MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM

Department of Computer Applications

Mini Project Report

DETECTION OF DEPRESSION FROM TEXTUAL DATA IN SOCIAL MEDIA USING MACHINE LEARNING

Done by

Malavika P

Reg No:MAC23MCA-2037

Under the guidance of
Prof. Biju Skaria

2023-2025

MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM

CERTIFICATE

Detection of Depression from Textual data in social media using Machine learning

Certified that this is the bonafide record of project work done by

Malavika P
Reg No: MAC23MCA-2037

during the third semester, in partial fulfilment of requirements for award of the degree

Master of Computer Applications

of

APJ Abdul Kalam Technological University, Thiruvananthapuram

Faculty Guide   **Head of the Department**

Prof. Biju Skaria   Prof. Biju Skaria

Project Coordinator   **Internal Examiners**

Prof. Sonia Abraham

ACKNOWLEDGEMENT

ABSTRACT

The "Depression Detection from Social Media Using Machine Learning" project addresses the urgent global need for identifying undiagnosed cases of depression by analysing textual data from social media platforms. Depression, a leading cause of suicide worldwide, often goes undetected and untreated. Leveraging social media data provides a promising pathway to identifying at-risk individuals earlier, allowing for timely intervention. This study utilizes advanced feature extraction techniques and machine learning classifiers to detect depressive content with high accuracy, ultimately aiming to create a user-friendly and effective tool for both individuals and mental health professionals.

Previous research has demonstrated the potential of various feature extraction and classification techniques in detecting depression. In particular, one study achieved 91% accuracy by integrating Linguistic Inquiry and Word Count (LIWC), Latent Dirichlet Allocation (LDA), and bigrams with a Multilayer Perceptron (MLP) model on Reddit posts, proving effective for identifying depressive language patterns. Another study demonstrated that Bag-of-Words and N-grams, when combined with Logistic Regression (LR) and Random Forest (RF), could identify depressive social media posts with over 90% accuracy, without depending on explicit keywords such as "depression." A third study underscored the effectiveness of Decision Trees in detecting depression in Facebook comments, achieving 60-80% accuracy and tracking depressive trends over time.

In this project, a combination of N-grams and Term Frequency-Inverse Document Frequency (TF-IDF) is employed for feature extraction, which highlights various facets of text data, including emotional processes, linguistic style, and temporal patterns. The classification model incorporates ensemble learning, utilizing MLP and RF for robust predictions, while Decision Trees add complementary insights, optimizing the system's detection capabilities. This approach not only identifies depressive content but also offers recommendations for support, thus addressing both detection and intervention in a holistic manner.

The system is accessible through a user-friendly web interface, built with the Flask framework, enabling real-time analysis of social media posts or comments. Users can input textual content to receive instant feedback on whether the content may indicate depressive tendencies. For those identified as potentially at risk, the system provides support suggestions, thus extending beyond detection to deliver actionable insights for users seeking help or for mental health professionals tracking their patients' progress.

This model was trained and evaluated on a Kaggle dataset curated by B.B.eye, comprising 10,314 tweets focused on depression analysis. This dataset, with its depth and specificity, allows the system to learn nuanced patterns associated with depressive language and behaviours. By analysing text on social media, this system can facilitate timely support for those at risk, contributing to broader mental health awareness and intervention efforts on a global scale.

LIST OF TABLES

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

Depression is a leading cause of suicide worldwide, with severe consequences for individuals and society. Many cases remain undiagnosed and untreated, often due to social stigma, lack of awareness, and limited access to mental health services. According to the World Health Organization (WHO), depression affects over 300 million people globally, accounting for more than two-thirds of suicides each year. These barriers to treatment highlight the need for alternative approaches to detection and intervention.

Social media platforms have emerged as valuable resources for studying mental health patterns, with platforms like Twitter, Facebook, and Reddit allowing people to freely express their thoughts and feelings. Studies show that individuals with major depressive disorder often share subtle indicators of their condition through language and sentiment in their posts, which can be analyzed to predict the likelihood of depression and offer a potential pathway for early intervention. By examining emotional expression, linguistic style, and temporal indicators, researchers can identify patterns unique to depressive behaviour, providing insights into how individuals manage their symptoms over time.

A recent survey indicates that many people, especially teenagers and young adults, turn to social media to express symptoms of depression and anxiety. However, existing research often relies on specific keywords such as "depression" and "diagnose" when analysing social media data, limiting effectiveness as users may avoid these terms due to stigma. Social media, then, becomes a space for informal expression rather than direct admission of mental health struggles.

