

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590018

2020-2021



A

Project Phase-2 Report

On

“DETECTION OF DEPRESSION IN TEXT SEQUENCES ”

Submitted in the partial fulfillment of the requirement for the VIII Semester

Project phase-2 Work-17CSP85 for the award of degree of

Bachelor of Engineering

In

“Computer Science and Engineering”

By

SAVITHA SHREE M

1GV17CS064

INFANCIA R

1GV17CS077

SANDRA CAROLIN S

1GV17CS078

MARIA REBECCA D

1GV17CS079

Under the Guidance of

Mrs. SOPHIA S,

Assistant Professor,

Dept. of CSE, Dr.TTIT, KGF



Dr. T. THIMMAIAH INSTITUTE OF TECHNOLOGY

(Formerly Golden Valley Institute of Technology)

Department of Computer Science and Engineering

Kolar Gold Fields – 563120

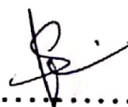


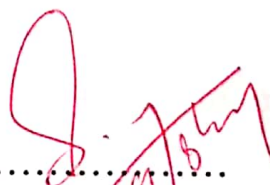
(Formerly Golden Valley Institute of Technology)
Oorgaum Kolar Gold Fields – 563120

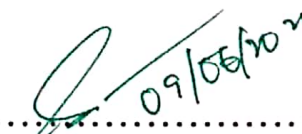
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Project work entitled **“DETECTION OF DEPRESSION IN TEXT SEQUENCES”** is a bonafide work carried out by **SAVITHA SHREE M - 1GV17CS064, INFANCIA R – 1GV17CS077, SANDRA CAROLIN S - 1GV17CS078 and MARIA REBECCA D – 1GV17CS079** in the partial fulfillment for the award of degree Bachelor of Engineering in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi during the academic year 2020-2021.** It is certified that all corrections/suggestions indicated for the assessment have been incorporated in the report deposited in the department library. The Project Phase-2 report has been approved as it satisfies the academic requirement in respect of **Project Phase-2 work 17CSP85** prescribed for the Bachelor of Engineering Degree.


.....
Signature of guide
Mrs. SOPHIA S
(Asst. Prof., Dept. of CSE)


.....
Signature of HOD
Dr. S SREEDHAR KUMAR
(HOD, Dept. of CSE)


.....
Signature of Principal
Dr. SYED ARIFF



ACKNOWLEDGEMENT

It is with the profound feeling of gratitude we would like to express our sincere thanks to our institution **Dr.T. THIMMAIAH INSTITUTE OF TECHNOLOGY, K.G.F** for providing excellent infrastructure for the successful completion of project.

We wish to express a wholehearted thanks to our **principal Dr. Syed Ariff** for his kind support in carrying out this project work.

We would like to extend hearty thanks to our **HOD Dr. Sreedhar Kumar S , Dep. of CSE** for being a constant support of encouragement to carry out the project successfully.

We would like to extend hearty thanks to our guide **Mrs. Sophia S, Assistant Professor, CSE** for her valuable suggestions, guidance and support in the completion of this project.

We would also like to thank project Co-Ordinator **Mrs. Shalini S, Assistant Professor**, for her timely support in the completion of this project.

We would also like to thank all teaching and non-teaching staff who were directly and indirectly supported for carrying out this project successfully.

We extend our hearty thanks to our parents, friends for all the moral support provided during the preparation for the project.

SAVITHA SHREE M	(1GV17CS064)
INFANCIA R	(1GV17CS077)
SANDRA CAROLIN S	(1GV17CS078)
MARIA REBECCA D	(1GV17CS079)

ABSTRACT

Depression is ranked as the largest contributor to global disability and is also a major reason for suicide. Still, many individuals suffering from forms of depression are not treated for various reasons. Previous studies have shown that depression also has an effect on language usage and that many depressed individuals use social media platforms or the internet in general to get information or discuss their problems.

This paper addresses the early detection of depression using machine learning models based on messages on a social platform. In particular, a convolutional neural network based on different word embeddings is evaluated and compared to a classification based on user-level linguistic metadata. An ensemble of both approaches is shown to achieve state-of-the-art results in a current early detection task.

Furthermore, the currently popular ERDE score as metric for early detection systems is examined in detail and its drawbacks in the context of shared tasks are illustrated. A slightly modified metric is proposed and compared to the original score. Finally, a new word embedding was trained on a large corpus of the same domain as the described task and is evaluated as well.

CONTENTS

DETAILS	PAGE NO
CHAPTER 1 INTRODUCTION	1-4
1.1 History	2
1.2 Scope of project	3
1.3 Objective	3-4
CHAPTER 2 LITERATURE SURVEY	5-7
CHAPTER 3 SYSTEM REQUIREMENTS SPECIFICATION	8-10
3.1 Functional Requirement	8
3.2 Non functional Requirement	8
3.3 Resource Requirement	9-10
CHAPTER 4 SYSTEM ANALYSIS	11
4.1 Feasibility Analysis	11
4.2 Proposed System	11
CHAPTER 5 SYSTEM DESIGN	12-17
5.1 Methodology	12
5.2 System Architecture	12-13
5.3 Classes designed	14
CHAPTER 6 IMPLEMENTATION	18-57
6.1 Implementation Requirements	18
6.2 Coding Guideline	20
6.3 Pseudo Code	21
6.4 Source Code	22-58

CHAPTER 7	TESTING	59-60
	7.1 Dataset	59
	7.2 Test procedure	59
	7.3 Unit testing	59
	7.4 Integrated testing	60
	7.5 Summary	60
CHAPTER 8	RESULT ANALYSIS	61-65
	8.1 Snapshots	61
REFERENCES		66-67

LIST OF FIGURES

FIG.NO	NAME OF THE FIGURE	PAGE NO.
Fig 5.2	System Architecture	12
Fig 5.3	Class Diagram	14
Fig 5.4	Use Case Diagram	15
Fig 5.5	Data Flow Diagram	16
Fig 5.5.1	Level 0	16
Fig 5.5.2	Level 1	17
Fig 5.5.3	Classification process	17

CHAPTER 1

INTRODUCTION

According to World Health Organization (WHO), more than 300 million people worldwide are suffering from depression, which equals about 4.4% of the global population. While forms of depression are more common among females (5.1%) than males (3.6%) and prevalence differs between regions of the world, it occurs in any age group and is not limited to any specific life situation. Latest results from the 2016 National Survey on Drug Use and Health in the United States report that, during the year 2016, 12.8% of adolescents between 12 and 17 years old and 6.7% of adults had suffered a major depressive episode (MDE).

Researchers found out that shame and self-stigmatization seem to be much stronger reasons to not seek psychiatric help than actual perceived stigma and negative reactions of others. While depression and other mental illnesses may lead to social withdrawal and isolation, it was found that social media platforms are indeed increasingly used by affected individuals to connect with others, share experiences, and support each other. Based on these findings, peer-to-peer communities on social media can be able to challenge stigma, increase the likelihood to seek professional help, and directly offer help online to people with mental illness. All this emphasizes the importance of research toward ways to assist depressed individuals on social media platforms and on the internet in general.

It is therefore focused on ways to classify indications of depression in written texts as early as possible based on machine learning methods. The work presented in this paper is structured as follows: Section 2 gives an overview of related work concerning depression, its influence on language, and natural language processing methods. Section 3 describes the dataset used in this work, analyzes the evaluation metric of the corresponding task, and proposes an alternative. Section 4 introduces the userbased metadata features used for classification, while Section 5 describes the neural network models utilized for this task. Section 6 contains an experimental evaluation of these models and compares them to published results. Finally, Section 7 concludes this work and summarizes the results.

1.1 HISTORY

At the worst, depression can lead to suicide. WHO estimates that, in the year 2015, 788,000 people have died by suicide and that it was the second most common cause of death for people between 15 and 29 years old worldwide.

While depression and other mental illnesses may lead to social withdrawal and isolation, it was found that social media platforms are indeed increasingly used by affected individuals to connect with others, share experiences, and support each other. Based on these findings, peer-to-peer communities on social media can be able to challenge stigma, increase the likelihood to seek professional help, and directly offer help online to people with mental illness.

Internet users with stigmatized illnesses like depression or urinary incontinence are more likely to use online resources for health-related information and for communication about their illness than people with another chronic illness. All this emphasizes the importance of research toward ways to assist depressed individuals on social media platforms and on the internet in general.

This project is therefore focused on ways to classify indications of depression in written texts as early as possible based on machine learning methods.

1.2 SCOPE OF PROJECT

Internet users with stigmatized illnesses like depression or urinary incontinence are more likely to use online resources for health-related information and for communication about their illness than people with another chronic illness. All this emphasizes the importance of research toward ways to assist depressed individuals on social media platforms and on the internet in general.

This project is therefore focused on ways to classify indications of depression in written texts as early as possible based on machine learning methods.

CHAPTER 2

LITERATURE SURVEY

Torii et.al[1] have presented a technique called "An exploratory study of a text classification framework for Internet-based surveillance of emerging epidemics" Early detection of infectious disease outbreaks is crucial to protecting the public health of a society. Online news articles provide timely information on disease outbreaks worldwide. In this study, we investigated automated detection of articles relevant to disease outbreaks using machine learning classifiers.

Schomerus et.al[2] have presented a technique called "The stigma of psychiatric treatment and help-seeking intentions for depression" The stigma of mental illness has often been considered a potential cause for reluctant willingness to seek help for mental problems, but there is little evidence on this issue. We examine two aspects of stigma related to seeing a psychiatrist and their association with help-seeking intentions for depression: anticipated discrimination by others when seeking help and desire for social distance from those seeking help.

Gowen et.al[3] have presented a technique called "Young adults with mental health conditions and social networking websites: Seeking tools to build community" This study examined ways that young adults with mental illnesses (1) currently use social networking; and (2) how they would like to use a social networking site tailored for them. The authors examined differences between those with mental health conditions and those without. Methods: An online survey was administered by the National Alliance on Mental Illness (NAMI) to 274 participants; of those, 207 reported being between 18 and 24 years old.

