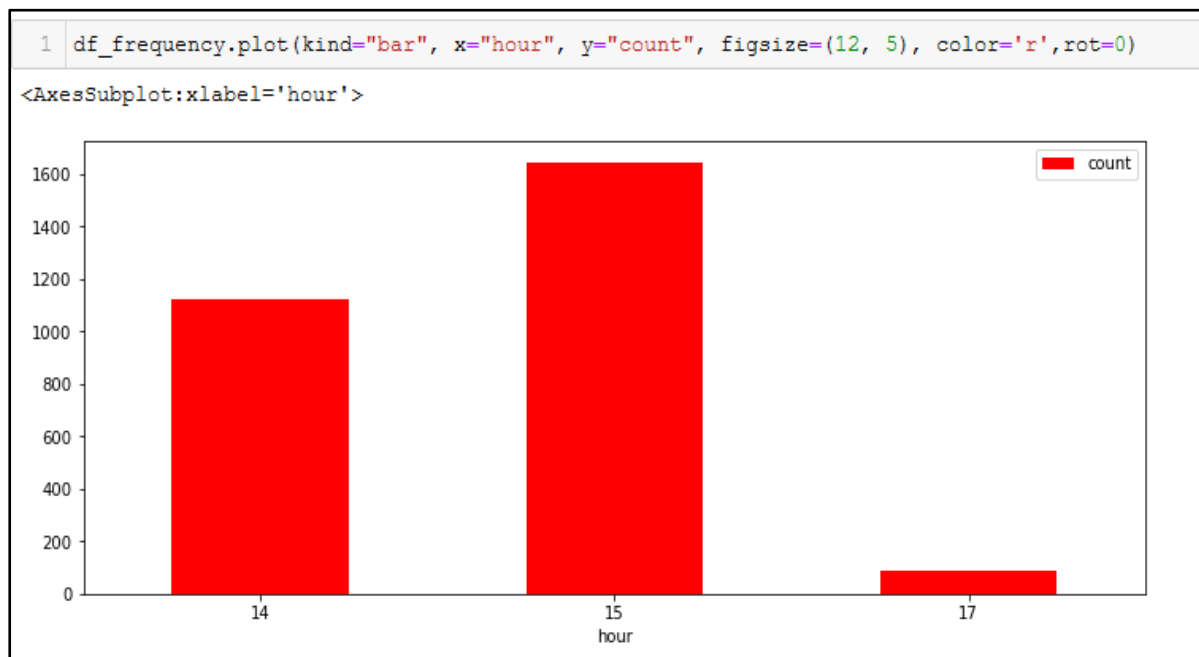
```

Snapshot 6: Pie chart code



Snapshot 7: Pie chart

4.4.2 Bar plot -

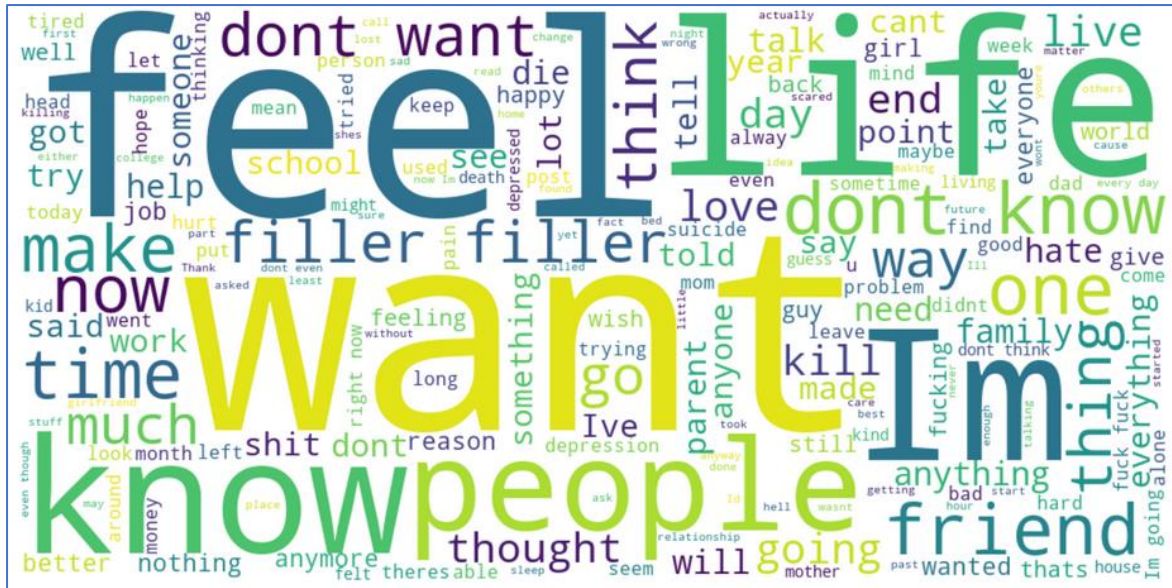


Snapshot 8: Bar plot

Word Cloud –

```
plt.rcParams["figure.figsize"] = 30, 30
wordcloud = create_wordcloud(set(df_final.text), "white")
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
```

Snapshot 9: word cloud code



Snapshot 10: Word Cloud

4.5 Data Preparation-

Vectorization

Vectorization basically tries to map a word using a dictionary to a vector. We pass only these vectors to our machine learning models. For this purpose, we have tried Tfidf Vectorizer, Bag of Words and Word2Vec on our final dataset.