This project seeks to address these challenges by developing a machine learning model capable of detecting depressive tendencies in social media posts without relying on explicit keywords. Through advanced text analysis, the project aims to identify nuanced patterns associated with depression and provide early detection, bridging the gap between traditional mental health services and the informal support offered on social media. By making mental health support more accessible and responsive in today's digital age, this project hopes to contribute meaningfully to early diagnosis and intervention, potentially improving the lives of individuals affected by untreated depression.

2. SUPPORTING LITERATURE

2.1. LITERATURE REVIEW

Paper 1: [1] Tadesse MM, Lin H, Xu B, Yang L. "Detection of depression-related posts in reddit social media forum." Ieee Access. 2019 Apr 4.

The paper titled "Detection of Depression-Related Posts in Reddit Social Media Forum" by Michael M. Tadesse et al. presents a study on identifying depression-related content on Reddit using natural language processing (NLP) and machine learning techniques. The authors address the growing concern of depression as a major contributor to global disability and a significant cause of suicide. They aim to examine Reddit users' posts to detect factors that may reveal depressive attitudes.

The researchers use natural language processing (NLP) and machine learning techniques to train and test their method. They identify words commonly used by people with depression and analyse their connection to mental health. Methods like N-grams (word combinations), LIWC (emotional and psychological content analysis), and LDA (topic identification) help them understand how language on social media can indicate depression.

The study compares single and combined feature sets using various text classification methods like including Logistic Regression, Support Vector Machine, Random Forest, Adaptive Boosting, and Multilayer Perceptron. Results show that proper feature selection and combination lead to higher performance. The best depression detection performance (91% accuracy, 0.93 F1 score) is achieved by combining LIWC, LDA, and bigram features using a Multilayer Perceptron neural network. This paper helps improve detecting depression on social media, like Reddit, by combining language analysis and machine learning, which could lead to earlier support for those at risk. Table 2.1 shows the summary of paper 1.

Title of the paper	Detection of Depression-Related Posts in Reddit social media Forum
Area of work	Mental health detection in social media using NLP and machine learning.
Dataset	The dataset was built by Inna Pirina et al. and consists of a list of depressed and non-depressed users.
Methodology / Strategy	Data pre-processing, Feature extraction, Text classification using various machine learning algorithms.
Algorithm	LR, SVM, RF, AdaBoost, MLP
Result/Accuracy	91% accuracy and 0.93 F1 score using LIWC+LDA+bigram features with MLP.

Table 2.1 Summary of paper 1

Paper 2: [2] Chiong R, Budhi GS, Dhakal S, Chiong F. "A textual-based featuring approach for depression detection using machine learning classifiers and social media texts." Computers in Biology and Medicine. 2021 Aug 1

The paper "A textual-based featuring approach for depression detection using machine learning classifiers and social media texts" by Raymond Chiong et al. introduces a new way to detect depression by analyzing social media posts with machine learning. The paper focus on the problem of undiagnosed depression and its serious consequences. They propose a method that uses text-based features and different machine learning classifiers to find signs of depression in social media posts, even when specific words like "depression" or "diagnosis" are not used.

The study uses various text preprocessing and feature extraction methods, such as bag-of-words and n-grams. The authors test their approach on multiple datasets from different social media platforms and compare the performance of individual and combined machine learning classifiers. They also tackle the issue of imbalanced datasets with dynamic sampling techniques. The results show that their method effectively detects depression across different social media sources, even when trained on one platform and tested on others. This paper helps advance machine learning, natural language processing, and mental health research by providing a method to detect early signs of depression using social media data. Table 2.2 shows the summary of paper 2.

Title of the paper	A textual-based featuring approach for depression detection using machine learning classifiers and social media texts
Area of work	Depression detection using social media data and machine learning.
Dataset	**Shen et al.(2017):** Depression dataset, **Eye BB (2020)**: Focuses on depression analysis, **Tanwar R (2020)**: Based on the diary of a 17-year-old girl, Victoria, who committed suicide due to depression, **Komati N (2020)**: Includes posts from Reddit communities r/SuicideWatch and r/depression, **Virahonda S (2020)**: Consists of comments related to depression and anxiety
Methodology / Strategy	Text preprocessing, Feature extraction using bag-of-words and n-grams, Training and testing ML models on Twitter datasets, Testing trained models on non-Twitter depression datasets, Experimenting with sampling techniques for imbalanced data.
Algorithm	LR, SVM, MLP, DT, RF, AdaBoost, Bagging, Gradient Boosting
Result/Accuracy	LR (92.61% accuracy on Eye's dataset), RF (balanced accuracy for depression/non-depression classes)