Rani et.al [4] have presented a technique called "An empirical study of machine learning techniques for affect recognition in human-robot interaction" implicit communication in human interactions, it would be valuable to have this capability in robotic systems wherein a robot can detect the motivations and emotions of

the person it is working with. Recognizing affective states from physiological cues is an effective way of implementing implicit human-robot interaction. Paykel et.al[5] have presented a technique called "Basic concepts of depression" This paper reviews concepts of depression, including history and classification. The original broad concept of melancholia included all forms of quiet insanity. The term depression began to appear in the nineteenth century as did the modern concept of affective disorders, with the core disturbance now viewed as one of mood. The 1930s saw the introduction of defined criteria into official diagnostic schemes.

Naslund et.al [6] have presented a technique called "The future of mental health care: peer-to-peer support and social media" People with serious mental illness are increasingly turning to popular social media, including Facebook, Twitter or YouTube, to share their illness experiences or seek advice from others with similar health conditions. This emerging form of unsolicited communication among self-forming online communities of patients and individuals with diverse health concerns is referred to as peer-to-peer support. We offer a perspective on how online peer-to-peer connections among people with serious mental illness could advance efforts to promote mental and physical wellbeing in this group.

Porges et.al [7] have presented a technique called "Emotion recognition in children with autism spectrum disorders: relations to eye gaze and autonomic state" Respiratory Sinus Arrhythmia (RSA), heart rate, and accuracy and latency of emotion recognition were evaluated in children with autism spectrum disorders (ASD) and typically developing children while viewing videos of faces slowly transitioning from a neutral expression to one of six basic emotions (e.g., anger, disgust, fear, happiness, sadness, and surprise). Children with ASD were slower in emotion recognition and selectively made more errors in detecting anger.

Kim et.al[8] have presented a technique called "Emotion recognition system using short-term monitoring of physiological signals" A physiological signal-based emotion recognition system is reported. The system was developed to operate as a user-independent system, based on physiological signal databases obtained from multiple

subjects. The input signals were electrocardiogram, skin temperature variation and electrodermal activity, all of which were acquired without much discomfort from the body surface, and can reflect the influence of emotion on the autonomic nervous system. The system consisted of preprocessing, feature extraction and pattern classification stages.

Collignon et.al[9] have presented a technique called "Audio-visual integration of emotion expression" Regardless of the fact that emotions are usually recognized by combining facial and vocal expressions, the multisensory nature of affect perception has scarcely been investigated. In the present study, we show results of three experiments on multisensory perception of emotions using newly validated sets of dynamic visual and non-linguistic vocal clips of affect expressions.

Berger et.al[10] have presented a technique called "Internet use and stigmatized illness" People with stigmatized illnesses often avoid seeking health care and education. The internet may be a useful health education and outreach tool for this group. This study examined patterns of internet use for health information among those with and without stigmatized illnesses.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

Software requirement specification captures all the functionalities to be implemented in this project.

3.1 FUNCTIONAL REQUIREMENT

The functionalities to be implemented are

1. Convert Text sentences to word embedding vector
2. Train a Convolutional Neural network to classify the word embedding vector to depression level using the CLEF 2017 Dataset
3. Collect any text input document from the user and classify the depression level for it
4. Measure the performance of the proposed depression level prediction system using CLEF 2017 dataset.

3.2 NON-FUNCTIONAL REQUIREMENT

The non functional comes under following category,

1. Product Requirements
2. Organizational Requirements
3. User Requirements
4. Basic Operational Requirements.

3.2.1 Product Requirements:

Portability:The project can work on any platform

Accuracy:The classification accuracy must be higher.

Convenience:It should easy for the users to use this software.

3.2.2 Organizational Requirements:

Process Standards: IEEE models are utilized to build up the application which is the standard utilized by the greater part of the standard programming designers everywhere throughout the world.

Design Methods:Modular Design approach is used to design the project.

3.2.3 User Requirements:

The user must be able to view the training and classification results.

The GUI must be easy to use and informative.

3.2.4 Basic Operational Requirements:

Mission profile or situation: The mission of the project is design early depression diagnosis system from social media messages.

3.3 RESOURCE REQUIREMENT

Hardware Requirements:

Processors	:	Intel I3 2.2 GHZ
RAM	:	4 GB.
Storage	:	100 GB.
Monitor	:	15"

Software Requirements:

Coding : Python

Platform : Python 3.6

Tool : IDLE

CHAPTER 4

SYSTEM ANALYSIS

Feasibility of the project is different perspective is presented in this section

4.1 FEASIBILITY ANALYSIS

4.1.1 Technical feasibility:

The project will be implemented in Python since it has all necessary modules needed for implementing the project.

4.1.2 Cost feasibility:

The project uses only open source tools for implementation and the dataset are also downloaded free of cost. So cost wise the project is feasible.

4.1.3 Resource feasibility:

The project code is estimated to be around 1000 lines and necessary python expertise needed for project is available. The project can be completed within 3 months with available resources. So there is no risk in implementation of the project.

4.2 PROPOSED SYSTEM

The project combines natural language processing and machine learning to detect depression level from text messages.

Natural language processing is used to create a word embedding model which can convert text sentences to feature vectors.

Machine learning using Neural Networks is used to classify the features vector of sentences to depression level.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM DEVELOPMENT METHODOLOGY

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

5.2 SYSTEM ARCHITECTURE

The System architecture is shown below.

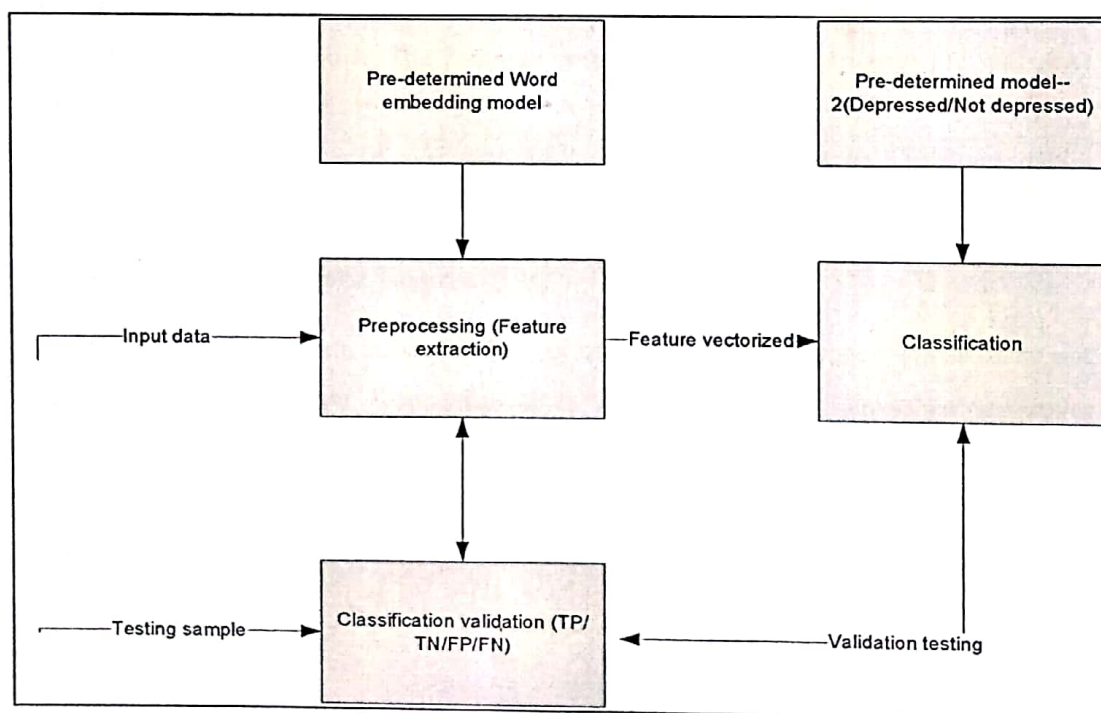


fig 5.2 : System Architecture

Following are the modules in the system

Pre-determined word embedding model : This model is constructed from the training dataset. The training data set will be form of sentence vs emotion conveyed. This training data set is converted to Term Factor notation for the each of the sentence in the training dataset.

Pre-processing: This module extracts the features from the input text document and converts them to a feature vector based on Pre-determined word embedding model for converting text to feature vector. The feature vector is in form of Term Factor notation (TF). Each term in the Term factor notation is an noun or adjective. The occurrence frequency of each term is calculated and given as feature vectors.

Classification: Classification module trains a convolutional neural network based on labelled model of feature vector against depression status. The convolutional neural network classifies the input feature vector to depression or no depression. User can give any sentence. This sentence will be converted to feature vector and given to trained convolutional neural network. The convolutional neural network will give the output as depressed or not depressed.

Classification validation: This module checks the performance of the proposed classification model using standard metrics of True Positive (TP). False positive (FP), True Negative (TN) and False Negative (FN).

The validation will be done against a test set of sentences who label (depressed or not depressed is already know). Each sentence in the test set will be converted to feature vector and given to trained convolutional neural network. Convolutional neural network give the actual result. This actual result is compared with expected result in the label to calculate TP, FP, TN , FN. The counts of TP, FP, TN, FN are calculated as follows

Table 5.2.1

Expected / Actual result	Depressed	Not Depressed
Depressed	Increment TP	Increment TN
Not Depressed	Increment FP	Increment FN

5.3 CLASSES DESIGNED FOR THE SYSTEM

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

The class diagram is shown below.