```
df_final.text = df_final.text.astype(str)
print(np.where(pd.isnull(df_final.text)))
print(len(df_final))
df_final=df_final.dropna(subset=['text'])
print(np.where(pd.isnull(df_final.text)))
print(len(df_final))

(array([], dtype=int64),)
231717
(array([], dtype=int64),)
231717
```

4.6 Model Training

In this phase of implementations, we tried to implement some of the machine learning classification algorithms to train our model. We also considered and implemented ensemble learning techniques mainly Voting classifier and Stacking Classifier.

4.6.1 Implementing base paper models

We have implemented Naive Bayes and Support Vector Machine which were mentioned in our base research paper.

- **Naive Bayes -**

```
# prepare the cross-validation procedure
cv = KFold(n_splits=50, random_state=1, shuffle=True)

model = naive_bayes.MultinomialNB()
# evaluate model
scores = cross_val_score(model, X, Y, scoring='accuracy', cv=cv)

# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores)*100, std(scores)))
```

Snapshot 11: Naïve Bayes Training

The accuracy obtained – **88.066 %**

- **Support Vector Machine -**

```
# prepare the cross-validation procedure
cv = KFold(n_splits=10, random_state=1, shuffle=True)

model = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
# evaluate model
scores = cross_val_score(model, X, Y, scoring='accuracy', cv=cv)

# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores)*100, std(scores)))
```

Snapshot 12: SVM Training

The accuracy obtained - **91.817 %**

4.6.2 Implementing other Machine Learning models

We tried to implement some other machine learning classifiers individually before considering them into our ensemble learning models. Following are those models along with the obtained accuracy.

- **Logistic Regression – 87.06 %**

- Decision Tree - 75.63%
- Support Vector Machine - 91.81%
- K-NN - 78.83%
- Naive Bayes - 88.06%
- Multilayer Perceptron (MLP) - 88.10%

4.6.3 Implementing Ensemble Learning Techniques –

4.6.3.1 Voting Classifier -

We considered the traditional voting classifier ensemble learning technique in which the input is fed to multiple machine learning models and all these models predict their respective outputs and the majority voted output will be considered as final output.

```
from sklearn.ensemble import VotingClassifier

estimators=[("naive_model",naive_model),("log_reg",log_reg),("svm_model",svm_model)]
ensemble = VotingClassifier(estimators, voting="hard")

#fit model to training data
ensemble.fit(Train_X_Tfidf , Train_Y)#test our model on the test data
(ensemble.score(Test_X_Tfidf, Test_Y))*100
```

Snapshot 13: Voting Classifier Training

Here we considered Naive Bayes, Logistic Regression and support vector machine model as a base model to the voting classifier and obtained an accuracy of around **92.67%**

4.6.3.2 Stacking Classifier -

Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs -- meta-features -- of the individual classification models in the ensemble.

```
# Create Base Learners
l_1 = [
    ('rf_1', RandomForestClassifier(n_estimators=10, random_state=42)),
    ('rf_2', KNeighborsClassifier(n_neighbors=4)),
    ('rf_4', DecisionTreeClassifier(max_depth = 2)),
    ('rf_6', MLPClassifier(hidden_layer_sizes=(512,256,128,64,32),\
        activation="relu",random_state=1).fit(X, Train_Y))
]
l_2 = [ ('dt_2', naive_bayes.MultinomialNB()),
        ('rf_5', svm.SVC()) ]

second = StackingClassifier(estimators=l_2, final_estimator= LogisticRegression())
# Create Final model by
clf = StackingClassifier(estimators=l_1, final_estimator=second)
# Extract score
X_train, X_test, y_train, y_test = train_test_split(X, Train_Y, stratify=Y, random_state=42)
clf.fit(X_train, y_train).score(X_test, y_test)
```

Snapshot 14: Stacking Classifier Training

Here in the stacking classifier, we implemented -

1. Random Forest
2. K-NN
3. Decision Tree
4. MLP
5. Logistic Regression

And we implemented a 2-level stacking classifier and finally received accuracy at around **93.05%**

4.7 Program Code

4.7.1 Data Preprocessing and Model Training

```
#!pip install tweepy
import tweepy
import csv
import pandas as pd
from tqdm import tqdm
import numpy as np
import ast
import string
```

```
import datetime

# input your credentials here
consumer_key = 'EcwAltC0oIBlwQlesRg2Ux5A8'
consumer_secret = 'tMltHbelqqDAWJhoTSPa2TDrB4LWL4FK9B40F7qGsU7wUEmN9J'
access_token = '1331612828281683973-9SYgUjN2pYpoTVpONPRh3SCsMsMbB1'
access_token_secret = 'n901ArAC0NHH1EExcCEn8BMApnGfjeJHM6V0Cz3wtS5Mq'