Table 2.2 Summary of paper 2

Paper 3: [3] Islam MR, Kabir MA, Ahmed A, Kamal AR, Wang H, Ulhaq A. "Depression detection from social network data using machine learning techniques." Health information science and systems. 2018 Dec

The paper titled "Depression detection from social network data using machine learning techniques" by Islam et al. presents an innovative approach to detect depression through analysis of social media data, specifically Facebook comments. The authors explore how social networks can be used to monitor and improve mental health and uses machine learning to study the language and emotions in Facebook comments to find signs of depression.

The study utilizes a dataset of 7,145 Facebook comments, categorized into depression-indicative and non-depression-indicative groups. The authors extract psycholinguistic features using the Linguistic Inquiry and Word Count (LIWC) software, focusing on emotional processes, temporal processes, and linguistic style. They then apply and compare four machine learning classifiers: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Trees (DT), and Ensemble methods.

The researchers conduct experiments to evaluate the performance of these classifiers across different feature sets. Their results indicate that the Decision Tree classifier generally outperforms other methods, achieving the highest accuracy in detecting depression-indicative comments. The study also includes a time series analysis, revealing that depression-indicative comments are more prevalent during AM hours.

Overall, this paper contributes to the field of mental health informatics by demonstrating the potential of machine learning techniques in detecting depression through social media data. The findings suggest that automated analysis of social network content could be a valuable tool for early detection and intervention of depression, potentially improving mental health outcomes. Table2.3 shows the summary of paper 3.

Title of the paper	Depression detection from social network data using machine learning techniques.
Area of work	Mental Health, Social Network Analysis.
Dataset	Publicly available Facebook data (from bipolar, depression, and anxiety Facebook pages).
Methodology / Strategy	Collected Facebook comments data, Used LIWC software to extract linguistic features, Applied machine learning classifiers.
Algorithm	DT, KNN, SVM, Ensemble Methods.
Result/Accuracy	Decision Tree performed best overall Accuracy between 60-80% for different features

Table 2.3 Summary of paper 3

2.2. LITERATURE REVIEW SUMMARY

Paper 1	The paper "Detection of Depression-Related Posts in Reddit Social Media Forum" explores detecting depression on Reddit using natural language processing (NLP) and machine learning. It focuses on analyzing language patterns of depressed users, examining features such as n-grams, LIWC categories, and LDA topic modeling. Various classifiers, including LR, SVM, RF, AdaBoost, and MLP, were tested. The SVM classifier with bigrams achieved 80% accuracy, while combining LIWC, LDA, and bigram features with a multilayer perceptron reached 91% accuracy and a 0.93 F1 score. The study highlights that effective feature selection and integration can enhance depression detection in social media text.
Paper 2	The paper "A Textual-Based Featuring Approach for Depression Detection Using Machine Learning Classifiers and Social Media Texts" by Raymond Chiong et al. presents a method for detecting depression through social media posts without relying on specific keywords. The approach uses textual features extracted via bag-of-words, and n-grams, and evaluates various classifiers, including LR, SVM, MLP, and ensembles. Experiments on Twitter and non-Twitter datasets demonstrate that the method effectively identifies depression with over 90% accuracy. Notably, LR and RF performed best, and dynamic under sampling improved results on imbalanced datasets. The study shows that generalized machine learning techniques can detect depression in social media texts, regardless of keyword usage.
Paper 3	The paper explores detecting depression from Facebook comments using machine learning techniques. It synthesizes emotion detection literature and identifies four feature types: emotional process, temporal process, linguistic style, and a combined set. Experiments were conducted using LIWC software to extract psycholinguistic features, and various classifiers, including Decision Trees, SVM, KNN, and Ensemble methods, were tested. Decision Trees performed best, with the highest recall and F-measure, achieving 60-80% accuracy in detecting depression-indicative comments. The study also includes time series analysis to track depression patterns over different times and months.

Table 2.4 Summary Table of Literature Review

2.3. FINDINGS AND PROPOSALS

Based on the analysis of the above 3 papers, it is evident that depression is a widespread issue in contemporary society, with social media emerging as an informal platform for individuals to express their feelings. Many individuals suffering from depression tend to express their emotions through social media posts and comments.

