VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Project Report
on

Depression Detection using Tweets

Submitted in partial fulfillment of the requirements for the VIII Semester of degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

by

Abhinav Soni
1RN18IS003

Under the Guidance of

Mr. R Rajkumar
Associate Professor
Department of ISE

ESTD:2001
An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru-560098

2021-2022

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Internship work entitled ***Deperession Detection using Tweets*** has been successfully completed by **Abhinav Soni (1RN18IS003)** a bonafide student of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements of 8th semester for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2021-2022**. The internship report has been approved as it satisfies the academic requirements in respect of internship work for the said degree.

Mr. R Rajkumar Internship Guide Associate Professor Department of ISE	**Dr. Suresh L** Professor and HoD Department of ISE RNSIT	**Dr. M K Venkatesha** Principal RNSIT

External Viva

Name of the Examiners	**Signature with Date**
1. _____	**1.** _____
2. _____	**2.** _____

DECLARATION

I, **ABHINAV SONI [USN: 1RN18IS003]** student of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***Depression Detection using Tweets*** has been carried out by us and submitted in partial fulfillment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2021-2022.

Place : Bengaluru

Date :

ABHINAV SONI

(1RN18IS003)

ABSTRACT

Social networks have been developed as a great point for its users to communicate with their interested friends and share their opinions, photos, and videos reflecting their moods, feelings and sentiments. This creates an opportunity to analyse social network data for user's feelings and sentiments to investigate their moods and attitudes when they are communicating via these online tools.

Although diagnosis of depression using social networks data has picked an established position globally, there are several dimensions that are yet to be detected. In this study, we aim to perform depression analysis on Facebook data collected from an online public source. To investigate the effect of depression detection, we propose machine learning technique as an efficient and scalable method.

The proliferations of internet and communication technologies, especially the online social networks have rejuvenated how people interact and communicate with each other electronically. The applications such as Facebook, Twitter, Instagram and alike not only host the written and multimedia contents but also offer their users to express their feelings, emotions and sentiments about a topic, subject or an issue online. On one hand, this is great for users of social networking site to openly and freely contribute and respond to any topic online; on the other hand, it creates opportunities for people working in the health sector to get insight of what might be happening at mental state of someone who reacted to a topic in a specific manner.

ACKNOWLEDGMENT

The fulfillment and rapture that go with the fruitful finishing of any assignment would be inadequate without the specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

I would like to profoundly thank **Management** of **RNS Institute of Technology** for providing such a healthy environment to carry out this Project work.

I would like to express my thanks to our Principal **Dr. M K Venkatesha** for his support and inspired me towards the attainment of knowledge.

I wish to place on record my words of gratitude to **Dr. Suresh L.,** Professor and Head of the Department, Information Science and Engineering, for being the enzyme and master mind behind my Project work.

I would like to express my profound and cordial gratitude to my Lab and Faculty In-charge **Dr. R. Rajkumar**, Assistant Professor, Department of Information Science and Engineering for their valuable guidance, constructive comments and continuous encouragement throughout the Project work.

I would like to express my profound and cordial gratitude to my Guide **Mrs. Kusuma S.,** Assitant Professor ,Department of Information Science and Engineering for his valuable guidance in preparing Project report.

I would like to thank all other teaching and non-teaching staff of Information Science & Engineering who have directly or indirectly helped me to carry out the project work.

And lastly, I would hereby acknowledge and thank my parents who have been a source of inspiration and also instrumental in carrying out this Project work.

ABHINAV SONI **USN- 1RN18IS003**

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

GUI	-	Graphical User Interface
RNN	-	Recurrent neural networks
RAM	-	Random Access Memory
ANN	-	Artificial neural network
LSTM	-	Long short-term memory

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATION

New Age Solutions Technologies (NASTECH) is a Microsoft Partner Network Company and CATC. With the vision to bridge the Academia-Industry Skill gap, we bring to you two of the most accepted concepts that are in demand currently and will huge scope in corporates for years to come.

1.1.1 COMPANY PROFILE

NASTECH is formed with the purpose of bridging the gap between Academia and Industry. Nastech is one of the leading Global Certification and Training service providers for technical and management programs for educational institutions. We collaborate with educational institutes to understand their requirements and form a strategy in consultation with all stakeholders to fulfil those by skilling, reskilling and upskilling the students and faculties on new age skills and technologies.

1.1.2 DOMAIN/TECHNOLOGY

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to natural intelligence displayed by animals including humans. Leading AI textbooks define the field as the study of "intelligent agents": any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however, this definition is rejected by major AI researchers.

Recurrent neural networks (RNN) are the state of the art algorithm for sequential data and are used by Apple's Siri and Google's voice search. It is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for machine learning problems that involve sequential data. It is one of the algorithms behind the scenes of the amazing achievements seen in deep learning over the past few years. In this post, we'll cover the basic concepts of how recurrent neural networks work, what the biggest issues are and how to solve them.

1.1.3 DEPARTMENT

Information Science and Engineering is entering its 20th year of academic excellence in the year 2021. The department was founded in 2001, from the starting of the Institution, with the goal of providing high-quality technical education in the field of information Science and Engineering. The department is accredited by the NBA [twice in 2011 and 2018 for three years] because it follows a quality procedure to prepare students to be industry ready and encourages them to pursue higher education.

Vision: Fostering winning professionals of Strong informative potential.

Mission: Imparting high quality education in the area of Information Science so as to graduate the students with good fundamentals, "Information System Integration", "Software Creation" capability & suitably train them to thrive in Industries, higher schools of learning with a comprehensive perspective.

1.2 PROBLEM STATEMENT

"***Depression Detection using Social Media Posts***", we choose Twitter as our platform to carry out our project because Tweets can be easily scrapped and they are easy to classify. Twitter also allows students to make a Twitter Developer account which will help us build a Twitter bot for further enhancements of our project.

1.2.1 Proposed solution

Our depression detector can be incorporated into existing products such as GBoard on Android, the Google Keyboard, which uses federated learning to improve the user experience based on their search history. As GBoard already collects user history, it is conceivable it can be extended to incorporate our model and identify instances of depression, especially based on user´s textual input over a period of time.

If signs of depression has been detected, then it would be desirable to suggest the user to use a self-care chatbot. (We need to think of a way to do this without infringing on the user's privacy, for example, we do not want to send this diagnosis back to the server in its raw form) Perhaps the suggestion of self-care bot can be an automatic feature that is integrated into GBoard upon depression detection, so that the raw data does not need to go back to the centralised server, and does not require revelation of the user's identity.

1.2.2 Problem Formulation

In this opportunity we'll go through a very particular topic. We all know the lockdown during the COVID-19 is affecting all of us in different ways, but the most frequents are depression and anxiety which is an expected outcome - the natural responses to confinement are precisely these, and most of the people don't even know it. It's been a hard time, people are afraid of uncertainty, of losing their jobs as many people have already done, the conditions are met for a major emotional imbalance.

Experts recommend to stay away from social media because it accelerates the depression process, and who is depressed already will be even more, however people expressions on it are a key instrument to determine how a population is feeling. Most of the social media active people express how they feel in tweets, Facebook posts, comments and even Instagram captions. So, starting from there, can we implement Deep Learning to discover depression and anxiety on tweets out there?

Overview:

- Topic Modelling - where we'll be looking for two labels: 1 - Depression & anxiety comments, 0 - Other, in Facebook comments prior known to contain depression expressions
- Topic Classification - We'll implement a Keras Recurrent Neural Network to find out depression in a tweets dataset.

To achieve both tasks, we'll go through:

- Data collection - Getting data from different sources to accomplish the main objective.
- Data cleaning - We'll have to take all the data which is already in different formats and clean it up to then be able to use it.
- Natural Language Processing for Topic Modelling - We'll need to transform the text data into a type that can be interpreted by ML models.
- Unsupervised Learning tasks for Topic Modelling - This is crucial, because most of the data we can find out there is unlabelled, so we first need to identify patterns in it.
- Supervised Learning tasks for Topic Classification- Once the data is labelled, we'll go through a Neural Network creation & training to then classify tweets.
- Predict depression and anxiety in unseen tweets before and after lockdown.
- Results' charting and conclusions.