#accessing data using twitter api via tweepy
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)

#a csv file for storing depressive tweets
csvFile = open('tweet-dataset-depressive.csv', 'a')

# Use csv Writer
csvWriter = csv.writer(csvFile)
query = [ 'sad' , 'disappoint' , 'hurtful' , 'upset']

csvWriter.writerow(["id", "timestamp","tweet_text"])
for qu in query:
    print(qu)
    for tweet in tqdm(tweepy.Cursor(api.search,
q=qu,tweet_mode="extended",lang="en",since="2020-04-01").items(5000)):
        if (not tweet.retweeted) and ('RT @' not in tweet.full_text):
            csvWriter.writerow([tweet.id, tweet.created_at,tweet.full_text.encode('utf-8')])

    print("Processing for the word { } finished".format(qu))

#csv file for storing normal tweets
csvFile = open('tweet-dataset-non-depressive.csv', 'a')

# Use csv Writer
csvWriter = csv.writer(csvFile)
# 'happy', 'blissful', 'satisfied', 'delighted' , 'pleased' ,
query = ['joyful', 'cheerful', 'merry', 'glad']

csvWriter.writerow(["id", "timestamp","tweet_text"])
for qu in query:
    print(qu)
    for tweet in tqdm(tweepy.Cursor(api.search, q=qu,tweet_mode="extended",\
lang="en",since="2020-04-01").items(6000)):
        if (not tweet.retweeted) and ('RT @' not in tweet.full_text):
            csvWriter.writerow([tweet.id, tweet.created_at,tweet.full_text.encode('utf-8')])

    print("Processing for { } finished".format(qu))

# reading csv file in df_depressive
df_depressive = pd.read_csv("Depressive_Merged.csv",encoding="utf-8")
```

```
#df_depressive.columns = ["id","time_stamp","tweet_text"]
print(len(df_depressive))
df_depressive.head()

# reading csv file containing non-depressive words in tweets
df_normal = pd.read_csv("Happy_Merged.csv",encoding="utf-8")

df_normal.columns = ["id","time_stamp","tweet_text"]
print(len(df_normal))
df_normal.head()

# if want to drop some rows
df_normal = df_normal.drop([0,1]) -> drops 1st and 2nd rows.
df_normal.head()

# !pip install emoji
import re
import emoji
import ast

# function for replacing urls(starting from http) with an empty string.
def remove_urls(text):
    return re.sub(r'http\S+', "", text)

# function for replacing usernames(starting with @____) with an empty string
def remove_usernames(text):
    return re.sub('@[\^s]+', "", text)

def remove_hashtags(text):
    return re.sub('[!@_%^*#$=-]', "", text)

def remove_emojis(text):
    allchars = [j for j in text]
    emoji_list = [c for c in allchars if c in emoji.UNICODE_EMOJI]
    clean_text = ''.join([k for k in text.split() if not any(i in k for i in emoji_list)])
    return clean_text

def remove_digits(text):
    return re.sub(r'\b\d+\b', "", text)

def remove_digits_str(text):
    return re.sub(r'\d+', "", text)

# Data Cleaning on Dataset containing Depressive words
df_depressive['tweet_text'] = df_depressive['tweet_text'].apply(ast.literal_eval).str.decode("utf-8")

# .apply() is an inbuilt function in pandas
df_depressive.tweet_text = df_depressive.tweet_text.apply(remove_urls)
```

```
df_depressive['tweet_text'].dropna(inplace=True) # drops null string
df_depressive.tweet_text = df_depressive.tweet_text.apply(remove_usernames)
df_depressive.tweet_text = df_depressive.tweet_text.apply(lambda x:".join([i for i in x if i
not in string.punctuation]))
df_depressive.tweet_text = df_depressive.tweet_text.apply(remove_emojis)
df_depressive.tweet_text = df_depressive.tweet_text.apply(remove_hashtags)
df_depressive.tweet_text = df_depressive.tweet_text.apply(remove_digits)
df_depressive.tweet_text = df_depressive.tweet_text.apply(remove_digits_str)
```

```
df_depressive['tweet_text'].dropna(inplace=True) # considering only tweet_text column
df_depressive = df_depressive.drop_duplicates(keep='first') # dropping duplicates (keeping
1st)... when same timestamp , id , tweet
# considering whole table for matching id , time and text
df_depressive.head()
```

```
# Data Cleaning on Dataset containing Non-Depressive words
df_normal['tweet_text'] = df_normal['tweet_text'].apply(ast.literal_eval).str.decode("utf-8")
df_normal.tweet_text = df_normal.tweet_text.apply(remove_urls)
df_normal['tweet_text'].dropna(inplace=True) # drops null string
df_normal.tweet_text = df_normal.tweet_text.apply(remove_usernames)
df_normal.tweet_text = df_normal.tweet_text.apply(lambda x:".join([i for i in x if i not in
string.punctuation]))
```