In response to this, my proposed system aims to detect depression from textual data posted on social media using machine learning classifiers and feature extraction methods. I have chosen to employ N-gram, and TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique methods to extract features for detecting depression in text, regardless of specific keywords such as 'depressed' or 'not depressed'. The system synthesizes emotion detection literature and identifies 4 types of features: emotional process, linguistic style, temporal process and a combined set.

The project intends to utilize Multi-Layer Perceptron (MLP) and Random Forest (RF) for ensemble methods, as well as Decision Trees (DT) to achieve the most accurate results. Additionally, the system will include a feature that suggests remedies for users identified as having depression.

The expected output of the system is to analyse text or any social media comment or post. If the post indicates any tone of sadness or contains depressive content, the model will determine that the user is experiencing depression, and vice versa. Furthermore, the system will provide suggested remedies for users identified as depressed.

To facilitate user interaction, I plan to implement a web interface using Flask. This interface will allow users to input text to be evaluated for depression. The output will indicate whether the text suggests the user is experiencing depression, and if so, will provide suggested remedies in an additional text field.

By utilizing advanced machine learning techniques and comprehensive feature extraction methods, the system aims to provide an accurate and practical tool for detecting depression in social media text and offering supportive suggestions.

3. SYSTEM ANALYSIS
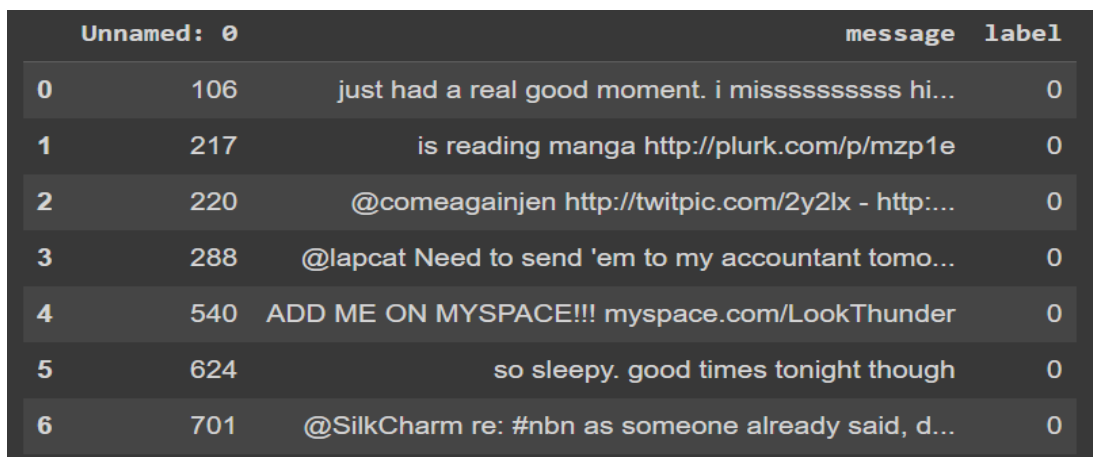
3.1. ANALYSIS OF DATASET

3.1.1. About the Dataset

The source of the Dataset is from Kaggle.com

LINK: https://www.kaggle.com/datasets/bababullseye/depression-analysis

Kaggle is a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish datasets, explore and build models in a web-based data-science environment.

Depression analysis dataset by BB eye is a valuable resource for depression analysis in social media textual data. It consists of 10314 records and is structured into three columns: the text of the post or comment, a unique identifier (id), and a class label. The class label indicates whether a piece of text is depressive or non-depressive, with "0" representing non-depressive content and "1" representing depressive content.

This dataset is particularly useful for training supervised learning models aimed at detecting depression-related content from social media posts. The absence of explicit keywords like "depressed" or "not depressed" presents an interesting challenge for the model, emphasizing the need for more nuanced text classification techniques. Figure 3.1 shows the snapshot of the dataset.

	Unnamed: 0	message	label
0	106	just had a real good moment. i missssssssss hi...	0
1	217	is reading manga http://plurk.com/p/mzp1e	0
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0
3	288	@lapcat Need to send 'em to my accountant tomo...	0
4	540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0
5	624	so sleepy. good times tonight though	0
6	701	@SilkCharm re: #nbn as someone already said, d...	0

Figure 3.1 Snapshot of dep.csv

3.1.2. Explore the Dataset

Figure 3.2 Shape of dep.csv

Figure 3.2 shows that the dataset has 10314 rows and 3 columns.