CHAPTER 2

REQUIREMENT ANALYSIS & TOOLS

Systems analysis is the process of observing systems for troubleshooting or development purposes. It is applied to information technology, where computer-based systems require defined analysis according to their makeup and design.

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

2.1 RESOURCE REQUIREMENTS

HARDWARE REQUIREMENTS:

- processor : Pentium 4 processor
- Processor Speed : 2.4 GHz
- RAM : 1GB
- Storage Space : 40GB
- Monitor : 1024x768 or 1280x1024

SOFTWARE REQUIREMENTS:

- IDE : Jupyter Notebook
- operating system : Windows

2.2 Functional Requirements

Jupyter Notebook

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from Ipython in 2014 by Fernando Pérez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab. Jupyter is a NumFOCUS fiscally sponsored project.

Jupyter Notebook can connect to many kernels to allow programming in different languages. A Jupyter kernel is a program responsible for handling various types of requests (code execution, code completions, inspection), and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ, and thus can be on the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

Python

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a cycle-detecting garbage collection system (in addition to reference counting). Python 3.0 was released in 2008 and was a major revision of the language that is not complete backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020. Python consistently ranks as one of the most popular programming languages.

2.3 Tools, Languages

Tools used:

- Feature_extraction
- Regular Expression
- Numpy
- Pandas
- matplotlib
- pyplot
- nltk
- gensim
- NMF
- pyLDAvis
- pipeline
- logistic Regression
- svm
- confusion matrix

Language used **:** Python

Platform : Google Colab or Jupyter Notebook

CHAPTER 3

DESIGN AND IMPLEMENTATION

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

3.1 ARCHITECTURE

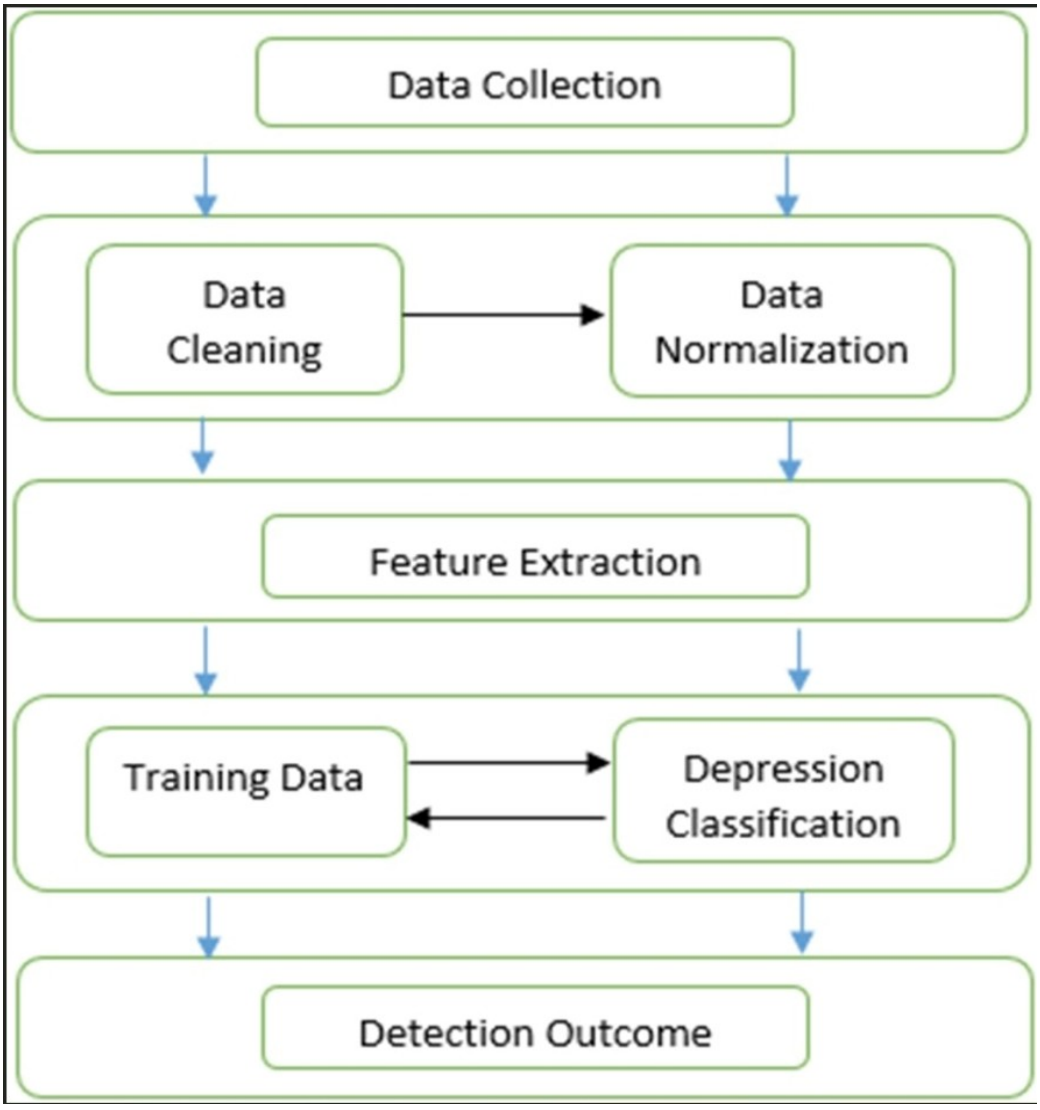

Fig 3.1 Architecture of the project

Data is collected from different tweets where we scrap the data from twitter using twint - Twint is an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter's API.
Twint utilizes Twitter's search operators to let you scrape Tweets from specific users, scrape Tweets relating to certain topics, hashtags & trends, or sort out sensitive information from Tweets like e-mail and phone numbers. I find this very useful, and you can get really creative with it too.

Twint also makes special queries to twitter allowing you to also scrape a Twitter user's followers, Tweets a user has liked, and who they follow without any authentication, API, Selenium, or browser emulation.

Later the data is cleaned by removing all the unnecessary tweets, We remove the tweets with URLs, Tweets with emails, lowercase all text, remove punctuation signs, remove stop words.

Model training: We'll explore how LDA and NMF can create the topics and depending on the outcomes we'll select the proper one for this project. Essentially we're looking for focused topics, otherwise the purpose of this project won't be reached.

The crucial difference between both models is that LDA adds a Dirichlet prior on top of the data generating process, meaning NMF qualitatively leads to worse mixtures, which could affect our dataset's topic quality.

Regarding the library we'll be using: Scikit-Learn - the reasons are more than obvious, even when Gensim has more capabilities, it's also more complex and much slower - we're looking to keep the things as simpler as possible and get results as quick as possible.

The outcome of this stage will be the original data frame with its labels: 1 for depression/anxiety comments and 0 for other type of comments.

Then we classify the scrapped tweets into either depressive or non – depressive with target value of 1 or 0 respectively and check for the accuracy of the model and the F- score.