```
df_normal.tweet_text = df_normal.tweet_text.apply(remove_emojis)
df_normal.tweet_text = df_normal.tweet_text.apply(remove_hashtags)
df_normal.tweet_text = df_normal.tweet_text.apply(remove_digits)
df_normal.tweet_text = df_normal.tweet_text.apply(remove_digits_str)
```

```
df_normal['tweet_text'].dropna(inplace=True)
df_normal = df_normal.drop_duplicates(keep='first')
df_normal.head()
```

```
#form a pair of word and p/n inside word_state_pair list
word_state_pair = []
for i in range(len(word_dictionary)):
    temp = []
    temp.append(word_dictionary.loc[i,"word"])
    temp.append(word_dictionary.loc[i,"p/n"])
    word_state_pair.append(temp)
print(word_state_pair[:3])
```

```
#checking if there are any null strings present in tweet_text columns and if present then
remove them
dep = pd.read_csv("1-output-depressive.csv")
dep['tweet_text'].dropna(inplace=True)
dep = dep.dropna(how='any')
print(np.where(pd.isnull(dep)))
print(np.where(dep.applymap(lambda x: x == "")))
```



```
print(len(dep))
dep.to_csv("1-output-depressive.csv",index=True)

#tagging of tweets present output output-depressive.csv file
dep = pd.read_csv("1-output-depressive.csv")

k = []
data_dict_depressive = {}
some_milby = []

print("length of depressive tweets dataset {}".format(len(dep)))
for i in range(len(dep)):
    k.append(dep.loc[i,"id"])

def add_polarity_depressive_table():
    counter = 0
    ids = []
    texts = []
    pol = []
    time_stamp = []
    for i in tqdm(range(len(dep))):
        tweet_token = dep.loc[i,"tweet_text"]
        time = dep.loc[i,"timestamp"]

        token = word_tokenize(tweet_token)
        sumnum = 0
        sum_word = 0
        for t in token:
            for d in word_state_pair:
                if t == d[0]:
                    sentiment = d[1]
                    if sentiment == "positive":
                        sumnum += 1
                        sum_word += 1
                    elif sentiment == "negative":
                        sumnum += -1
                        sum_word += 1
                    else:
                        sumnum += 0
                        sum_word += 1
                break
        if sum_word != 0.0:
            sum_more = sumnum / sum_word
            if sum_more >= 0.3:
                sum_more = 1
            else:
                sum_more = 0

        varid = k[counter]
        ids.append(varid)
```

```
time_stamp.append(time)
texts.append(tweet_token)
pol.append(sum_more)
counter += 1

data_dict_depressive['ids'] = ids
data_dict_depressive['time'] = time_stamp
data_dict_depressive['tweet_text'] = texts
data_dict_depressive['tag'] = pol

NonDep = pd.read_csv("1-output-normal.csv")
NonDep['tweet_text'].dropna(inplace=True)
NonDep = NonDep.dropna(how='any')
print( np.where(pd.isnull(NonDep)) )
print(np.where(NonDep.applymap(lambda x: x == "")))
print(len(NonDep))
NonDep.to_csv("1-output-normal.csv",index=True)

#tagging of tweets present output output-depressive.csv file
# = pd.read_csv("1-output-normal.csv")

k = []
data_dict_normal = {}
some_milby = []

print("length of normal tweets dataset {}".format(len(NonDep)))
for i in range(len(NonDep)):
    k.append(NonDep.loc[i,"id"])

def add_polarity_normal_table():
    counter = 0
    ids = []
    texts = []
    pol = []
    time_stamp = []
    for i in tqdm(range(len(NonDep))):
        tweet_token = NonDep.loc[i,"tweet_text"]
        time = NonDep.loc[i,"timestamp"]
        token = word_tokenize(tweet_token)
        sumnum = 0
        sum_word = 0
        for t in token:
            for d in word_state_pair:
                if t == d[0]:
                    sentiment = d[1]
                    if sentiment == "positive":
                        sumnum += 1
                        sum_word += 1
                    elif sentiment == "negative":
                        sumnum += -1
```

```
        sum_word += 1
    else:
        sumnum += 0
        sum_word += 1
    break
if sum_word != 0.0:
    sum_more = sumnum / sum_word
    if sum_more >= 0.4:
        sum_more = 1
    elif (sum_more < 0.4) and (sum_more > -0.5):
        sum_more = 0
    elif sum_more <= -0.5:
        sum_more = 0
    else:
        pass

    varid = k[counter]
    ids.append(varid)
    time_stamp.append(time)
    texts.append(tweet_token)
    pol.append(sum_more)
    counter += 1

data_dict_normal['ids'] = ids
data_dict_normal['time'] = time_stamp
data_dict_normal['tweet_text'] = texts
data_dict_normal['tag'] = pol

#checking
print("depressive file length :",len(depressive))
print("normal file length :",len(normal))
print("final dataset file length :",len(depressive)+len(normal))
print("final dataset after combined length :",len(final_dataset))

df_tweet_tag = df_final[['tweet_text','tag']]
df_tweet_tag.tweet_text = df_tweet_tag.tweet_text.astype(str)
print(np.where(pd.isnull(df_tweet_tag.tweet_text)))
print(len(df_tweet_tag))
df_tweet_tag=df_tweet_tag.dropna(subset=['tweet_text'])
print(np.where(pd.isnull(df_tweet_tag.tweet_text)))
print(len(df_tweet_tag))