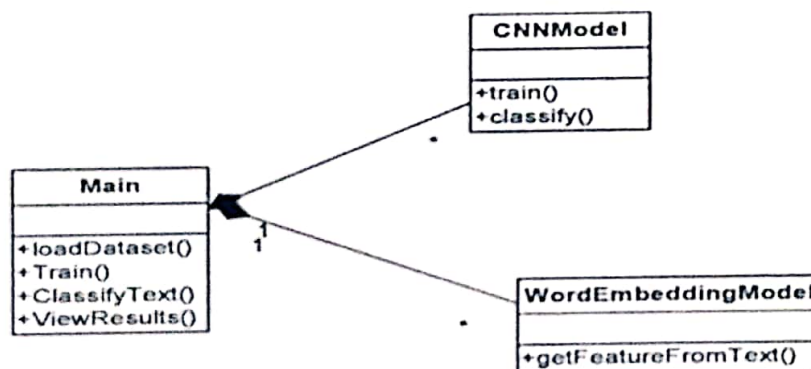


Fig 5.3 : Class Diagram

There are three classes

Main is the user interaction class. It interacts with all other classes to implement the functionality

CNNModel class abstracts the functionality of Convolutional neural network for training and classification

WordEmbeddingModel class abstracts the functionality of converting the text sentences to WordEmbedding features.

5.4 USE CASE DIAGRAM OF THE SYSTEM

A use case diagram is a type of behavioral diagram created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

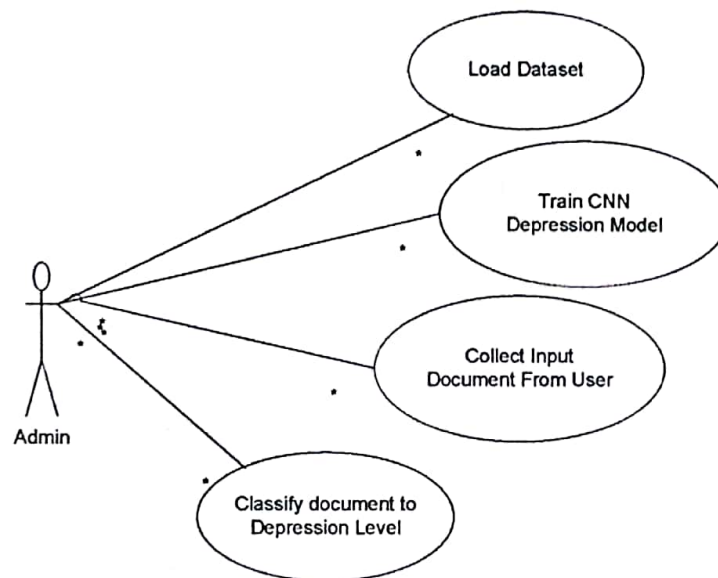


Fig 5.4 : Use Case Diagram

Admin is the user and execute following use cases.

Load dataset: Admin loads the training dataset of sentences vs the depression levels.

Train CNN depression model: Admin train a CNN model for taking the sentence as input and providing the depression level as output.

Collect Input document from user: Admin can load a text document from which depression level must be identified.

Classify document to depression level: After loading the text document, admin can find the depression level by analyzing the document.

5.5 DATA FLOW DIAGRAM OF THE SYSTEM

The input, output and the process flow in the system is given in this section.

Level 0 Data flow diagram

Training and Classification are the two process

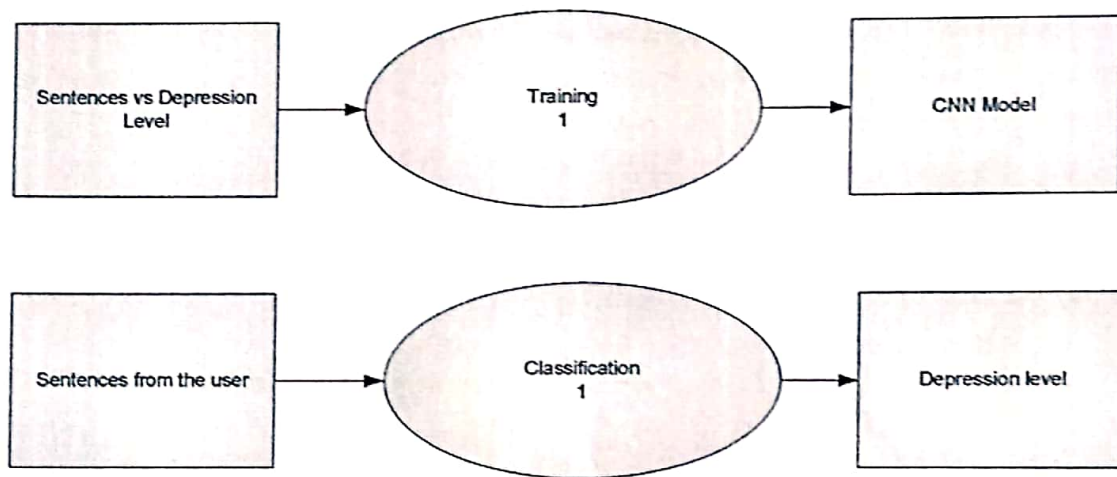


Fig 5.5.1 : Level 0 Data flow Diagram

Training process takes the sentences vs Depression level from the CLEF 2017 dataset , and creates a CNN(Convolutional neural network).

Classification process takes the Sentences from the user as input and provides the depression level as output.

Level 1 Data flow diagram

Training process is split into sub process as shown below

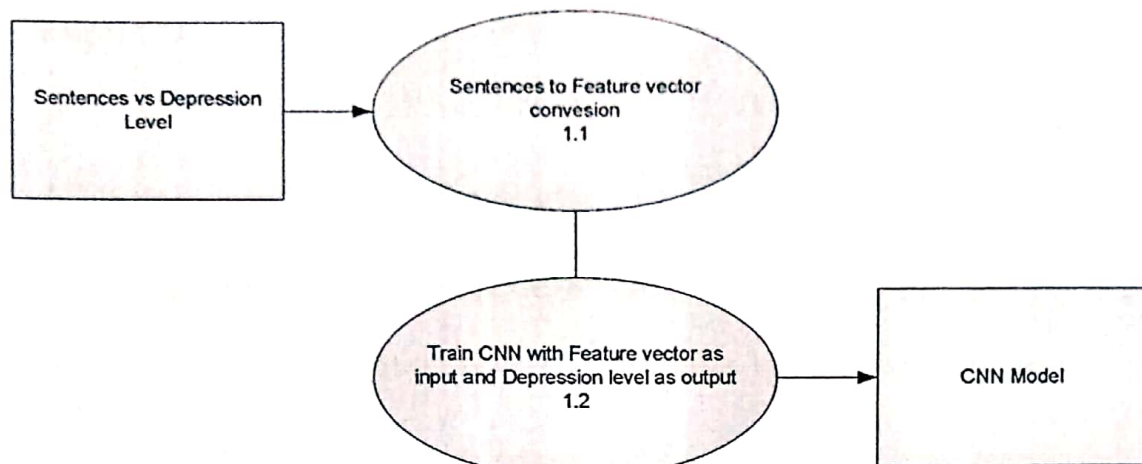


Fig 5.5.2 : Level 1 Data flow Diagram

The sentences are converted to feature vector using word embedding model . With Feature vector as input and the corresponding depression level as output , a Convolutional Neural Network is trained.

Classification process is split into sub process as shown below

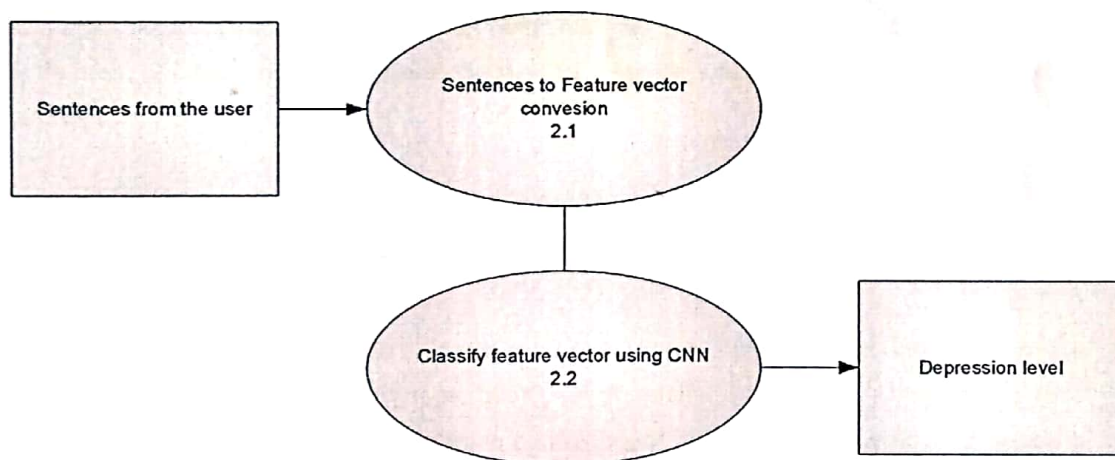


Fig 5.5.2 : Classification Process

Sentences from the user is taken as input and it is converted to feature vector using word embedding model. The feature vector is then classified using CNN to give the depression level as output.

Chapter 6

IMPLEMENTATION

6.1 Implementation requirements

6.1.1 Programming language selection

Python is selected as the programming language for this project. It is selected for the following reasons.

1. The time to develop the code is reduced due to availability of image processing and machine learning modules in python
2. It is easy to debug errors and fix it in python
3. The code becomes portable to any environment.

6.1.2 Key features of python

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language.

Features in Python

There are many features in Python, some of which are discussed below –

Easy to code: Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

Free and open source: Python language is freely available at the official website. Since it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

Object oriented language: One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

GUI support: Graphical User interfaces can be made using a module such as PyQt5, PyQt4,. PyQt5 is the most popular option for creating graphical apps with Python.

High level language: Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

Extensible language: Python is a Extensible language. We can write us some Python code into C or C++ language and also we can compile that code in C/C++ language.

Portable language: Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

Integrated language: Python is also an Integrated language because we can easily integrated python with other languages like c, c++, etc.

Interpreted language: Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.

Standard library: Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

Dynamically typed language: Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

6.1.2Tools Used

IDLE 3.6 tool is used for development of the project. IDLE (Integrated Development and Learning Environment) is an integrated development environment (IDE) for Python. The Python installer for Windows contains the IDLE module by default. IDLE can be used to execute a single statement just like Python Shell and also to create, modify, and execute Python scripts.