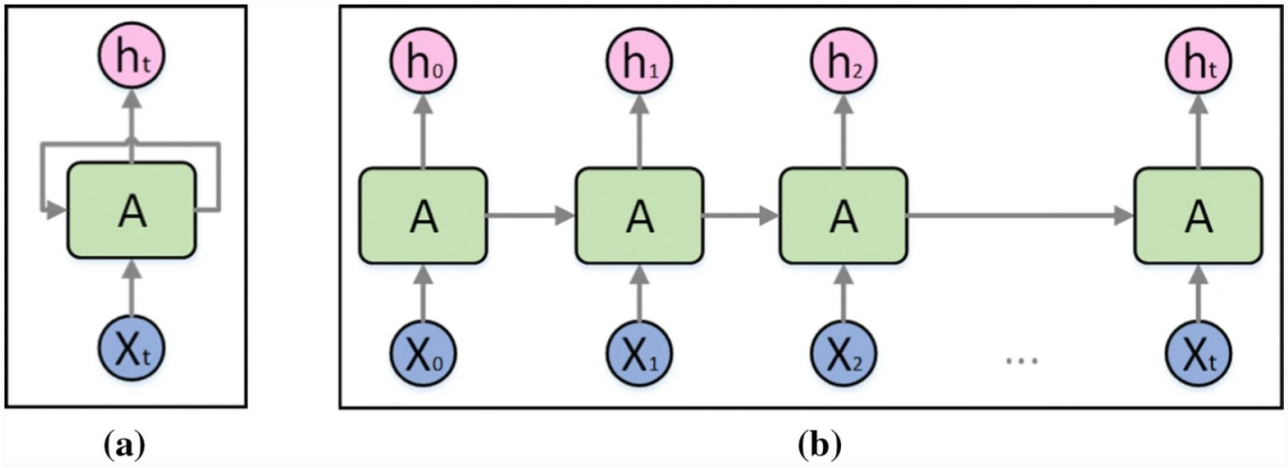

Fig 3.2 Basic architecture of Recurrent Neural Networks

Recurrent Neural Network (RNN) are a type of Neural Network. Where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

RNN have a “memory” which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

3.2 FUNCTIONAL MODULES

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import re
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import string
from nltk.corpus import stopwords
import nltk
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
from collections import Counter
from wordcloud import WordCloud
from nltk.corpus import stopwords
import nltk
from gensim.utils import simple_preprocess
from nltk.corpus import stopwords
import gensim
from sklearn.model_selection import train_test_split
import spacy
from sklearn.decomposition import NMF, LatentDirichletAllocation
import pyLDAvis
import pyLDAvis.sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from pprint import pprint
from time import time
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
from sklearn.metrics import precision_score, recall_score, accuracy_score,roc_auc_score
import warnings
warnings.filterwarnings('ignore')
from datetime import datetime
import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import keras
```

Fig 3.2.1 Importing all the Tools

```
path = '/content/gdrive/MyDrive/depression detection using tweets/dep_excel.xlsx'
dataset = pd.read_excel(path)
dataset.describe
```

```
<bound method NDFrame.describe of                                              Comments
0     So, when you ask what the two illnesses are.....    NaN
1     In addition, people with BPD quite often have ...    NaN
2     Borderline Personality Disorder, like all othe...    NaN
3     LONG ANSWER: Bipolar disease is caused by a ch...    NaN
4     Well think of bipolar as a rollar coaster you ...    NaN
...                                                  ...    ...
7140  Please contact SADAG (south African depression...    NaN
7141  It gave me severe ataxia.  (ataxia = People wi...    NaN
7142  I have. Caused muscles to tense up swelling an...    NaN
7143  Lamictal is my miracle med.Only took me 26 yea...    NaN
7144  I take Wellbutrin, topamax, and buspar (antide...    NaN

[7145 rows x 2 columns]>
```

Fig 3.2.2 Importing dataset from Drive

```
[ ] # Build the bigram and trigram models
    bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # higher threshold fewer phrases.
    trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

    # Faster way to get a sentence clubbed as a trigram/bigram
    bigram_mod = gensim.models.phrases.Phraser(bigram)
    trigram_mod = gensim.models.phrases.Phraser(trigram)

    # See trigram example
    print(trigram_mod[bigram_mod[data_words[0]]])
```

```
['so', 'when', 'you', 'ask', 'what', 'the', 'two', 'illnesses', 'are', 'theyre', 'similar', 'in', 'that', 'they', 'tend',
```

Fig 3.3.3 Cleaning Dataset 1

```
[ ]  def remove_urls(text):
         url_pattern = re.compile(r'https?://\S+|www\.\S+')
         return url_pattern.sub(r'', text)

     for i in range(len(dataset)):
       dataset.at[i,'Comments Text'] = remove_urls(dataset.iloc[i]['Comments Text'])
     dataset.describe
```

```
<bound method NDFrame.describe of                                          Comments Text
0     So, when you ask what the two illnesses are.....    NaN
1     In addition, people with BPD quite often have ...    NaN
2     Borderline Personality Disorder, like all othe...    NaN
3     LONG ANSWER: Bipolar disease is caused by a ch...    NaN
4     Well think of bipolar as a rollar coaster you ...    NaN
...                                                  ...    ...
7140  Please contact SADAG (south African depression...    NaN
7141  It gave me severe ataxia.  (ataxia = People wi...    NaN
7142  I have. Caused muscles to tense up swelling an...    NaN
7143  Lamictal is my miracle med.Only took me 26 yea...    NaN
7144  I take Wellbutrin, topamax, and buspar (antide...    NaN

[7145 rows x 2 columns]>
```

Fig 3.3.4 Cleaning Dataset 2

```
[ ]  # Convert to list
     data = dataset['Comments Text'].values.tolist()

     # Remove Emails
     data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

     # Remove new line characters
     data = [re.sub('\s+', ' ', sent) for sent in data]

     # Remove distracting single quotes
     data = [re.sub("\'", "", sent) for sent in data]

     print(data[:1])
```

```
['So, when you ask what the two illnesses are...theyre similar in that they tend to have moodiness involved, impulsivity
```

Fig 3.3.5 Cleaning Dataset 3

```
[ ]  # Define functions for stopwords, bigrams, trigrams and lemmatization

     stop_words = set(stopwords.words("english"))


     def remove_stopwords(texts):
         return [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]

     def make_bigrams(texts):
         return [bigram_mod[doc] for doc in texts]

     def make_trigrams(texts):
         return [trigram_mod[bigram_mod[doc]] for doc in texts]

     def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
         texts_out = []
         for sent in texts:
             doc = nlp(" ".join(sent))
             texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
         return texts_out
```

Fig 3.3.6 Cleaning Dataset 4

```
# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)

# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# python3 -m spacy download en
nlp = spacy.load('en', disable=['parser', 'ner'])

# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])

print(data_lemmatized[:1])
```

```
[['ask', 'illness', 'be', 'similar', 'tend', 'moodiness', 'involved', 'impulsivity', 'self', 'damaging', 'behavior',
```

Fig 3.3.7 Cleaning Dataset 5

3.3 METHODS

```
[ ]  dataset = []
     for i in range(len(data_lemmatized)):
         dataset.append(" ".join(data_lemmatized[i]))
     dataset = pd.Series(dataset)
```

Data vectorization for topic modeling

```
no_features = 15000

# NMF is able to use tf-idf
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,3), max_features=no_features)
tfidf = tfidf_vectorizer.fit_transform(dataset)
tfidf_feature_names = tfidf_vectorizer.get_feature_names()

# LDA can only use raw term counts for LDA because it is a probabilistic graphical model
tf_vectorizer = CountVectorizer(min_df=0.05,max_features=no_features)
tf = tf_vectorizer.fit_transform(dataset)
tf_feature_names = tf_vectorizer.get_feature_names()
```

```
no_topics = 2

# Run NMF
nmf = NMF(n_components=no_topics, random_state=1, alpha=.1, l1_ratio=.5,
          max_iter=10000).fit(tfidf)

# Run LDA
lda = LatentDirichletAllocation(n_components=no_topics, max_iter=10,
      learning_method='online', learning_offset=50.,random_state=0).fit(tf)
```

Fig 3.3.8 Data Vectorization

```
# Create Document — Topic Matrix
lda_output = lda.transform(tf)
# column names
topicnames = ['Topic' + str(i) for i in range(lda.n_components)]
# index names
docnames = ['Doc' + str(i) for i in range(len(dataset))]
# Make the pandas dataframe
df_document_topic = pd.DataFrame(np.round(lda_output, 2), columns=topicnames, index=docnames)
# Get dominant topic for each document
dominant_topic = np.argmax(df_document_topic.values, axis=1)
df_document_topic['dominant_topic'] = dominant_topic

df_document_topics = df_document_topic
path2 = '/content/gdrive/MyDrive/depression detection using tweets/dep_excel_2.xlsx'
dataset2 = pd.read_excel(path2)
df_document_topics.reset_index(inplace=True,drop=True)
dataset2['label'] = df_document_topics['dominant_topic']
```