# Remove Stop words, Non-Numeric and perform Word Stemming/Lemmenting.#
WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective
etc. By default it is set to Noun
tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
```

```
for index,entry in enumerate(df_tweet_tag['tweet_text'] ):

    # Declaring Empty List to store the words that follow the rules for this step
    Final_words = []

    # Initializing WordNetLemmatizer()
    word_Lemmatized = WordNetLemmatizer()

    # pos_tag function below will provide the 'tag' i.e if the word is Noun(N) or Verb(V) or
    something else.
    for word, tag in pos_tag(entry):

        # Below condition is to check for Stop words and consider only alphabets
        if word not in stopwords.words('english') and word.isalpha():
            word_Final = word_Lemmatized.lemmatize(word,tag_map[tag[0]])
            Final_words.append(word_Final)

    # The final processed set of words for each iteration will be stored in 'text_final'
    df_tweet_tag.loc[index,'text_final'] = str(Final_words)

# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)# Use accuracy_score function to get the
accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)

# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
Model = SVM.fit(Train_X_Tfidf,Train_Y)# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)# Use accuracy_score function to get the
accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)

from sklearn.ensemble import VotingClassifier#create a dictionary of our models

# estimators=[("knn", knn), ("naive_model", naive_model), ("log_reg", log_reg) ,
("svm_model" , svm_model)]#create our voting classifier, inputting our models

estimators=[ ("naive_model", naive_model), ("log_reg", log_reg) , ("svm_model" ,
svm_model)]#create our voting classifier, inputting our models

ensemble = VotingClassifier(estimators, voting="hard")

# Create Base Learners
l_1 = [
    ('rf_1', RandomForestClassifier(n_estimators=10, random_state=42)),
    ('rf_2', KNeighborsClassifier(n_neighbors=4)),
    ('rf_4', DecisionTreeClassifier(max_depth = 2)),
```

```
(rf_6', MLPClassifier(hidden_layer_sizes=(512,256,128,64,32),\
                        activation="relu",random_state=1).fit(X, Train_Y))
    ]
l_2 = [ ('dt_2', naive_bayes.MultinomialNB()),
        ('rf_5', svm.SVC()) ]

second = StackingClassifier(estimators=l_2, final_estimator= LogisticRegression())
# Create Final model by
clf = StackingClassifier(estimators=l_1, final_estimator=second)
# Extract score
X_train, X_test, y_train, y_test = train_test_split(X, Train_Y, stratify=Y, random_state=42)
clf.fit(X_train, y_train).score(X_test, y_test)
```

4.7.2 Web Deployment

```
# ! pip install tweepy

import tweepy

import csv

import pandas as pd

from tqdm import tqdm

import numpy as np

import ast

import string

import datetime


# input your credentials here

consumer_key = 'EcwAltC0olBlwQlesRg2Ux5A8'

consumer_secret = 'tMltHbelqqDAWJhoTSPa2TDrB4LWL4FK9B40F7qGsU7wUEmN9J'

access_token = '1331612828281683973-9SYgUjN2pYpoTVpONPRh3SCsMsMbB1'

access_token_secret = 'n901ArAC0NHH1EEExcCEn8BMApnGfjejHM6V0Cz3wtS5Mq'


#accessing data using twitter api via tweepy

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)

auth.set_access_token(access_token, access_token_secret)
```

```
api = tweepy.API(auth, wait_on_rate_limit=True)

def get_tweets(userID):
    user_exist = False
    try:
        tweets = api.user_timeline(screen_name=userID,
                                    # 200 is the maximum allowed count
                                    count=200,
                                    include_rts = False,
                                    # Necessary to keep full_text
                                    # otherwise only the first 140 words are extracted
                                    tweet_mode = 'extended'
                                    )

        user_exist = True
        raw_tweets = []
        for tweet in tweets:
            raw_tweets.append(tweet.full_text)

        tweet_details = []
        for tweet in tweets:
            tweet_details.append([tweet.created_at, tweet.full_text])

        return raw_tweets, tweet_details, user_exist

    except Exception:

        return list(), list(), user_exist
```

```
def predict_result(tweets_list):  
    df = pd.DataFrame({'tweets':tweets_list})  
  
    df = df.astype({'tweets':str})  
    X = df['tweets']  
    print("X :")  
    print(X)  
    transformer = TfidfTransformer()  
  
    # countvectorizer saved model  
    root = "saved_model/"  
    c_name = root + "Feature.pkl"  
    loaded_vec = CountVectorizer(decode_error="replace",lowercase=False,vocabulary=pickle.load(open(c_name, "rb")))  
    tfidf = transformer.fit_transform(loaded_vec.fit_transform(X))  
  