IDLE can be used to execute a single statement just like Python Shell and also to create, modify, and execute Python scripts. IDLE provides a fully-featured text editor to create Python script that

includes features like syntax highlighting, autocompletion, and smart indent. It also has a debugger with stepping and breakpoints features.

6.2 Coding guideline

PEP 8 coding standard is followed for the development of this project.

PEP 8, sometimes spelled PEP8 or PEP-8, is a document that provides guidelines and best practices on how to write Python code. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The primary focus of PEP 8 is to improve the readability and consistency of Python code.

PEP stands for Python Enhancement Proposal, and there are several of them. A PEP is a document that describes new features proposed for Python and documents aspects of Python, like design and style, for the community.

Following are the convention used for each of features of the language in this project.

Type	Naming Convention	Examples
Function	Use a lowercase word or words. Separate words by underscores to improve readability.	function, my_function
Variable	Use a lowercase single letter, word, or words. Separate words with underscores to improve readability.	x, var, my_variable
Class	Start each word with a capital letter. Do not separate words with underscores. This style is called camel case.	Model, MyClass
Method	Use a lowercase word or words. Separate words with underscores to improve readability.	class_method, method
Constant	Use an uppercase single letter, word, or words. Separate words with underscores to improve readability.	CONSTANT, MY_CONSTANT, MY_LONG_CONSTANT

Type	Naming Convention	Examples
Module	Use a short, lowercase word or words. Separate words with underscores to improve readability.	module.py, my_module.py
Package	Use a short, lowercase word or words. Do not separate words with underscores.	package, mypackage

6.3 Pseudo code

The pseudo code for training algorithm is given below

Algorithm: Training

- Input : Labelled dataset
- Output : Neural Classifier

Step1 : Vectorize training dataset (remove stop words)

Step2 : Create a Term Frequency(TF) vector for each sentence in dataset

Step 3 : Create training set with TF as input and label (neutral, depressed, not depressed) as output

Step 4: Train Neural Network with TF as input and label as output

Algorithm : Classification

- Input : social media messages from user, Neural model
- Output : Neutral/Depressed/ Not Depressed

StatusCount= [0,0,0]

Stmsg= ["Neutral", "Depressed", "Not Depressed"]

For each message in input

TF<- vectorize the message

status = Classify using neural model (TF)

StatusCount[status]+=1;

End

Return Stmsg[Index(Max(StatusCount))]

6.4 Summary

The implementation details in terms of pseudo code, programming language and tools used are detailed in this chapter.

Type	Naming Convention	Examples
Module	Use a short, lowercase word or words. Separate words with underscores to improve readability.	module pv, my_module pv
Package	Use a short, lowercase word or words. Do not separate words with underscores.	package, mypackage

6.3 Pseudo code

The pseudo code for training algorithm is given below

Algorithm: Training

- Input : Labelled dataset
- Output : Neural Classifier

Step1 : Vectorize training dataset (remove stop words)

Step2 : Create a Term Frequency(TF) vector for each sentence in dataset

Step 3 : Create training set with TF as input and label (neutral, depressed, not depressed) as output

Step 4: Train Neural Network with TF as input and label as output

Algorithm : Classification

- Input : social media messages from user, Neural model
 - Output : Neutral/Depressed/ Not Depressed
- StatusCount= [0,0,0]

Stmsg= ["Neutral", "Depressed", "Not Depressed"]

For each message in input

TF<- vectorize the message

status = Classify using neural model (TF)

StatusCount[status] += 1;

End

Return Stmsg[Index(Max(StatusCount))]

6.4 Summary

The implementation details in terms of pseudo code, programming language and tools used are detailed in this chapter.

SOURCE CODE :

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
"""
```

Created on the day we all start to love our self.

@author: Nikie Jo Deocampo

```
"""
```

```
import json
```

```
import pandas as pd
```

```
import time
```

```
import numpy as np
```

```
import itertools
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.metrics import confusion_matrix
```

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
from sklearn import metrics
```

```
#from sklearn.metrics import roc_auc_score
```

```
tweets_data = []
```

```
x = []
```

```
y = []
```

```
vectorizer = CountVectorizer(stop_words='english')
```

```
def retrieveTweet(data_url):
```



```
tweets_data_path = data_url

tweets_file = open(tweets_data_path, "r")

for line in tweets_file:

    try:

        tweet = json.loads(line)

        tweets_data.append(tweet)

    except:

        continue

def retrieveProcessedData(Pdata_url):

    sent = pd.read_excel(Pdata_url)

    for i in range(len(tweets_data)):

        if tweets_data[i]['id']==sent['id'][i]:

            x.append(tweets_data[i]['text'])

            y.append(sent['sentiment'][i])

def plot_confusion_matrix(cm, classes,

                           normalize=False,

                           title='Confusion matrix',

                           cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick_marks = np.arange(len(classes))
```

```
plt.xticks(tick_marks, classes, rotation=45)

plt.yticks(tick_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()

plt.ylabel("True label")

plt.xlabel("Predicted label")

def nbTrain():

from sklearn.naive_bayes import MultinomialNB

start_timenb = time.time()

train_features = vectorizer.fit_transform(x)

actual = y

nb = MultinomialNB()

nb.fit(train_features, [int(r) for r in y])

test_features = vectorizer.transform(x)

predictions = nb.predict(test_features)

fpr, tpr, thresholds = metrics.roc_curve(actual, predictions, pos_label=1)

nbscore = format(metrics.auc(fpr, tpr))
```

```
nbscore = float(nbscore)*100

# nb_matrix = confusion_matrix(actual, predictions)

# plt.figure()

# plot_confusion_matrix(nb_matrix, classes=[-1,0,1], title='Confusion matrix For NB
classifier')

print("\n")

print("Naive Bayes Accuracy : \n", nbscore,"%")

print(" Completion Speed", round((time.time() - start_timenb),5))

print()

def datree():

from sklearn import tree

    start_timedt = time.time()

    train_featurestree = vectorizer.fit_transform(x)

    actual1 = y

    test_features1 = vectorizer.transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train_featurestree, [int(r) for r in y])

    prediction1 = dtree.predict(test_features1)

ddd, ttt, thresholds = metrics.roc_curve(actual1, prediction1, pos_label=1)

dtreescore = format(metrics.auc(ddd, ttt))

dtreescore = float(dtreescore)*100

print("Decision tree Accuracy : \n", dtreescore, "%")
```

```
print(" Completion Speed", round((time.time() - start_time),5))

print()

def Tsvm()

from sklearn.svm import SVC

    start_timesvm = time.time()

    train_featuressvm = vectorizer.fit_transform(x)

    actual2 = y

    test_features2 = vectorizer.transform(x)

svc = SVC()

svc = svc.fit(train_featuressvm, [int(r) for r in y])

    prediction2 = svc.predict(test_features2)

sss, vvv, thresholds = metrics.roc_curve(actual2, prediction2, pos_label=1)

svc = format(metrics.auc(sss, vvv))

svc = float(svc)*100

print("Support vector machine Accuracy : \n", svc, "%")

print(" Completion Speed", round((time.time() - start_timesvm),5))

print()

def knN():

from sklearn.neighbors import KNeighborsClassifier

    start_timekn = time.time()

    train_featureskn = vectorizer.fit_transform(x)

    actual3 = y
```



```
test_features3 = vectorizer.transform(x)
kn = KNeighborsClassifier(n_neighbors=2)
kn = kn.fit(train_featureskn, [int(i) for i in y])
prediction3 = kn.predict(test_features3)
kkk, nnn, thresholds = metrics.roc_curve(actual3, prediction3, pos_label=1)
kn = format(metrics.auc(kkk, nnn))
kn = float(kn)*100
print("Kneighborsclassifier Accuracy : \n", kn, "%")
print(" Completion Speed", round((time.time() - start_timekn),5))
print()
def RanFo():
from sklearn.ensemble import RandomForestClassifier
start_timerf = time.time()
train_featuresrf = vectorizer.fit_transform(x)
actual4 = y
test_features4 = vectorizer.transform(x)
rf = RandomForestClassifier(max_depth=2, random_state=0)
rf = rf.fit(train_featuresrf, [int(i) for i in y])
prediction4 = rf.predict(test_features4)
rrr, fff, thresholds = metrics.roc_curve(actual4, prediction4, pos_label=1)
kn = format(metrics.auc(rrr, fff))
kn = float(kn)*100
```

```
print("Random Forest Accuracy : \n", kn, "%")
print(" Completion Speed", round((time.time() - start_timerf),5))
print()
print()
def runall():
    retrieveTweet('data/tweetdata.txt')
    retrieveProcessedData('processed_data/output.xlsx')
    nbTrain()
    datree()
    Tsvm()
    knN()
    RanFo()
    def datreeINPUT(inputtweet):
        from sklearn import tree
        train_featurestree = vectorizer.fit_transform(x)
        dtree = tree.DecisionTreeClassifier()
        dtree = dtree.fit(train_featurestree, [int(r) for r in y])
        inputdtree= vectorizer.transform([inputtweet])
        predictt = dtree.predict(inputdtree)
        if predictt == 1:
            predictt = "Positive"
        elif predictt == 0:
```

```
predictt = "Neutral"

elif predictt == -1:

predictt = "Negative"

else:

print("Nothing")

print("\n*****")

print(predictt)

print("*****")

runall()

#print("Input your tweet : ")

#inputtweet = input()

#

#datreeINPUT(inputtweet)
```

SOURCE CODE :

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-

"""

Created on the day we all start to love our self.