```
dataset2.describe
```

```
<bound method NDFrame.describe of
0     So, when you ask what the two illnesses are......      0
1     In addition, people with BPD quite often have ...      0
2     Borderline Personality Disorder, like all othe...      0
3     LONG ANSWER: Bipolar disease is caused by a ch...      1
4     Well think of bipolar as a rollar coaster you ...      0
...                                                 ...    ...
7140  Please contact SADAG (south African depression...      1
7141  It gave me severe ataxia.  (ataxia = People wi...      1
7142  I have. Caused muscles to tense up swelling an...      1
7143  Lamictal is my miracle med.Only took me 26 yea...      1
7144  I take Wellbutrin, topamax, and buspar (antide...      1

[7145 rows x 2 columns]>
```

Fig 3.3.8.1 Dataset description

```
# Convert to list
data = dataset1['Comments Text'].values.tolist()

# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]

# Remove distracting commas
data = [re.sub(",", "", sent) for sent in data]

# Remove distracting commas
data = [sent.lower() for sent in data]

# Remove distracting dots
data = [sent.replace('.', '') for sent in data]

print(data[:1])
```

```
['so when you ask what the two illnesses aretheyre similar in that they tend to have
```

Fig 3.3.9 Describing Dataset

```
from keras.models import Sequential
from keras import layers
#from keras.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras import regularizers
max_words = 20000
max_len = 400

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(tweets)
sequences = tokenizer.texts_to_sequences(tweets)
tweets = pad_sequences(sequences, maxlen=max_len)
print(tweets)
```

```
[[   0    0    0 ...  213    2  116]
 [   4  145 1962 ...  276    2 6012]
 [   2   33 8154 ...   33    4  114]
 ...
 [   0    0    0 ...    2  290  439]
 [   0    0    0 ...  160  219  244]
 [   0    0    0 ...   10   72   58]]
```

Fig 3.3.10 Pre-processing Dataset

```
X_train, X_test, y_train, y_test = train_test_split(tweets,labels, random_state=0)
print (len(X_train),len(X_test),len(y_train),len(y_test))
```

```
5358 1787 5358 1787
```

```
model1 = Sequential()
model1.add(layers.Embedding(max_words, 40))
model1.add(layers.LSTM(40,dropout=0.5))
model1.add(layers.Dense(1,activation='sigmoid'))

model1.compile(optimizer='rmsprop',loss='binary_crossentropy', metrics=['accuracy'])

history = model1.fit(X_train, y_train, epochs=7,validation_data=(X_test, y_test))
```

```
Epoch 1/7
168/168 [==============================] - 41s 226ms/step - loss: 0.5677 - accuracy: 0.6926
Epoch 2/7
168/168 [==============================] - 37s 223ms/step - loss: 0.3296 - accuracy: 0.8686
Epoch 3/7
168/168 [==============================] - 39s 233ms/step - loss: 0.2465 - accuracy: 0.9048
Epoch 4/7
168/168 [==============================] - 38s 226ms/step - loss: 0.1954 - accuracy: 0.9293
Epoch 5/7
168/168 [==============================] - 38s 227ms/step - loss: 0.1709 - accuracy: 0.9343
Epoch 6/7
168/168 [==============================] - 38s 224ms/step - loss: 0.1398 - accuracy: 0.9500
Epoch 7/7
168/168 [==============================] - 38s 224ms/step - loss: 0.1379 - accuracy: 0.9520
```

Fig 3.3.11 Training Model 1

```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test, np.around(y_pred, decimals=0))
import seaborn as sns
conf_matrix = pd.DataFrame(matrix, index = ['Not Depression/Anxiety','Anxiety/Depression'],
#Normalizing
conf_matrix = conf_matrix.astype('float')
conf_matrix.sum(axis=1)[:, np.newaxis]
plt.figure(figsize = (15,15))
sns.heatmap(conf_matrix, annot=True, annot_kws={"size": 15})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa63d46e610>
```

Fig 3.3.11.1 Plotting matrix

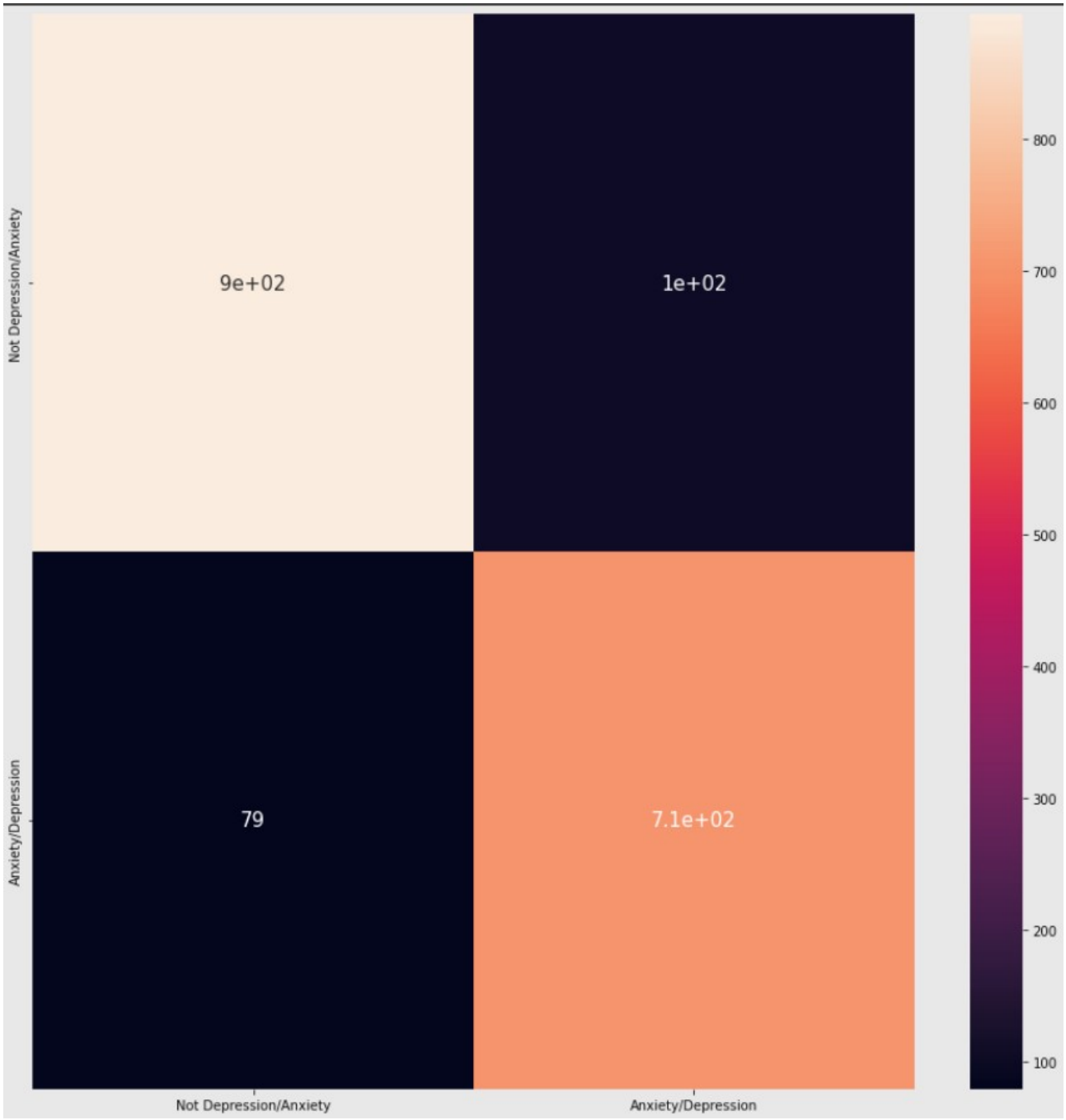

Fig 3.3.12 Matrix Classification

```
[ ]  test_loss, test_acc = model2.evaluate(X_test,  y_test, verbose=2)
     print('Model accuracy: ',test_acc)

     56/56 - 4s - loss: 0.2684 - accuracy: 0.9049 - 4s/epoch - 70ms/step
     Model accuracy:  0.904868483543396
```

Fig 3.3.13 Model1 Accuracy