    # model name  
    filename = root + "Pickle_Naive.pkl"  
    loaded_model = joblib.load(open(filename, 'rb'))  
  
    # result prediction  
    result = loaded_model.predict(tfidf)  
    u, counts = np.unique(result, return_counts=True)  
    dictionary = dict(zip(u, counts))  
    print(dictionary)  
    # percentage calculation  
    print(result)  
    percentage = 0  
    if 1 in dictionary:
```

```
    one_count = dictionary[1]

    percentage = 0

    if 0 in dictionary:

        zero_count = dictionary[0]

        percentage = (zero_count / len(result))*100

    return percentage

def predict_singletext_result(normal_text):

    df = pd.DataFrame({'tweets':normal_text})

    df = df.astype({'tweets':str})
    X = df['tweets']

    transformer = TfidfTransformer()

    # countvectorizer saved model
    root = "saved_model/"
    c_name = root + "Feature.pkl"

    loaded_vec = CountVectorizer(decode_error="replace",lowercase=False,vocabulary=pickle.load(open(c_name, "rb")))

    tfidf = transformer.fit_transform(loaded_vec.fit_transform(X))

    # model name
    filename = root + "Pickle_Naive.pkl"
    loaded_model = joblib.load(open(filename, 'rb'))

    # result prediction
    result = loaded_model.predict(tfidf)
```



```
if result == 0:
    normal_text_result = '0'
else:
    normal_text_result = '1'
```

```
return normal_text_result
```

- Run.py

```
'''
```

```
code for webapp
```

```
'''
```

```
import os
from uuid import uuid4
from flask import Flask
from flask import request
from flask import render_template
from flask import send_from_directory
import flask
# import cv2
from get_recent_tweets import get_tweets
from preprocessing_tweets import text_preprocessing
from prediction import predict_result
from prediction import predict_singletext_result

app = Flask(__name__)
root_dir = os.path.dirname(os.path.abspath(__file__))

#index.html file
@app.route("/")
def index():
```

```
return render_template("index.html")

#colour identification from uploaded file

@app.route("/checking_username", methods=["POST","GET"])
def search_username():
    if flask.request.method == 'POST':
        username = request.form['username']
        print(username)
        temp = 0
        if username != "":
            tweets_list,tweet_details,user_exist = get_tweets(username)
            if len(tweets_list) >=1:
                preprocess_tweets = text_preprocessing(tweets_list)
                temp = int(predict_result(preprocess_tweets))
                percentage = str(round(predict_result(preprocess_tweets),2))
                tweets_list = tweets_list[:5]
                tweet_details = tweet_details[:5]
            else:
                percentage = 0

            dep = False

            if temp > 50 :
                dep = True
            else:
                dep = False

            return render_template("index.html", dep = dep, username = username,user_exis
t=user_exist, tweets_list = tweets_list,tweet_details = tweet_details,percentage=percentage
)
```

```
        else:

            return render_template("error.html")

@app.route("/checking_text", methods=["POST", "GET"])
def search_text():

    if flask.request.method == 'POST':

        normal_text = request.form['normal_text']

        print(normal_text)

        if normal_text != "":

            preprocess_tweets = text_preprocessing([normal_text])

            normal_text_result = predict_singletext_result(preprocess_tweets)

            normal_text_result = [normal_text_result]

            return render_template("index.html", normal_text = normal_text, normal_text_resu
lt = normal_text_result)

        else:

            return render_template("error.html")

@app.route('/upload/<filename>')
def send_image(filename):

    return send_from_directory("images", filename)

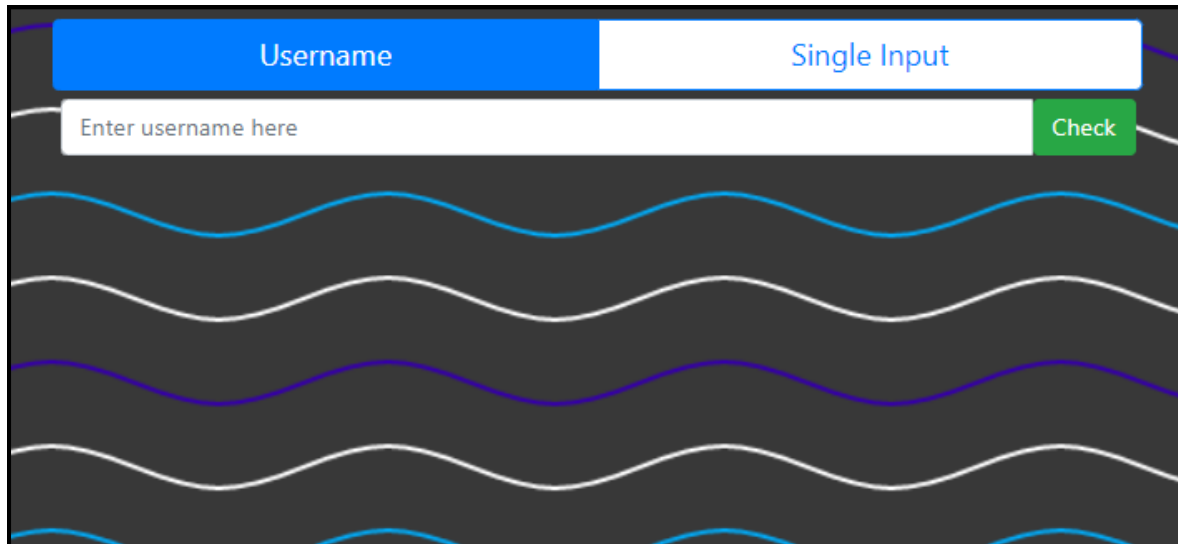
@app.route('/error')
def error():