"""

import json

import pandas as pd
```

```
import time

import numpy as np

import itertools

import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix

from sklearn.feature_extraction.text import CountVectorizer

from sklearn import metrics

import os

from tkinter import *

from tkinter import filedialog

import tkinter as tk

import numpy as np

import pandas as pd

from tkinter import filedialog as fd

from tkinter import messagebox

from tkinter import simpledialog

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVR

from PIL import ImageTk, Image

import operator

from sklearn.neural_network import MLPClassifier

#from sklearn.metrics import roc_auc_score
```



```
tweets_data = []

x = []

y = []

vectorizer = CountVectorizer(stop_words='english')

def retrieveTweet(data_url):

    tweets_data_path = data_url

    tweets_file = open(tweets_data_path, "r")

    for line in tweets_file:

        try:

            tweet = json.loads(line)

            tweets_data.append(tweet)

        except:

            continue

def retrieveProcessedData(Pdata_url):

    sent = pd.read_excel(Pdata_url)

    for i in range(len(tweets_data)):

        if tweets_data[i]['id']==sent['id'][i]:

            x.append(tweets_data[i]['text'])

            y.append(sent['sentiment'][i])

def plot_confusion_matrix(cm, classes,

                           normalize=False,
```

```
title='Confusion matrix',
cmap=plt.cm.Blues):
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()

tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)

fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], fmt),
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

plt.show()

def nnTrain():
    from sklearn.naive_bayes import MultinomialNB

    start_timenb = time.time()
```

```
train_features = vectorizer.fit_transform(x)

actual = y

nb = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(35), random_state=1)

nb.fit(train_features, [int(r) for r in y])

test_features = vectorizer.transform(x)

predictions = nb.predict(test_features)

fpr, tpr, thresholds = metrics.roc_curve(actual, predictions, pos_label=1)

nbscore = format(metrics.auc(fpr, tpr))

nbscore = float(nbscore)*100

nb_matrix = confusion_matrix(actual, predictions)

plt.figure()

plot_confusion_matrix(nb_matrix, classes=[-1,0,1], title='Confusion matrix For NN
classifier')

print("\n")

# test_try= vectorizer.transform(["Lets help those in need, fight anxiety and bring
happiness"])

# test_try2= vectorizer.transform(["Dont look down at people with anxiety rather give
love and respect to all. shout! Equality."])

# predictr = nb.predict(test_try)

# predictt = nb.predict(test_try2)

# print(predictr)

# print(predictt)
```

```
print("Neural Network Accuracy : \n", nb_score,"%")

print(" Completion Speed", round((time.time() - start_time_nb),5))

print()

def nbTrain():

from sklearn.naive_bayes import MultinomialNB

    start_time_nb = time.time()

    train_features = vectorizer.fit_transform(x)

actual = y

nb = MultinomialNB()

nb.fit(train_features, [int(r) for r in y])

    test_features = vectorizer.transform(x)

predictions = nb.predict(test_features)

fpr, tpr, thresholds = metrics.roc_curve(actual, predictions, pos_label=1)

nb_score = format(metrics.auc(fpr, tpr))

nb_score = float(nb_score)*100


    #nb_matrix = confusion_matrix(actual, predictions)

    #plt.figure()

    #plot_confusion_matrix(nb_matrix, classes=[-1,0,1], title='Confusion matrix For NB
classifier')

    #print("\n")
```



```
# test_try= vectorizer.transform(["Lets help those in need, fight anxiety and bring
happiness"])

# test_try2= vectorizer.transform(["Dont look down at people with anxiety rather give
love and respect to all. shout! Equality."])

# predictr = nb.predict(test_try)

# predictt = nb.predict(test_try2)

# print(predictr)

# print(predictt)

print("Naive Bayes Accuracy : \n", nb.score,"%")

print(" Completion Speed", round((time.time() - start_timenb),5))

print()

def datree():

from sklearn import tree

    start_timedt = time.time()

    train_featurestree = vectorizer.fit_transform(x)

    actual1 = y

    test_features1 = vectorizer.transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train_featurestree, [int(r) for r in y])

    prediction1 = dtree.predict(test_features1)

ddd, ttt, thresholds = metrics.roc_curve(actual1, prediction1, pos_label=1)
```

```
dtreescore = format(metrics.auc(ddd, ttt))

dtreescore = float(dtreescore)*100

print("Decision tree Accuracy : \n", dtreescore, "%")

print(" Completion Speed", round((time.time() - start_timedt),5))

print()

def Tsvm():

from sklearn.svm import SVC

    start_timesvm = time.time()

    train_featuressvm = vectorizer.fit_transform(x)

    actual2 = y

    test_features2 = vectorizer.transform(x)

svc = SVC()

svc = svc.fit(train_featuressvm, [int(r) for r in y])

    prediction2 = svc.predict(test_features2)

sss, vvv, thresholds = metrics.roc_curve(actual2, prediction2, pos_label=1)

svc = format(metrics.auc(sss, vvv))

svc = float(svc)*100

print("Support vector machine Accuracy : \n", svc, "%")

print(" Completion Speed", round((time.time() - start_timesvm),5))

print()

def knN():
```

```
from sklearn.neighbors import KNeighborsClassifier

start_timekn = time.time()

train_featureskn = vectorizer.fit_transform(x)

actual3 = y

test_features3 = vectorizer.transform(x)

kn = KNeighborsClassifier(n_neighbors=2)

kn = kn.fit(train_featureskn, [int(i) for i in y])

prediction3 = kn.predict(test_features3)

kkk, nnn, thresholds = metrics.roc_curve(actual3, prediction3, pos_label=1)

kn = format(metrics.auc(kkk, nnn))

kn = float(kn)*100

print("Kneighborsclassifier Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start_timekn),5))

print()

def RanFo():

from sklearn.ensemble import RandomForestClassifier

start_timerf = time.time()

train_featuresrf = vectorizer.fit_transform(x)

actual4 = y

test_features4 = vectorizer.transform(x)

rf = RandomForestClassifier(max_depth=2, random_state=0)
```

```
rf = rf.fit(train_featuresrf, [int(i) for i in y])

prediction4 = rf.predict(test_features4)

fpr, tpr, thresholds = metrics.roc_curve(actual4, prediction4, pos_label=1)

kn = format(metrics.auc(fpr, tpr))

kn = float(kn)*100

print("Random Forest Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start_timerf),5))

print()

print()

def runall():

retrieveTweet('data/tweetdata.txt')

retrieveProcessedData('processed_data/output.xlsx')

nnTrain()

nbTrain()

datree()

Tsvm()

knN()

RanFo()

def datrain():

from sklearn import tree

train_featurestree = vectorizer.fit_transform(x)
```



```
dtree = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(35),
random_state=1)

dtree = dtree.fit(train_featurestree, [int(r) for r in y])

return dtree

def datreeINPUT(inputtweet,dtree):

from sklearn import tree

inputdtree= vectorizer.transform([inputtweet])

predictt = dtree.predict(inputdtree)

print("The classification result for ',inputtweet,'is')

toret=0;

if predictt == 1:

predictt = "Positive"

toret=1;

elif predictt == 0:

predictt = "Neutral"

toret=0;

elif predictt == -1:

predictt = "Negative"

toret=2;

else:

print("Nothing")

toret=0;
```

```
print("\n*****")
print(predictt)
print("*****")
return toret
runall()
def processMessage():
    global nlabel
    global ndeplabel
    global rlabel
    filenameclassify = fd.askopenfilename()
    file1 = open(filenameclassify, 'r')
    Lines = file1.readlines()
    count = 0
    # Strips the newline character
    rstres=["Neutral","Not Depressed","Depressed"]
    rstatus=[0,0,0]
    datree=datrain()
    for line in Lines:
        count +=1
        pre=datreeINPUT(line,datree)
        print('got result',pre)
```

```
rstatus[pre]+=1

index, value = max(enumerate(rstatus), key=operator.itemgetter(1))

print(rstatus)

print(index)

print(value)

rlabel.config(text=rstres[index])


nums = "Total messages=" + str(count)

nlabel.config(text=nums)

if index==2:

    dl= value*100.0/count

    nums= "Depression level="+str(dl) + "%"

    ndeplabel.config(text=nums)

if __name__ == "__main__":

    global parent

    global nlabel

    global ndeplabel

    global rlabel

    parent = tk.Tk()

    parent.geometry("800x500")

    parent.configure(bg='black')
```

```
parent.title("Prediciton of Depression")

frame = tk.Frame(parent)
frame.pack()

image2 = Image.open('./back.jpg')
image1 = ImageTk.PhotoImage(image2)
your_label = Label(master=parent, image=image1)
your_label.place(x=0, y=0, relwidth=1, relheight=1)
your_label.pack()

w = tk.Label(frame, text="Depression Prediction from messages",
fg = "red",
font = ("Times New Roman", 18))
w.pack()

text_disp = tk.Button(frame,
text="SELECT MESSAGE TO CLASSIFY",
command=processMessage
)

text_disp.pack()

nlabel = tk.Label(frame, fg="green")
nlabel.pack()

ndeplabel = tk.Label(frame, fg="green")
```



```
ndeplabel.pack()

rlabel = tk.Label(frame, fg="red")

rlabel.pack()
```

```
parent.mainloop()

#print("Input your tweet : ")

#inputtweet = input()

#datreeINPUT(inputtweet)
```

SOURCE CODE :