```
[ ] X_train, X_test, y_train, y_test = train_test_split(tweets,labels, random_state=0)
    print (len(X_train),len(X_test),len(y_train),len(y_test))

    5358 1787 5358 1787
```

```
model2 = Sequential()
model2.add(layers.Embedding(max_words, 40))
model2.add(layers.LSTM(40,dropout=0.5,return_sequences=True))
model2.add(layers.LSTM(40,dropout=0.5))
model2.add(layers.Dense(1,activation='sigmoid'))

model2.compile(optimizer='rmsprop',loss='binary_crossentropy', metrics=['accuracy'])

history = model2.fit(X_train, y_train, epochs=5,validation_data=(X_test, y_test))
```

```
Epoch 1/5
168/168 [==============================] - 79s 442ms/step - loss: 0.5422 - accuracy: 0.7159
Epoch 2/5
168/168 [==============================] - 74s 439ms/step - loss: 0.3283 - accuracy: 0.8654
Epoch 3/5
168/168 [==============================] - 74s 439ms/step - loss: 0.2445 - accuracy: 0.9057
Epoch 4/5
168/168 [==============================] - 73s 436ms/step - loss: 0.1930 - accuracy: 0.9300
Epoch 5/5
168/168 [==============================] - 73s 437ms/step - loss: 0.1650 - accuracy: 0.9365
```

Fig 3.3.14 Training Model 2

```
[ ] matrix = confusion_matrix(y_test, np.around(y_pred, decimals=0))
    conf_matrix = pd.DataFrame(matrix, index = ['Not Depression/Anxiety','Anxiety/Depression'],
    #Normalizing
    conf_matrix = conf_matrix.astype('float')
    conf_matrix.sum(axis=1)[:, np.newaxis]
    plt.figure(figsize = (15,15))
    sns.heatmap(conf_matrix, annot=True, annot_kws={"size": 15})

    <matplotlib.axes._subplots.AxesSubplot at 0x7fa63ce44d10>
```

Fig 3.3.14.1 Plotting Matrix2

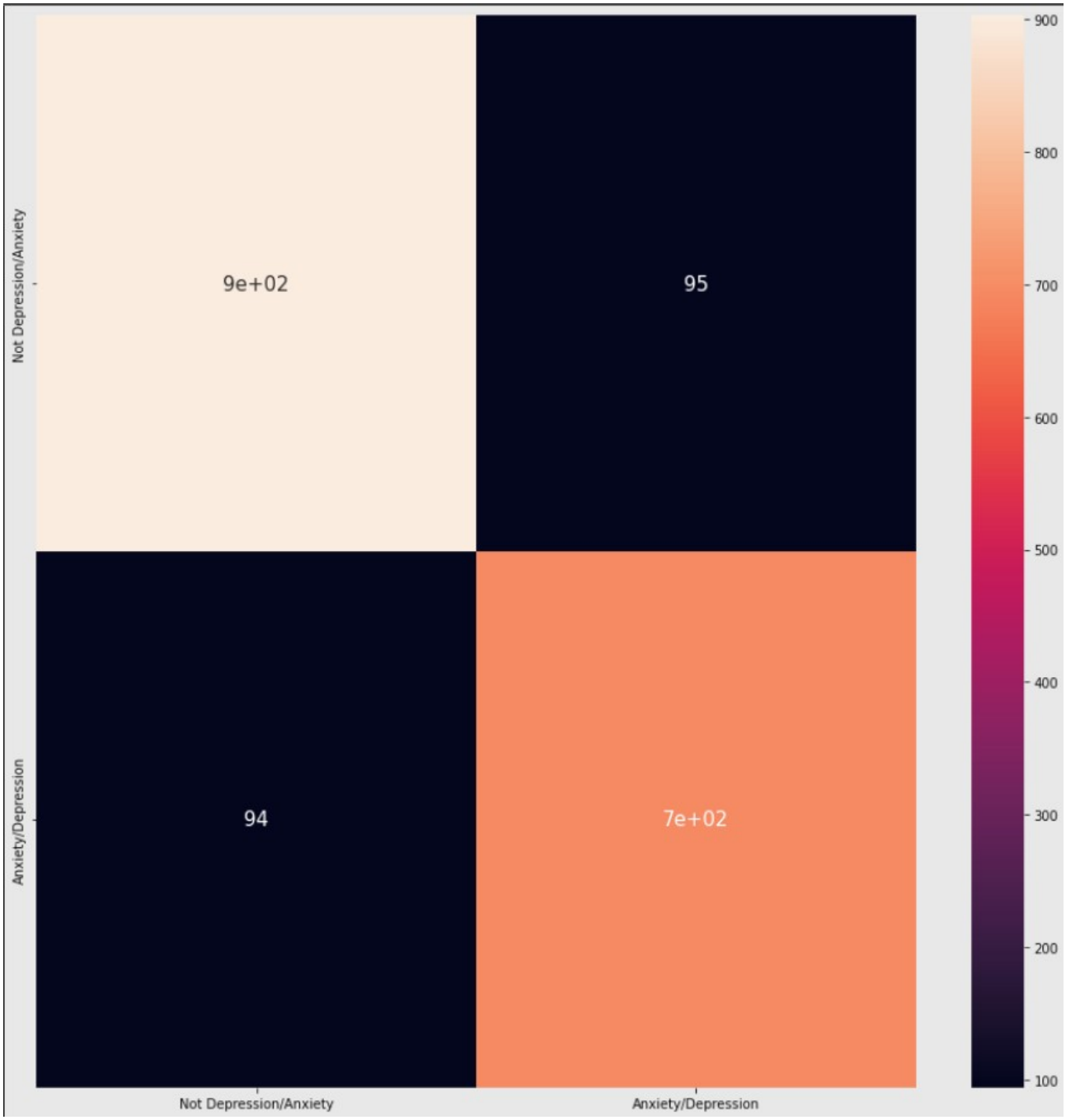

Fig 3.3.15 Matrix Classification

```
test_loss, test_acc = model2.evaluate(X_test,  y_test, verbose=2)
print('Model accuracy: ',test_acc)
```

```
56/56 - 4s - loss: 0.2684 - accuracy: 0.9049 - 4s/epoch - 70ms/step
Model accuracy:  0.904868483543396
```

Fig 3.3.16 Model2 Accuracy

```
model3 = Sequential()
model3.add(layers.Embedding(max_words, 40))
model3.add(layers.Bidirectional(layers.LSTM(40,dropout=0.5)))
model3.add(layers.Dense(1,activation='sigmoid'))

model3.compile(optimizer='rmsprop',loss='binary_crossentropy', metrics=['accuracy'])

history = model3.fit(X_train, y_train, epochs=8,validation_data=(X_test, y_test))
```

```
Epoch 1/8
168/168 [==============================] - 69s 384ms/step - loss: 0.6016 - accuracy: 0.6693
Epoch 2/8
168/168 [==============================] - 64s 383ms/step - loss: 0.3721 - accuracy: 0.8485
Epoch 3/8
168/168 [==============================] - 65s 385ms/step - loss: 0.2701 - accuracy: 0.8973
Epoch 4/8
168/168 [==============================] - 64s 383ms/step - loss: 0.2198 - accuracy: 0.9192
Epoch 5/8
168/168 [==============================] - 65s 387ms/step - loss: 0.1943 - accuracy: 0.9304
Epoch 6/8
168/168 [==============================] - 64s 381ms/step - loss: 0.1605 - accuracy: 0.9440
Epoch 7/8
168/168 [==============================] - 62s 368ms/step - loss: 0.1445 - accuracy: 0.9496
Epoch 8/8
168/168 [==============================] - 62s 366ms/step - loss: 0.1241 - accuracy: 0.9567
```

Fig 3.3.17 Training Model 3