    return render_template("error.html")

if __name__ == "__main__":

    app.run(debug=True)
```

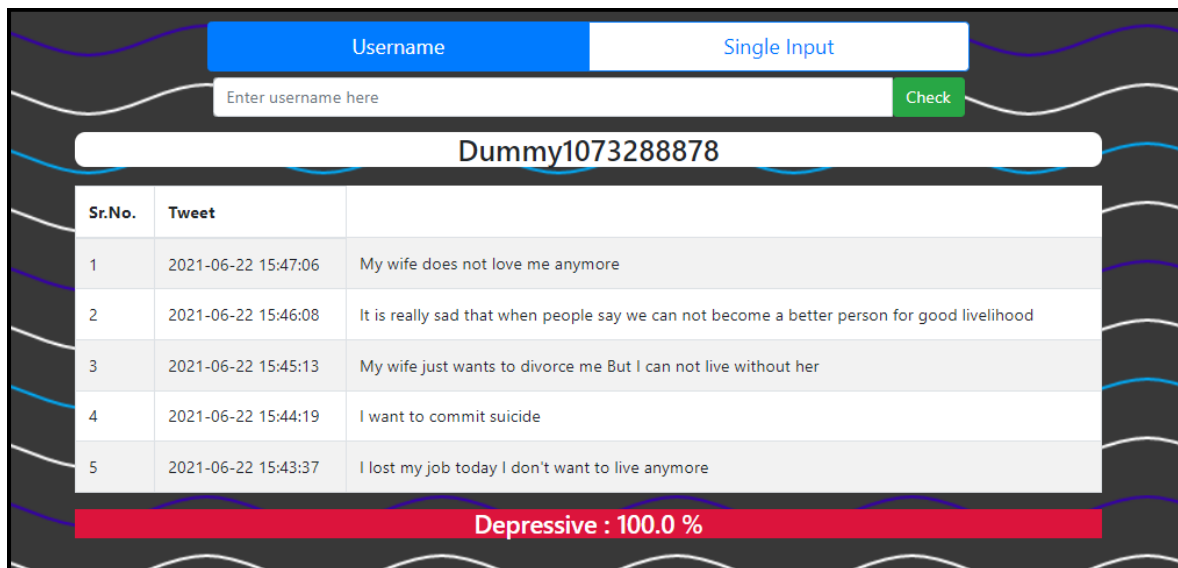
4.8 Web application snapshots-



Username Single Input

Enter username here Check

Snapshot 15: Web application GUI



Username Single Input

Enter username here Check

Dummy1073288878

Sr.No.	Tweet
1	2021-06-22 15:47:06 My wife does not love me anymore
2	2021-06-22 15:46:08 It is really sad that when people say we can not become a better person for good livelihood
3	2021-06-22 15:45:13 My wife just wants to divorce me But I can not live without her
4	2021-06-22 15:44:19 I want to commit suicide
5	2021-06-22 15:43:37 I lost my job today I don't want to live anymore

Depressive : 100.0 %

Snapshot 16: Twitter user prediction: Depressive

Sr.No.	Tweet
1	2021-07-06 15:55:46 @htmleverything Thank you Mike, you're awesome 🥰
2	2021-07-06 13:49:12 What? https://t.co/96uQ1piqFp
3	2021-07-06 11:02:36 @Prathkum @hrdk_codes https://t.co/m42oGuGdxZ
4	2021-07-06 00:17:34 @htmleverything I was supposed to post a picture of my birthday cake, but Twitter was like noooope 🤔
5	2021-07-05 07:57:29 Thank you for all the wishes, for some reason I've been very excited about this year's birthday than any other 🥰 I'll take sometime off from this bird app to enjoy with my friends and family.

Depressive : 46.86 %

Snapshot 17: Twitter user prediction: Non-Depressive

Text

I feel good today, I love my life

Non-Depressive

Snapshot 18: prediction of single text: Non-Depressive

Text

I feel sad

Depressive

Snapshot 19: prediction of single text: Depressive

5. Conclusion

Conclusion

5.1 Conclusion -

This Project defines the binary classification problem which identifies whether the person is depressed, based on his tweet's activity. Different machine learning algorithms are experimented and also different data splitting are applied Different pre-processing steps are applied or performed which includes data preparation, data cleaning, data labelling, feature extraction etc. Other than Individual classifiers the blending ensemble classifier is giving more better accuracy and results. Finally, we deployed the system on a web application where we can simply check whether the user is going through depression or not by simply typing his/her twitter user id in the search bar. We can also check a single text input and check whether that input sentence represents depression or not.

5.2 Future Scope -

This study can be extended in the future work by considering more factors like which tweets user like most, following, different retweets by user etc. these features will help to obtain better and more accurate prediction.

Images and videos can also be considered for the same purpose in the future scope. By analysing images and videos along with the text can give us more accurate and proper prediction and that will help us to make a better depression detection system.

We can extend the same system to other social media platforms like Facebook, reddit etc. we can simply make a universal system that will simply consider all the factors from all the available social media platforms to detect if a person needs help or not by check whether the person is going through depression or not.

Project Time Frame**July 2020 - July 2021**

Activity	JUL	AUG	SEPT	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL
Literature Survey	←→												
Research Gaps		←→											
Data Collection			←→										
Algorithm Selection			←→										
Experimental Execution					←→								
Experimental Analysis						←→							
Results and Analysis						←→							
Dissertation Writing										←→			
Dissertation Submission												←→	

Fig. 11 Time frame

References

References