```
from tweepy.streaming import StreamListener

from tweepy import OAuthHandler

from tweepy import Stream

consumer_key = 'd7RJDeV6M1TdKnXXdY29Zud5O'

consumer_secret = '8LV35luiAco2mBnQ1W6erOnA8cbMwVgxblfHjP5zk5dmAXGwd6'

access_token = '2206645458-9qlftwQ5eiovob7GCp21VrAoFRXi7AJLGt5ts3O'

access_secret = 'Oc9ZKbHSL0reJhZYcU0Vk9UERbVvsTwerIfDUTwiRNGYf'

class StdOutListener(StreamListener):

    def on_data(self, data):

        with open('data/tweetdata.txt','a') as tf:

            tf.write(data)
```

```
print(data)

return True

def on_error(self, status):

    print(status)


if __name__ == '__main__':

    l = StdOutListener()

    auth = OAuthHandler(consumer_key, consumer_secret)

    auth.set_access_token(access_token, access_secret)

    stream = Stream(auth, l)

    stream.filter(track=['depression', 'anxiety', 'mental health', 'suicide', 'stress', 'sad'])
```

SOURCE CODE:

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-

"""Created on the day we all start to love our self.

@author: Nikie Jo Deocampo

"""

import json

import pandas as pd

import time
```

```
import numpy as np

import itertools

import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix

from sklearn.feature_extraction.text import CountVectorizer

from sklearn import metrics

#from sklearn.metrics import roc_auc_score

tweets_data = []

x = []

y = []

vectorizer = CountVectorizer(stop_words='english')

def retrieveTweet(data_url):

    tweets_data_path = data_url

    tweets_file = open(tweets_data_path, "r")

    for line in tweets_file:

        try:

            tweet = json.loads(line)

            tweets_data.append(tweet)

        except:

            continue

def retrieveProcessedData(Pdata_url):

    sent = pd.read_excel(Pdata_url)
```

```
for i in range(len(tweets_data)):
    if tweets_data[i]['id']==sent['id'][i]:
        x.append(tweets_data[i]['text'])
        y.append(sent['sentiment'][i])

def plot_confusion_matrix(cm, classes,
                           normalize=False,
                           title='Confusion matrix',
                           cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
```



```
plt.xlabel('Predicted label')

def nbTrain():

    from sklearn.naive_bayes import MultinomialNB

    start_timenb = time.time()

    train_features = vectorizer.fit_transform(x)

    actual = y

    nb = MultinomialNB()

    nb.fit(train_features, [int(r) for r in y])

    test_features = vectorizer.transform(x)

    predictions = nb.predict(test_features)

    fpr, tpr, thresholds = metrics.roc_curve(actual, predictions, pos_label=1)

    nb_score = format(metrics.auc(fpr, tpr))

    nb_score = float(nb_score)*100

    # nb_matrix = confusion_matrix(actual, predictions)

    # plt.figure()

    # plot_confusion_matrix(nb_matrix, classes=[-1,0,1], title='Confusion matrix For NB
    classifier')

    print("\n")

    print("Naive Bayes Accuracy : \n", nb_score,"%")

    print(" Completion Speed", round((time.time() - start_timenb),5))
```

```
print()

def datree():

    from sklearn import tree

    start_timedt = time.time()

    train_featurestree = vectorizer.fit_transform(x)

    actual1 = y

    test_features1 = vectorizer.transform(x)

    dtree = tree.DecisionTreeClassifier()

    dtree = dtree.fit(train_featurestree, [int(r) for r in y])

    prediction1 = dtree.predict(test_features1)

    ddd, ttt, thresholds = metrics.roc_curve(actual1, prediction1, pos_label=1)

    dtreescore = format(metrics.auc(ddd, ttt))

    dtreescore = float(dtreescore)*100

    print("Decision tree Accuracy : \n", dtreescore, "%")

    print(" Completion Speed", round((time.time() - start_timedt),5))

    print()

def Tsvm():

    from sklearn.svm import SVC

    start_timesvm = time.time()

    train_featuessvm = vectorizer.fit_transform(x)

    actual2 = y

    test_features2 = vectorizer.transform(x)
```

```
svc = SVC()

svc = svc.fit(train_featuressvm, [int(r) for r in y])

prediction2 = svc.predict(test_features2)

sss, vvv, thresholds = metrics.roc_curve(actual2, prediction2, pos_label=1)

svc = format(metrics.auc(sss, vvv))

svc = float(svc)*100

print("Support vector machine Accuracy : \n", svc, "%")

print(" Completion Speed", round((time.time() - start_timesvm),5))

print()

def knN():

from sklearn.neighbors import KNeighborsClassifier

    start_timekn = time.time()

    train_featureskn = vectorizer.fit_transform(x)

    actual3 = y

    test_features3 = vectorizer.transform(x)

kn = KNeighborsClassifier(n_neighbors=2)

kn = kn.fit(train_featureskn, [int(i) for i in y])

    prediction3 = kn.predict(test_features3)

kkk, nnn, thresholds = metrics.roc_curve(actual3, prediction3, pos_label=1)

kn = format(metrics.auc(kkk, nnn))

kn = float(kn)*100

print("Kneighborsclassifier Accuracy : \n", kn, "%")
```

```
print(" Completion Speed", round((time.time() - start_timekn),5))

print()

def RanFo():

from sklearn.ensemble import RandomForestClassifier

    start_timerf = time.time()

    train_featuresrf = vectorizer.fit_transform(x)

    actual4 = y

    test_features4 = vectorizer.transform(x)

rf = RandomForestClassifier(max_depth=2, random_state=0)

rf = rf.fit(train_featuresrf, [int(i) for i in y])

    prediction4 = rf.predict(test_features4)

rrr, fff, thresholds = metrics.roc_curve(actual4, prediction4, pos_label=1)

kn = format(metrics.auc(rrr, fff))

kn = float(kn)*100

print("Random Forest Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start_timerf),5))

print()

print()

def runall():

retrieveTweet('data/tweetdata.txt')

retrieveProcessedData('processed_data/output.xlsx')
```



```
# nbTrain()

# datree()

# Tsvm()

# knN()

# RanFo()

def datreeINPUT(inputtweet):

    from sklearn import tree

    train_featurestree = vectorizer.fit_transform(x)

    dtree = tree.DecisionTreeClassifier()

    dtree = dtree.fit(train_featurestree, [int(r) for r in y])

    inputdttree= vectorizer.transform([inputtweet])

    predictt = dtree.predict(inputdttree)

    if predictt == 1:

        predictt = "Positive"

    elif predictt == 0:

        predictt = "Neutral"

    elif predictt == -1:

        predictt = "Negative"

    else:

        print("Nothing")

    print("\n*****")
```

```
print(predictt)

print("*****")

runall()

print("\nInput your tweet : ")

inputtweet = input()

datreeINPUT(inputtweet)
```

SOURCE CODE :

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-

"""
```

Created on Thu Jul 26 16:34:13 2018

@author: Nikie Jo Deocampo

```
"""

import json

import csv

from nltk.tokenize import word_tokenize

import string

import re

import time

import pandas as pd
```

```
tweets_data = []  
  
x = []  
  
y = []  
  
k = []  
  
some_milby = []  
  
print("=====")  
  
print("Starting Preprocess Function")  
  
print("===== \n\n")  
  
def getdata(dataurl):  
  
    print("=====")  
  
    print("Retrieving TXT File")  
  
        tweets_data_path = dataurl  
  
        tweets_file = open(tweets_data_path, "r")  
  
    for line in tweets_file:  
  
        try:  
  
            tweet = json.loads(line)  
  
                tweets_data.append(tweet)  
  
        except:  
  
            continue  
  
    print("=====")  
  
    print("Retrieving Successfull")  
  
    print("===== \n \n")
```

```
time.sleep(3)

processdata()

def processdata():

    print("=====")

    print("Recovering Data Teets")

    print("=====")

    time.sleep(1)

    RE_EMOJI = re.compile('[\U00010000-\U0010ffff]', flags=re.UNICODE)

    for i in range(len(tweets_data)):

        q = tweets_data[i]['text']

        o = tweets_data[i]['id_str']

        q = RE_EMOJI.sub(r'', q)

        i = q.translate(str.maketrans("", "", string.punctuation))

    x.append(i)

    k.append(o)

    print("=====")

    print("Data Tweets Recovered")

    print("=====\\n\\n")

    def readdict(dataurl):

        print("=====")

        print("Reading Dictionary")

        print("=====")
```

```
with open(dataurl) as tsvfile:
    reader = csv.reader(tsvfile, delimiter='\\t')
    for row in reader:
        i = []
        i.append(row[2])
        i.append(row[5])
        y.append(i)
        print("=====")
        print("Dictionary Preparation Done")
        print("=====\n\n")
        addpolarity()

def addpolarity():
    start_time = time.time()
    counter = 0
    print("=====")
    print("Processing please wait...")
    print("=====\n\n")
    for j in x:
        tweet_token = j
        token = word_tokenize(tweet_token)
        sumnum = 0
```



```
sum_word = 0

for t in token:

    for d in y:

        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 >= 0.2:

        sum_more = 1

    elif (sum_more < 0.2) and (sum_more > -0.5):

        sum_more = 0
```

```
elif sum_more <= -0.5:

    sum_more = -1

else:

    print("****")

    sum_var = []

    varid = k[counter]

    sum_var.append(varid)

    sum_var.append(sum_more)

    some_milby.append(sum_var)

    counter += 1

    print("Processing time: ", round((time.time() - start_time),8), "Seconds \n\n")

    time.sleep(3)

    print("=====")

    print("Processing Finish")

    print("=====")


savetoxlsx()

def savetoxlsx():

    df = pd.DataFrame(some_milby)

    df.to_excel('processed_data/output.xlsx', header=("id","sentiment"), index=False)

    #file = open("testfile_data.txt","w")

    #file.write(some_milby)
```

```
#file.close()

print("=====")

print("Data Saved!")