```
[ ] matrix = confusion_matrix(y_test, np.around(y_pred, decimals=0))
    conf_matrix = pd.DataFrame(matrix, index = ['Not Depression/Anxiety','Anxiety/Depression'],
    #Normalizing
    conf_matrix = conf_matrix.astype('float')
    conf_matrix.sum(axis=1)[:, np.newaxis]
    plt.figure(figsize = (15,15))
    sns.heatmap(conf_matrix, annot=True, annot_kws={"size": 15})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa636aaff90>
```

Fig 3.3.17.1 plotting matrix3

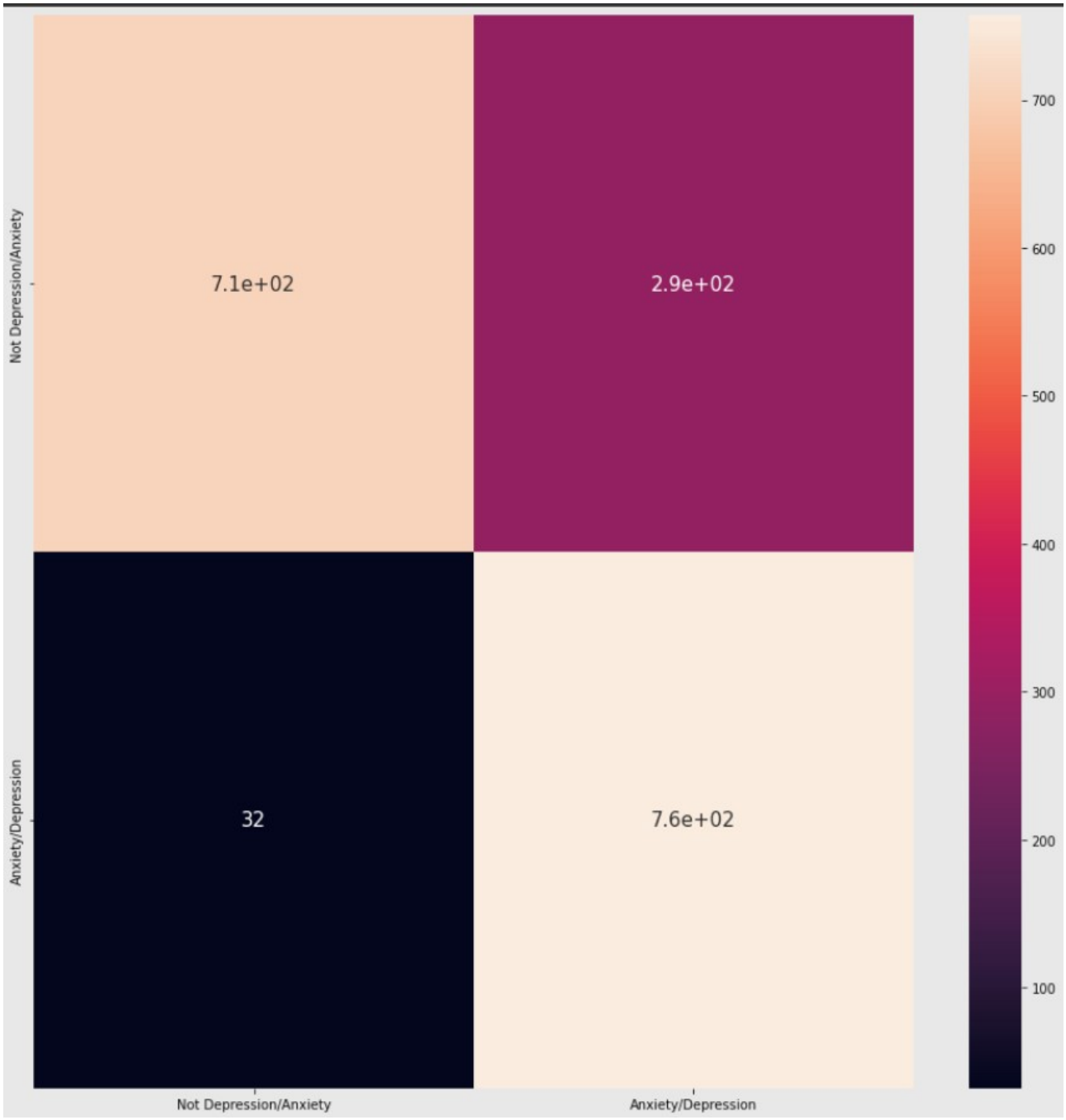

Fig 3.3.18 Matrix Classification

```
for dirname, _, filenames in os.walk(tPath):
    for filename in filenames:
        if filename!='0314_1.csv':
            temp = pd.read_csv(os.path.join(dirname, filename))
            tweets = pd.concat([tweets, temp], ignore_index=True)
```

```
[ ] tweets.shape

    (25425, 12)
```

```
tweets = pd.read_csv('/content/gdrive/MyDrive/depression detection using tweets/tweets_data/0314_1.csv')
tweets.describe
```

```
<bound method NDFrame.describe of                    date  ...
0    2020-03-14 23:55:21  ...  https://twitter.com/AlbertoxVazquez/status/123...
1    2020-03-14 23:54:45  ...  https://twitter.com/TheOGKennedy/status/123897...
2    2020-03-14 23:45:58  ...  https://twitter.com/megmarie5/status/123897468...
3    2020-03-14 23:43:10  ...  https://twitter.com/BoozyBillsBabe/status/1238...
4    2020-03-14 23:42:39  ...  https://twitter.com/megmarie5/status/123897385...
..                   ...  ...                                                ...
283  2020-03-14 00:14:34  ...  https://twitter.com/jw8c/status/12386194940981...
284  2020-03-14 00:14:19  ...  https://twitter.com/Lumae_tinkk/status/1238619...
285  2020-03-14 00:08:46  ...  https://twitter.com/dpink_dpanda/status/123861...
286  2020-03-14 00:07:19  ...  https://twitter.com/Anton10937175/status/12386...
287  2020-03-14 00:04:13  ...  https://twitter.com/intuitiveshoob/status/1238...

[288 rows x 12 columns]>
```

Fig. 3.3.19 Tweets description

```
[ ] test = np.array(['I feel stress, sadness and anxiety - just want to sleep until the lockdown ends'])
    test_sequence = tokenizer.texts_to_sequences(test)
    test_sequence = pad_sequences(test_sequence, maxlen=max_len)
    test_prediction = model3.predict(test_sequence)
    if np.around(test_prediction, decimals=0)[0][0] == 1.0:
        print('The model predicted depressive/anxious language')
    else:
        print("The model predicted other type of language")

    The model predicted depressive/anxious language
```

Fig 3.3.20 Depressive Tweet Result

```
test = np.array(['I just want to sleep until the lockdown ends'])
test_sequence = tokenizer.texts_to_sequences(test)
test_sequence = pad_sequences(test_sequence, maxlen=max_len)
test_prediction = model3.predict(test_sequence)
if np.around(test_prediction, decimals=0)[0][0] == 1.0:
    print('The model predicted depressive/anxious language')
else:
    print("The model predicted other type of language")
```

```
The model predicted other type of language
```

Fig 3.3.21 Non-Depressive Tweet Result

```
for i in range(10):
    print(tweets_dataset.iloc[i*2]['text'])
    print('\n')
```

```
Damn just remembered this front bottoms show is about to cure my depression

it's making me anxious lol

Have some wine . You'll feel less anxious.

Can't wait to tell my kids about the toilet paper depression of 2020

<—— anxious bunny https://twitter.com/GlennonDoyle/status/1238449159168053250 …

And here we are the Great Depression of toilet paper

Depression is knocking at the door #Tougaloo_RYS20 #Jackson_RYS20https://twitter.com/UnderRatedTim/status/1238589018662715

Take me back! Kind of been in a slight depression since being home. With all this virus crap, it's making it worse! Kind of

Take me back! Kind of been in a slight depression since being home. With all this virus crap, it's making it worse! Kind of

Seasonal depression is a fucking bitch, depression in general is a bitch
```

Fig 3.3.21 printing tweets

CHAPTER 4

OBSERVATIONS AND RESULTS

4.1 TRAINING

We'll explore how LDA and NMF can create the topics and depending on the outcomes we'll select the proper one for this project. Essentially we're looking for focused topics, otherwise the purpose of this project won't be reached.

The crucial difference between both models is that LDA adds a Dirichlet prior on top of the data generating process, meaning NMF qualitatively leads to worse mixtures, which could affect our dataset's topic quality.

Regarding the library we'll be using: Scikit-Learn - the reasons are more than obvious, even when Gensim has more capabilities, it's also more complex and much more slower - we're looking to keep the things as simpler as possible and get results as quick as possible.

The outcome of this stage will be the original dataframe with its labels: 1 for depression/anxiety comments and 0 for other type of comments.