- [1] A. Noureen, U. Qamar and M. Ali, "Semantic analysis of social media and associated psychotic behavior," 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, 2017, pp. 1621-1630, doi: 10.1109/FSKD.2017.8393009.
- [2] P. Arora and P. Arora, "Mining Twitter Data for Depression Detection," 2019 *International Conference on Signal Processing and Communication (ICSC)*, NOIDA, India, 2019, pp. 186-189, doi: 10.1109/ICSC45622.2019.8938353.
- [3] M. Deshpande and V. Rao, "Depression detection using emotion artificial intelligence," 2017 *International Conference on Intelligent Sustainable Systems (ICISS)*, Palladam, 2017, pp. 858-862, doi: 10.1109/ISS1.2017.8389299.
- [4] P. Gupta and B. Kaushik, "Suicidal Tendency on Social Media: A Case Study," 2019 *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Faridabad, India, 2019, pp. 273-276, doi: 10.1109/COMITCon.2019.8862236.
- [5] Guangyao Shen, Jia Jia, Liqiang Nie, Fuli Feng, Cunjun Zhang, Tianrui Hu, Tat-Seng Chua, & Wenwu Zhu (2017). Depression Detection via Harvesting Social Media: A Multimodal Dictionary Learning Solution. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (pp. 3838–3844).
- [6] N. A. Asad, M. A. Mahmud Pranto, S. Afreen, and M. M. Islam, "Depression Detection by Analyzing Social Media Posts of User," 2019 *IEEE International Conference on Signal Processing, Information, Communication & Systems (SPICSCON)*, Dhaka, Bangladesh, 2019, pp. 13-17, doi: 10.1109/SPICSCON48833.2019.9065101.
- [7] S. Jain, S. P. Narayan, R. K. Dewang, U. Bhartiya, N. Meena, and V. Kumar, "A Machine Learning based Depression Analysis and Suicidal Ideation Detection System using Questionnaires and Twitter," 2019 *IEEE Students Conference on Engineering and Systems (SCES)*, Allahabad, India, 2019, pp. 1-6, doi: 10.1109/SCES46477.2019.8977211.

- [8] A. U. Hassan, J. Hussain, M. Hussain, M. Sadiq, and S. Lee, "Sentiment analysis of social networking sites (SNS) data using machine learning approach for the measurement of depression," *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, 2017, pp. 138-140, doi: 10.1109/ICTC.2017.8190959.
- [9] P. V. Narayanrao and P. Lalitha Surya Kumari, "Analysis of Machine Learning Algorithms for Predicting Depression," *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, Gunupur, India, 2020, pp. 1-4, doi: 10.1109/ICCSEA49143.2020.9132963.
- [10] C. Zucco, B. Calabrese and M. Cannataro, "Sentiment analysis and affective computing for depression monitoring," *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Kansas City, MO, 2017, pp. 1988-1995, doi: 10.1109/BIBM.2017.8217966.
- [11] O. Obulesu, M. Mahendra and M. ThrilokReddy, "Machine Learning Techniques and Tools: A Survey," *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, 2018, pp. 605-611, doi: 10.1109/ICIRCA.2018.8597302.
- [12] T. N. Rincy and R. Gupta, "Ensemble Learning Techniques and its Efficiency in Machine Learning: A Survey," *2nd International Conference on Data, Engineering and Applications (IDEA)*, Bhopal, India, 2020, pp. 1-6, doi: 10.1109/IDEA49133.2020.9170675.
- [13] Dong, X., Yu, Z., Cao, W. *et al.* A survey on ensemble learning. *Front. Comput. Sci.* **14**, 241–258 (2020). <https://doi.org/10.1007/s11704-019-8208-z>
- [14] <https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/>
- [15] <https://www.who.int/news-room/fact-sheets/detail/depression>

Appendix: Research Paper Published

List of Published Paper

1. Suyash Dabhane and P. M. Chawan, “Depression Detection on Social Media using Machine Learning Technique – A survey” International Research Journal of Engineering and Technology (IRJET), Volume 07, Issue 11, Nov 2020
2. Suyash Dabhane and P. M. Chawan, “Depression Detection on Social Media using Machine Learning Technique” International Journal for Scientific Research and Development (IJSRD), Volume 09, Issue 04, June 2021