Figure 3.3 Datatypes of dep.csv

Figure 3.3 depicts that the dataset contains 2 numerical attributes and 1 non numerical attribute.

Figure 3.4 Value count of Class label

Figure 3.4 depicts the value counts in Class label which indicates whether the text is depressive or not. If the class label value is 0, the text is non depressive and the class label value is 1, the text is depressive.

During the analysis, I noticed that the dataset primarily includes tweets containing explicit keywords like "depressed" or "depression" to indicate depressive content. To address this limitation, I decided to augment the dataset with additional depressive tweets from another Kaggle dataset, enhancing the dataset's variety and improving the model's ability to detect depression beyond explicit keywords.

Source: Kaggle.com

LINK: https://www.kaggle.com/datasets/hyunkic/twitter-depression-dataset/data

After augmenting the dataset with additional depressive tweets to improve the model's ability to detect depression, the dataset exploration results are as follows:

Figure 3.5 Current Shape of dep.csv

Figure 3.6 Current Datatypes of dep.csv

Figure 3.7 Current Value Count of Class Label

3.2. DATA PREPROCESSING

3.2.1. Data Cleaning

Data cleaning is an important step in the machine learning process, as it involves preparing the data for modelling by identifying and addressing inconsistencies, errors, and missing values in the dataset. The quality of the data can have a significant impact on the accuracy and effectiveness of the machine learning model. The following are some common techniques used in data cleaning in machine learning:
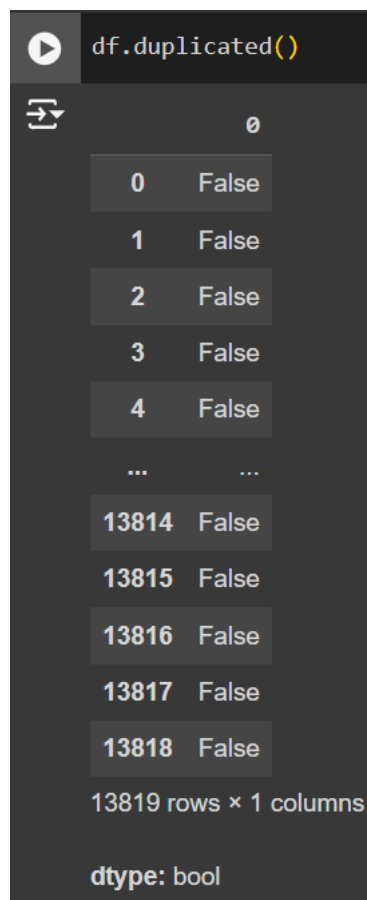
- Removing duplicates: This involves identifying and removing duplicate records from the dataset.

- Handling missing values: Missing values can be replaced with appropriate values, such as the mean, median, or mode of the data. Alternatively, the missing values can be removed altogether, but this approach should be used with caution as it can result in a loss of valuable data.

- Removing irrelevant features: Irrelevant features are variables that do not add value to the model and can reduce accuracy. Feature selection techniques, such as correlation analysis or model-based methods, can identify and remove these unnecessary features.

```
[ ] df.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 13819 entries, 0 to 13818
    Data columns (total 3 columns):
     #   Column      Non-Null Count  Dtype
    ---  ------      --------------  -----
     0   Unnamed: 0  13819 non-null  int64
     1   message     13819 non-null  object
     2   label       13819 non-null  int64
    dtypes: int64(2), object(1)
    memory usage: 324.0+ KB
```

Figure 3.8 Dataset Info

The info() method in Figure 3.8 provides a concise summary of a DataFrame, including the number of non-null values, the data type of each column, and the memory usage of the DataFrame. It also lists the total number of columns, and the number of entries in each column. This shows that the DataFrame has 13819 entries and 3 columns. The first column is an unnamed index and the other two columns are labelled "message" and "label". The first column is float data type, last column is integer datatype and second column is of type object (which usually contain strings). The output also shows that there are no null values in any of the columns. Finally, the memory usage of the DataFrame is shown to be 324.0+ KB.

Figure 3.9 Checking Duplicates

As mentioned in the Figure 3.9, there are no duplicate values in the dataset. So, it returns the dataset with all field values as False. If any field is True, then it returns True for that field.

Figure 3.10 Checking Missing Values

Figure 3.10 depicts that there are no missing values in the dataset.

Figure 3.11 Dropping Irrelevant Feature

As shown in Figure 3.11, Dropped Irrelevant feature column Unnamed: 0.