```
def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print("Topic %d:" % (topic_idx))
        print(", ".join([feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1]]))

no_top_words = 25
print('NMF')
display_topics(nmf, tfidf_feature_names, no_top_words)
print('LDA')
display_topics(lda, tf_feature_names, no_top_words)
```

```
NMF
Topic 0:
be, go, help, take, feel, get, know, time, med, make, try, thing, day, work, would, people, need, have,
Topic 1:
anxiety, depression, depression anxiety, bipolar, anxiety depression, take, bipolar depression anxiety,
LDA
Topic 0:
be, feel, go, know, day, people, get, time, think, say, life, want, thing, have, make, would, struggle,
Topic 1:
anxiety, take, help, med, work, depression, also, bipolar, get, need, find, medication, try, doctor, ye
```

Fig 4.1 Model training

4.2 TESTING

```
model1 = Sequential()
model1.add(layers.Embedding(max_words, 40))
model1.add(layers.LSTM(40,dropout=0.5))
model1.add(layers.Dense(1,activation='sigmoid'))

model1.compile(optimizer='rmsprop',loss='binary_crossentropy', metrics=['accuracy'])

history = model1.fit(X_train, y_train, epochs=7,validation_data=(X_test, y_test))
```

```
Epoch 1/7
168/168 [==============================] - 41s 227ms/step - loss: 0.5734 - accuracy: 0.6915 - val_loss: 0.3896 - val_accuracy: 0.8327
Epoch 2/7
168/168 [==============================] - 37s 223ms/step - loss: 0.3340 - accuracy: 0.8667 - val_loss: 0.2727 - val_accuracy: 0.8870
Epoch 3/7
168/168 [==============================] - 37s 220ms/step - loss: 0.2596 - accuracy: 0.9044 - val_loss: 0.2519 - val_accuracy: 0.8920
Epoch 4/7
168/168 [==============================] - 37s 220ms/step - loss: 0.2229 - accuracy: 0.9175 - val_loss: 0.3677 - val_accuracy: 0.8651
Epoch 5/7
168/168 [==============================] - 37s 221ms/step - loss: 0.1882 - accuracy: 0.9330 - val_loss: 0.2507 - val_accuracy: 0.8993
Epoch 6/7
168/168 [==============================] - 38s 224ms/step - loss: 0.1661 - accuracy: 0.9408 - val_loss: 0.2437 - val_accuracy: 0.8987
Epoch 7/7
168/168 [==============================] - 37s 221ms/step - loss: 0.1402 - accuracy: 0.9505 - val_loss: 0.3910 - val_accuracy: 0.8685
```

Fig 4.2 Model1 Testing

Model accuracy: 95.05%

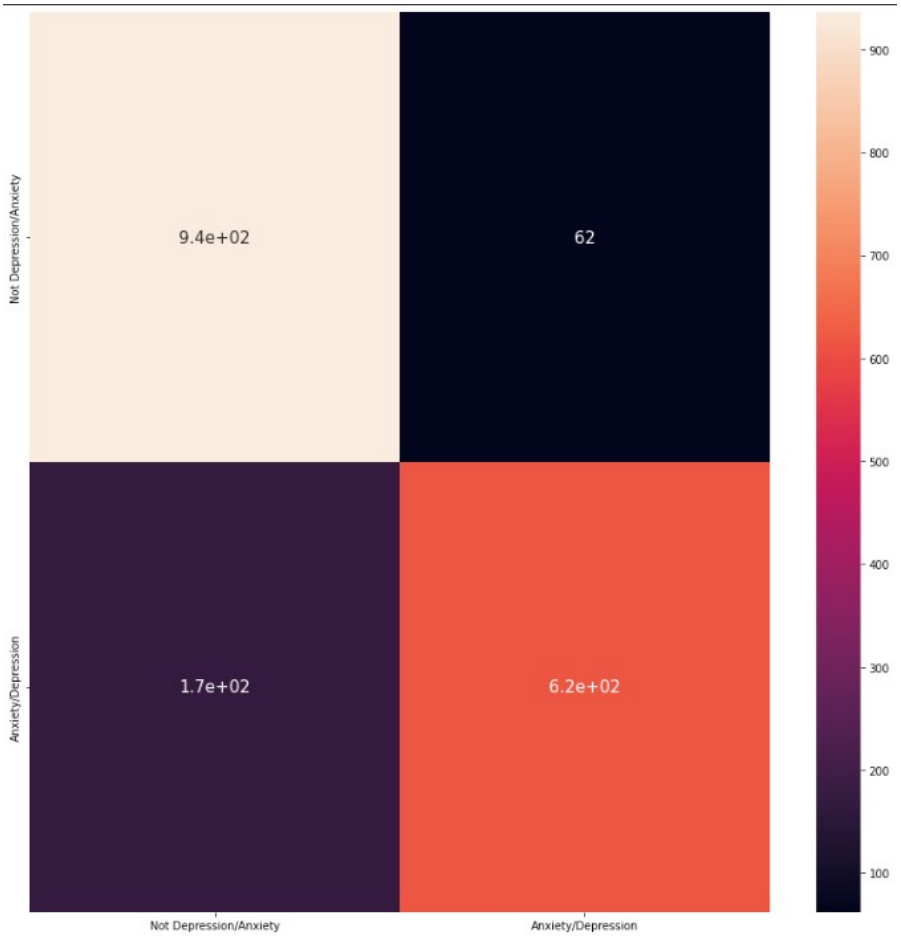

Fig 4.3 Model1 Matrix Output

```
model2 = Sequential()
model2.add(layers.Embedding(max_words, 40))
model2.add(layers.LSTM(40,dropout=0.5,return_sequences=True))
model2.add(layers.LSTM(40,dropout=0.5))
model2.add(layers.Dense(1,activation='sigmoid'))

model2.compile(optimizer='rmsprop',loss='binary_crossentropy', metrics=['accuracy'])

history = model2.fit(X_train, y_train, epochs=5,validation_data=(X_test, y_test))
```

```
Epoch 1/5
168/168 [==============================] - 79s 442ms/step - loss: 0.5422 - accuracy: 0.7159 -
Epoch 2/5
168/168 [==============================] - 74s 439ms/step - loss: 0.3283 - accuracy: 0.8654 -
Epoch 3/5
168/168 [==============================] - 74s 439ms/step - loss: 0.2445 - accuracy: 0.9057 -
Epoch 4/5
168/168 [==============================] - 73s 436ms/step - loss: 0.1930 - accuracy: 0.9300 -
Epoch 5/5
168/168 [==============================] - 73s 437ms/step - loss: 0.1650 - accuracy: 0.9365 -
```

Fig 4.4 Model2 Testing

Model 2 Accuracy: 93.93%

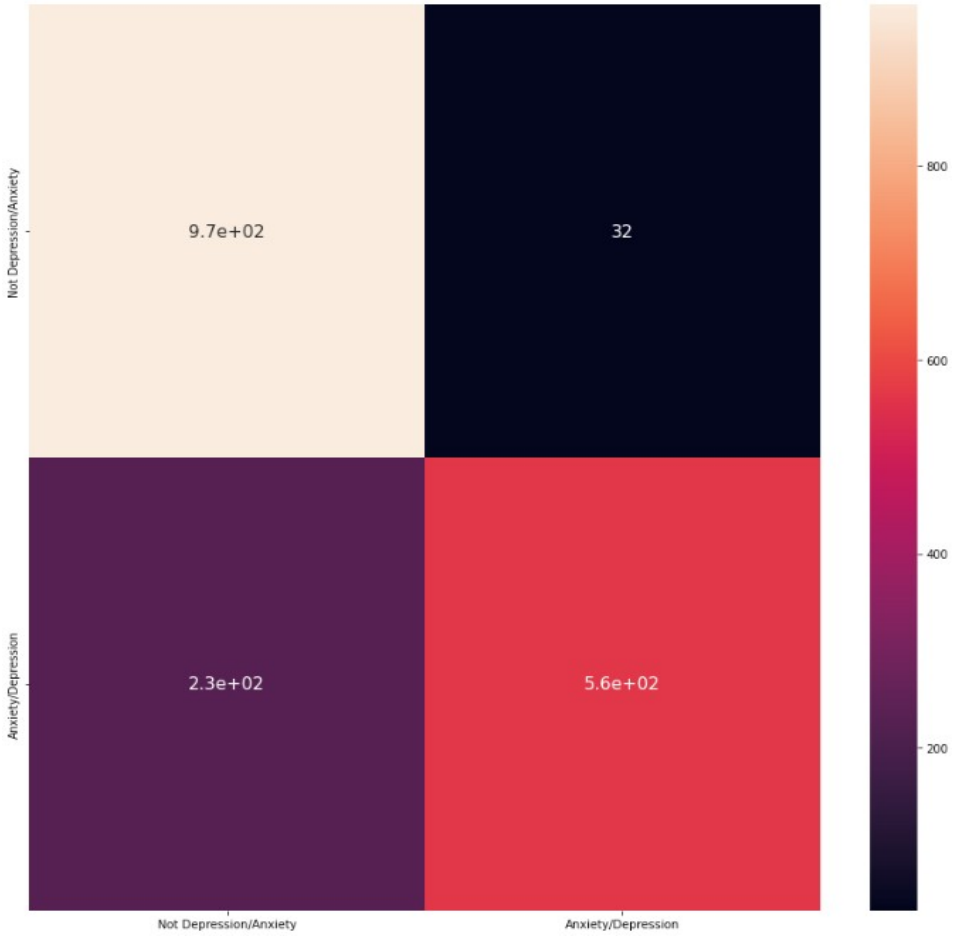

Fig 4.5 Model2 Matrix output