print("=====")

def runall():

    getdata('data/tweetdata.txt')

    readdict('data/dictionary.tsv')

runall()
```

Chapter 7

TESTING

7.1 DATASET

The dataset used for experimentation is collected from following link

- <https://www.kaggle.com/ywang311/twitter-sentiment>
- We use the sentiment140 dataset. It contains 1,600,000 tweets extracted using the twitter api. The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment.

7.2 TEST PROCEDURE

The data is split into 80:20 ratio.

80% of data is used for training following classifiers of

1. Neural network
2. Random Forest
3. Decision tree
4. Naïve Bayes
5. SVM

7.3 UNIT TESTING

The unit test cases for Main module is given below

Test case	Description	Expected output	Remarks
LoadDataset	User provides the training dataset	Training dataset is taken for processing	Pass
Train	Train the classifiers	Traning function is called on classifiers	Pass

classifyText	Text file is called for classification	Each line is processed	Pass
verifyResults	Accuracy and classification results for text file are queried	Queried results are returned	Pass

The unit test cases for CNN module is given below

Test case	Description	Expected output	Remarks
Train	CNN Classifier is trained	Trained CNN is available	Pass
Classify	Sentence is given as input for CNN	Sentiment of sentence is given as output	Pass

7.4 INTEGRATED TESTING

The integration test cases for the project is given below

Test case	Description	Expected output	Remarks
User starts the application	Training is done when application is started	All classifiers are trained and accuracy is printed	Pass
User browse and select message to classify	Depression is detected from sentences	Depression status is printed by processing each sentence in the input document	Pass

7.5 SUMMARY

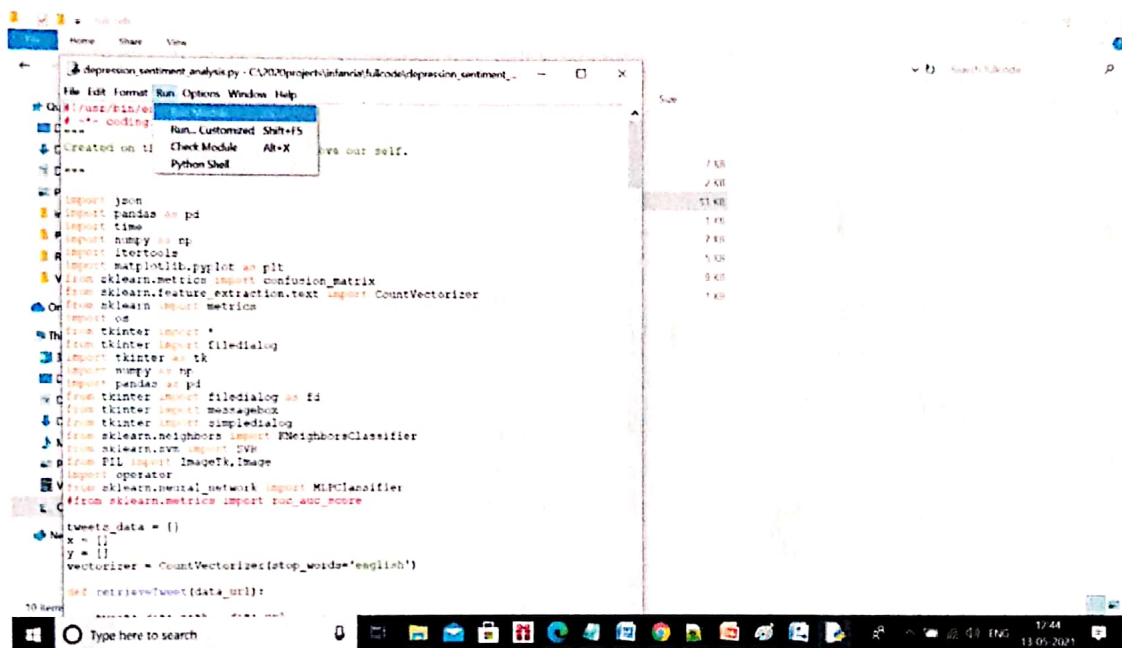
The test cases and test results are presented in this chapter.

Chapter 8

RESULT ANALYSIS

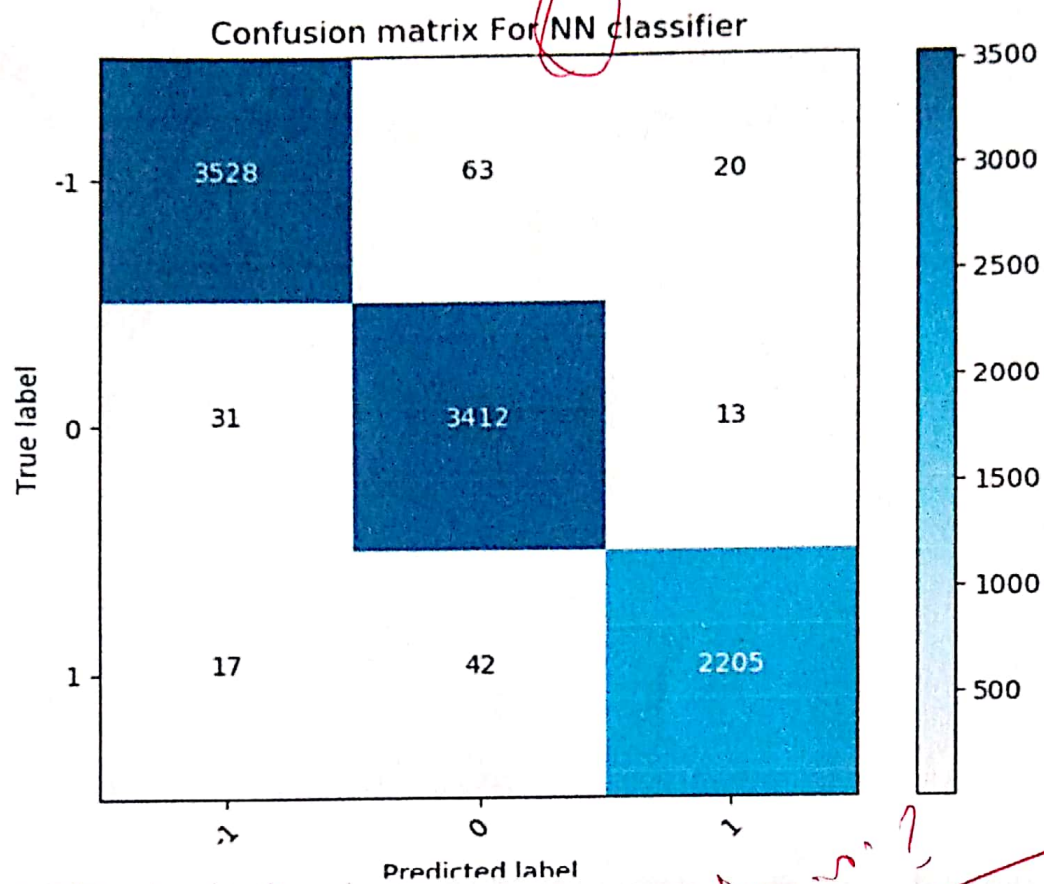
8.1 SNAPSHOTS

Execution of the application



Confusion matrix is Neural network

Fig. 2



Accuracy is calculated for each classifier

→ fig 5.1

Accuracy in each classifier

```
Neural Network Accuracy :  
98.74603179762005 %  
Completion Speed 66.11909  
  
Naive Bayes Accuracy :  
93.79406648429645 %  
Completion Speed 0.32654  
  
Decision tree Accuracy :  
98.55668748040587 %  
Completion Speed 1.56933  
  
Support vector machine Accuracy :  
93.62738823407057 %  
Completion Speed 13.66283  
  
Kneighborsclassifier Accuracy :  
81.464022923447 %  
Completion Speed 3.4172  
  
Random Forest Accuracy :  
46.92481190883222 %  
Completion Speed 1.31475
```

Project GUI opens up

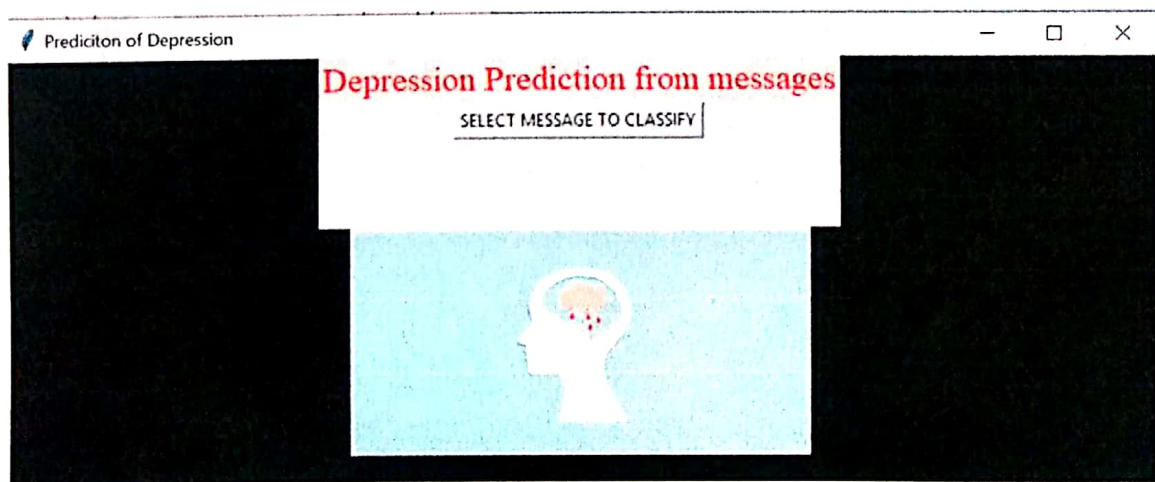
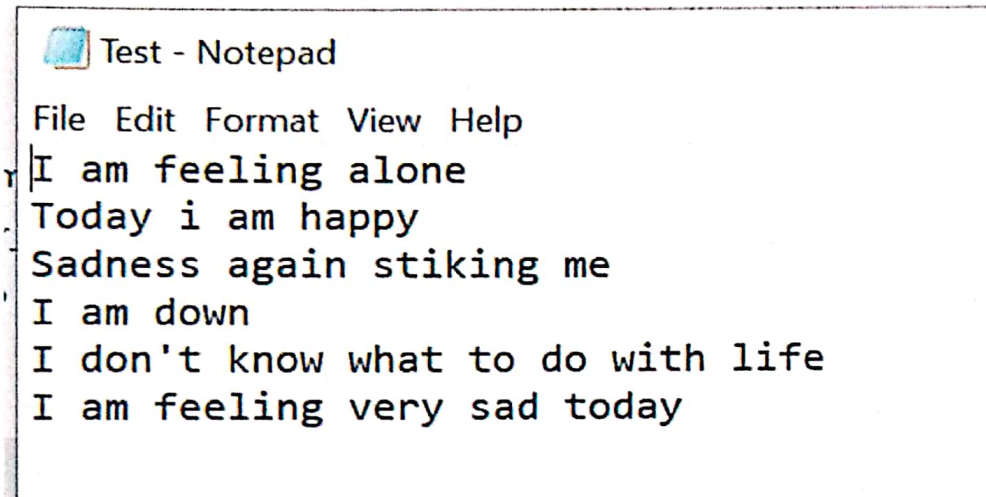


Fig no 1

Input messages given to classification



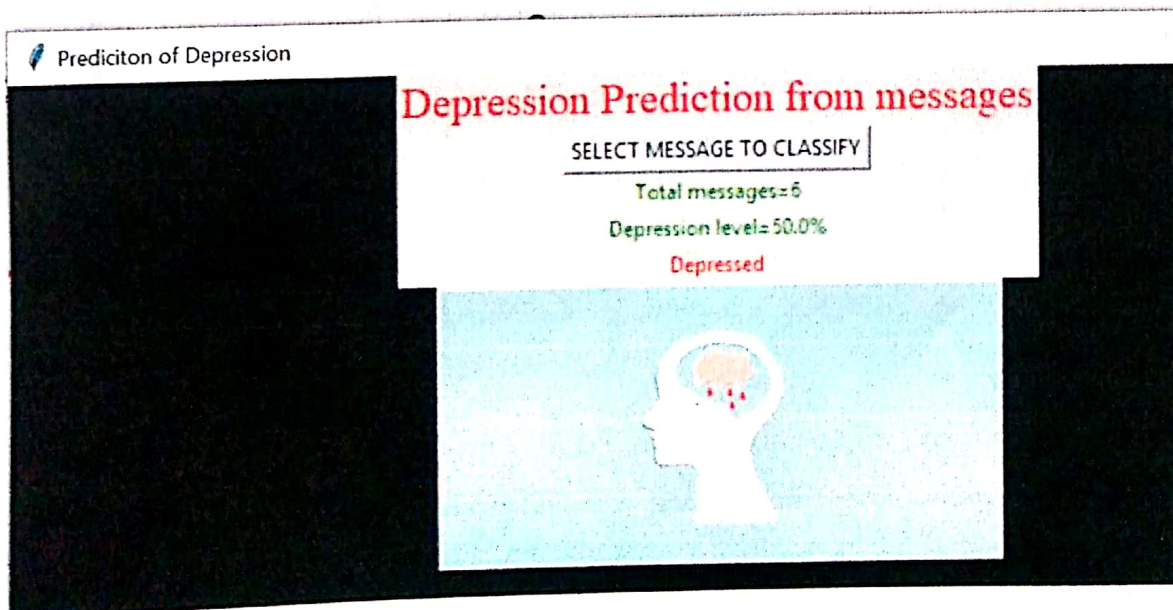
Each message is classified