```
model3 = Sequential()
model3.add(layers.Embedding(max_words, 40))
model3.add(layers.Bidirectional(layers.LSTM(40,dropout=0.5)))
model3.add(layers.Dense(1,activation='sigmoid'))

model3.compile(optimizer='rmsprop',loss='binary_crossentropy', metrics=['accuracy'])

history = model3.fit(X_train, y_train, epochs=8,validation_data=(X_test, y_test))
```

```
Epoch 1/8
168/168 [==============================] - 69s 384ms/step - loss: 0.6016 - accuracy: 0.6693
Epoch 2/8
168/168 [==============================] - 64s 383ms/step - loss: 0.3721 - accuracy: 0.8485
Epoch 3/8
168/168 [==============================] - 65s 385ms/step - loss: 0.2701 - accuracy: 0.8973
Epoch 4/8
168/168 [==============================] - 64s 383ms/step - loss: 0.2198 - accuracy: 0.9192
Epoch 5/8
168/168 [==============================] - 65s 387ms/step - loss: 0.1943 - accuracy: 0.9304
Epoch 6/8
168/168 [==============================] - 64s 381ms/step - loss: 0.1605 - accuracy: 0.9440
Epoch 7/8
168/168 [==============================] - 62s 368ms/step - loss: 0.1445 - accuracy: 0.9496
Epoch 8/8
168/168 [==============================] - 62s 366ms/step - loss: 0.1241 - accuracy: 0.9567
```

Fig 4.4 Model 3 Testing

Model 3 Accuracy: 95.46%

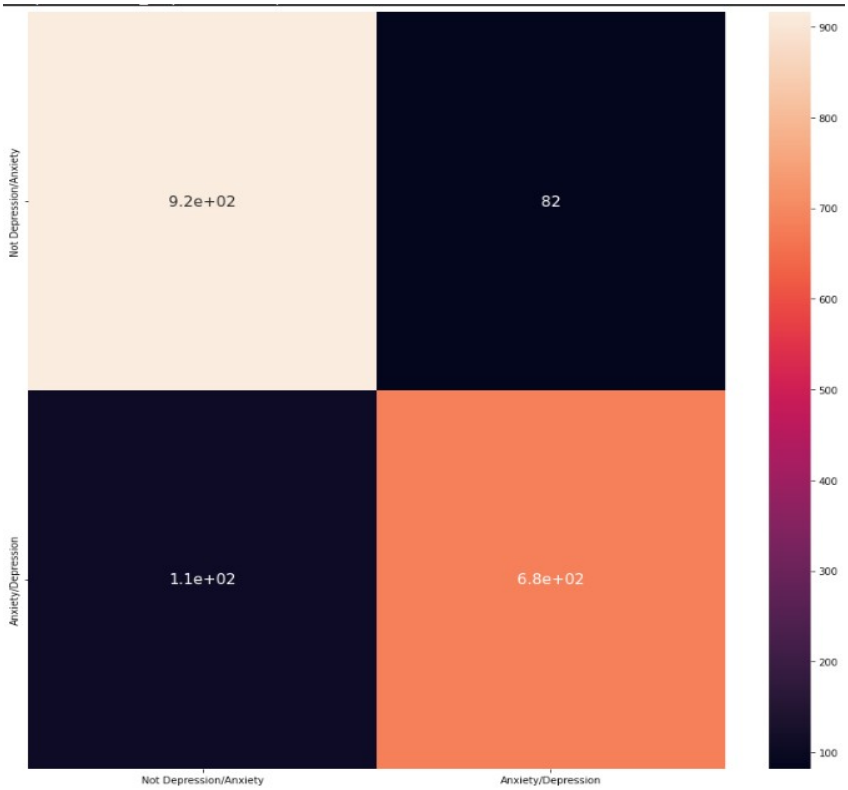

Fig 4.5 Model3 Matrix output

4.3 RESULTS

```
[ ] test = np.array(['I feel stress, sadness and anxiety - just want to sleep until the lockdown ends'])
    test_sequence = tokenizer.texts_to_sequences(test)
    test_sequence = pad_sequences(test_sequence, maxlen=max_len)
    test_prediction = model3.predict(test_sequence)
    if np.around(test_prediction, decimals=0)[0][0] == 1.0:
        print('The model predicted depressive/anxious language')
    else:
        print("The model predicted other type of language")

    The model predicted depressive/anxious language
```

Fig 4.6 Depressive text output

```
[ ] test = np.array(['I just want to sleep until the lockdown ends'])
    test_sequence = tokenizer.texts_to_sequences(test)
    test_sequence = pad_sequences(test_sequence, maxlen=max_len)
    test_prediction = model3.predict(test_sequence)
    if np.around(test_prediction, decimals=0)[0][0] == 1.0:
        print('The model predicted depressive/anxious language')
    else:
        print("The model predicted other type of language")

    The model predicted other type of language
```

Fig 4.7 Non- Depressive text output

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 CONCLUSION

We went through several steps, including Natural Language Processing for clustering and classification. The main point to highlight is that Deep Learning is an approach for when you have tons of data available, otherwise most of the times regular/classic ML models would perform better.

As it is showcased from the project, Deep Learning this time delivered an acceptable score, but for any other task such as in medical field, definitely you will need to get more data to increase the recall score.

5.2 FUTURE ENHANCEMENTS

- Making a Twitter Bot to autonomously reply to Depressive tweets and help them receive a helpline.
- Through this project, we may create a bot which will help to analyse the emotion of a person as soon as they tweet.
- We can enhance this project which may take videos and images as the data-set.
- We may also implement this model in different social media platform say Facebook and Instagram.

CHAPTER 6

REFERENCES

Links:

Dataset:

1. https://www.kaggle.com/sergiovirahonda/depression-and-anxiety-comments
2. https://www.kaggle.com/sergiovirahonda/depression-anxiety-tweets
3. https://link.springer.com/article/10.1007/s13755-018-0046-0
4. https://en.wikipedia.org/wiki/Recurrent_neural_network
5. https://en.wikipedia.org/wiki/Long_short-term_memory

Books:

1. Introduction to Machine Learning with Python: A Guide for Data Scientists, by Sarah Guido & Andreas C. Mueller.
2. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, by Aurelien Geron.
3. Mathematics for Machine Learning**,** by Marc Peter Deisenroth.