```

The classificaiton result for Today i am happy
is
*****
Positive
*****
got result 1
The classificaiton result for Sadness again stiking me
is
*****
Negative
*****
got result 2
The classificaiton result for I am down
is
*****
Negative
*****
got result 2
The classificaiton result for I don't know what to do with life
is
*****
Neutral
*****
got result 0
The classificaiton result for I am feeling very sad today is
*****
Negative
*****

```

Depression status is displayed



REFERENCES:

- [1] Torii, L. Yin, T. Nguyen, C. T. Mazumdar, . H. Liu, D. M. Hartley, and N. P. Nelson, "An exploratory study of a text classification framework for internet-based surveillance of emerging epidemics," *International Journal of Medical Informatics*, vol.80,no. 1, pp. 56–66, 2011.
- [2] Schomerus, H. Matschinger, and M. C. Angermeyer, "The stigma of psychiatric treatment and help-seeking intentions for de-pression," *European Archives of Psychiatry and Clinical Neuroscience*, vol. 259, no. 5, pp. 298–306, 2009.
- [3] Gowen, M. Deschaine, D. Gruttadara, and D. Markey, "Young adults with mental health conditions and social networking web-sites: Seeking tools to build community." *Psychiatric Rehabilitation Journal*, vol. 35, no. 3, pp. 245–250, 2012.
- [4] Rani, C. Liu, N. Sarkar, and E. Vanman, "An empirical study of machine learning techniques for affect recognition in human–robot interaction," *Pattern Analysis and Applications*, vol. 9, no. 1, pp. 58–69, 2006.
- [5] Paykel, "Basic concepts of depression," *Dialogues in Clinical Neuroscience*, vol. 10, no. 3, pp. 279–289, 2008
- [6] Naslund, K. A. Aschbrenner, L. A. Marsch, and S. J. Bartels, "The future of mental health care: peer-to-peer support and social media," *Epidemiology and Psychiatric Sciences*, vol. 25, no. 2, pp.113–122, 2016.
- [7] Porges, "Emotion recognition in children with autism spectrum disorders: Relations to eye gaze and autonomic state," *Journal of autism and developmental disorders*, vol. 40, no. 3, pp. 358–370, 2010.
- [8] Kim, S. Bang, and S. Kim, "Emotion recognition system using short-term monitoring of physiological signals," *Medical and biological engineering and computing*, vol.42, no.3, pp.419–427, 2004

- [9] Collignon, S. Girard, F. Gosselin, S. Roy, D. Saint-Amour, M. Las-sonde, and F. Lepore, "Audio-visual integration of emotion expression," vol.1242, pp.126–135, 2008.
- [10] Berger, T. H. Wagner, and L. C. Baker, "Internet use and stigmatized illness," *Social Science & Medicine*, vol. 61, no. 8, pp.1821–1827, 2005.
- [11] Rude, E.-M. Gortner, and J. Pennebaker, "Language use of depressed and depression-vulnerable college students," *Cognition & Emotion*, vol. 18, no. 8, pp. 1121–1133, 2004.
- [12] Mohammad and P. D. Turney, "Crowdsourcing a word-emotion association lexicon," *Computational Intelligence*, vol. 29, no. 3, pp. 436–465, 2013
- [13] Tausczik and J. W. Pennebaker, "The psychological meaning of words: Liwc and computerized text analysis methods," *Journal of Language and Social Psychology*, vol. 29, no. 1, pp. 24–54, 2010
- [14] Pang and L. Lee, "Opinion mining and sentiment analysis, *Foundations and Trends in Information Retrieval*, vol. 2, no. 1–2, pp.1–135, 2008.
- [15] Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.



GOLDEN VALLEY EDUCATIONAL TRUST

DR. T. THIMMAIAH INSTITUTE OF TECHNOLOGY

Oorgaam, KGF - 563 120.

Affiliated to Visvesvaraya Technological University Belagavi

Approved By AICTE Govt. of India New Delhi | ISO 21001: 2018 Certified




Certificate

This is to certify that *Miss. M Savitha Shree, Student, Dr.T.Thimmaiah institute of Technology,* presented a paper titled "*Utilizing Neural Network And Linguistic Metadata For Early Detection Of Depression Indications In Text Messages*" in the 3rd International Conference on Recent Trends in Technology, Engineering and Applied Science - ICRTEAS 2021 held virtually on 19th and 20th July 2021.


Dr. Palaniswamy K M
Convener


Prof. Ruckmani Divakaran
General Chair


Dr. H.S. Shenoy
Vice Principal


Dr. Syed Arif
Principal

CertificateID: ICRTEAS2021/R666



GOLDEN VALLEY EDUCATIONAL TRUST

Dr. T. THIMMAIAH INSTITUTE OF TECHNOLOGY

Oorgaam, KGf - 563 120.

Affiliated to Visvesvaraya Technological University Belagavi

Approved By AICTE Govt. of India New Delhi | ISO 21001: 2018 Certified



Certificate

This is to certify that *Miss. Infancia R, Student, Dr.T.Thimmaiah institute of Technology*, presented a paper titled "*Utilizing Neural Network And Linguistic Metadata For Early Detection Of Depression Indications In Text Messages*" in the 3rd International Conference on Recent Trends in Technology, Engineering and Applied Science - ICRTEAS 2021 held virtually on 19th and 20th July 2021.


Dr. Palaniswamy K M
Convener


Prof. Ruckmani Divakaran
General Chair


Dr. H.G. Shenoy
Vice Principal


Dr. Syed Ariff
Principal

CertificateID: ICRTEAS2021/R667



GOLDEN VALLEY EDUCATIONAL TRUST

DR. T. THIMMAIAH INSTITUTE OF TECHNOLOGY

Oorgaum, KGF - 563 120.

Affiliated to Visvesvaraya Technological University Belagavi

Approved By AICTE Govt. of India New Delhi | ISO 21001: 2018 Certified



Certificate

This is to certify that *Miss. Sandra Carolin S, Student, Dr.T.Thimmaiah institute of Technology,* presented a paper titled "*Utilizing Neural Network And Linguistic Metadata For Early Detection Of Depression Indications In Text Messages*" in the 3rd International Conference on Recent Trends in Technology, Engineering and Applied Science - ICRTTEAS 2021 held virtually on 19th and 20th July 2021.


Dr. Palaniswamy K M
Convener


Prof. Ruckmani Divakaran
General Chair


Dr. H.S. Shenoy
Vice Principal


Dr. Syed Arif
Principal

CertificateID: ICRTTEAS2021/R668



GOLDEN VALLEY EDUCATIONAL TRUST
Dr. T. THIMMAIAH INSTITUTE OF TECHNOLOGY

Oorgaum, KGF - 563 120.

Affiliated to Visvesvaraya Technological University Belagavi
Approved By AICTE Govt. of India New Delhi | ISO 21001: 2018 Certified



Certificate

This is to certify that *Miss. Maria Rebecca D, Student, Dr.T.Thimmaiah institute of Technology,*
presented a paper titled "*Utilizing Neural Network And Linguistic Metadata For Early
Detection Of Depression Indications In Text Messages*" in the 3rd International Conference on
Recent Trends in Technology, Engineering and Applied Science - ICRTTEAS 2021 held virtually on
19th and 20th July 2021.


Dr. Palaniswamy K M
Convenor


Prof. Ruckmani Divakaran
General Chair


Dr. H.G. Shenoy
Vice Principal


Dr. Syed Arif
Principal

CertificateID: ICRTTEAS2021/R669