From these interpretations our dataset doesn't contain any missing values, duplicates, and irrelevant features.

```python
# removal of links[https://abc.com]
giant_url_regex =  re.compile('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
tweets = tweet_name.str.replace(giant_url_regex, '',regex=True)

# removal of punctuations and numbers
punc_remove = tweets.str.replace("[^a-zA-Z]", " ",regex=True)
# remove whitespace with a single space
newtweet=punc_remove.str.replace(r'\s+', ' ')
# remove leading and trailing whitespace
newtweet=newtweet.str.replace(r'^\s+|\s+?$','')
# replace normal numbers with numbr
newtweet=newtweet.str.replace(r'\d+(\.\d+)?','numbr')
# removal of capitalization
tweet_lower = newtweet.str.lower()

# tokenizing
tokenized_tweet = tweet_lower.apply(lambda x: x.split())

 # stemming of the tweets
tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i in x])

 # removal of stopwords
 tokenized_tweet=  tokenized_tweet.apply(lambda x: [item for item in x if item not in stopwords])

 for i in range(len(tokenized_tweet)):
     tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
     tweets_p= tokenized_tweet

 return tweets_p

processed_tweets = preprocess(tweet)

df['processed_messages'] = processed_tweets
print(df[["message","processed_messages"]].head(10))
```

Figure 3.12 Text Preprocessing Task

The text preprocessing tasks aim to clean and standardize the Dataset, making it more suitable for subsequent analysis or modelling. The provided code in Figure 3.12 successfully performs these tasks, and the processed messages can be used for various natural language processing tasks, sentiment analysis, or machine learning applications.

NLTK is a powerful library for natural language processing in Python. It provides various tools and resources for tasks such as tokenization, stemming, and stopword removal. In this project, NLTK is employed to handle English stopwords and perform stemming. The re library is used for pattern matching and manipulation of strings. It plays a crucial role in the removal of mentions, hyperlinks, and other specific patterns from Twitter data. Figure 3.13 Shows the Pre-processed text using the above code.

- Removal of Extra Spaces, Mentions, and Hyperlinks: Regular expressions are used to remove extra spaces, mentions (starting with '@'), and hyperlinks from the messages.

- Removal of Punctuation, Numbers, and Capitalization: Punctuation and numbers are removed, and the messages are converted to lowercase. This step ensures uniformity and reduces noise in the text.

- Tokenization: The lowercase, punctuation-stripped tweets are tokenized into lists of words using the split() function.

- Stemming: The Porter stemming algorithm is applied to each tokenized message, reducing words to their stems.

- Removal of Stopwords: Stopwords, including custom Twitter-specific terms, are removed from the tokenized and stemmed tweets.

- Dataset Update: The processed messages are added as a new column ('processed_messages') to the original dataset.

Figure 3.13 Processed Text

3.2.2. Analysis of Feature Variables

Feature variables are input variables used to predict the class. In this dataset there is only one and the most important feature variable.

- message: This contains the text of the tweet. The text can be transformed into features using techniques like text vectorization (n-gram,TF-IDF).

3.2.3. Analysis of class variables

Class variables are target variables that is needed to be predicted.

- label: This appears to be a binary label indicating sentiment, with values such as 0 and 1.

This Dataset has 2 Classes:
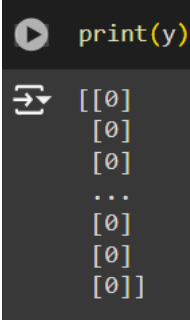
1. Non-Depressive: It is denoted as 0.
2. Depressive: It is denoted as 1.

```
[ ] dt_transformed=df[['label','processed_messages']]
    y=dt_transformed.iloc[:,:-1].values


[ ] print(dt_transformed)

           label                          processed_messages
    0          0              real good moment misssssssss much
    1          0                                     read manga
    2          0    need send em account tomorrow oddli even refer...
    3          0                    add myspac myspac com lookthund
    4          0                    sleepi good time tonight though
    ...      ...                                            ...
    13728      0    may not answer trust everyth work time day new...
    13729      0    woke today feel lighter like mayb thing start ...
    13730      0    take one step time life perfect grate littl mo...
    13731      0    today new day choos focu thing make happi matt...
    13732      0    sometim life tough learn embrac challeng growt...

    [13733 rows x 2 columns]
```

Figure 3.14 Analysing Class Variables

In Figure 3.14, the variable dt_transformed is created as a new DataFrame by selecting only the 'label' and 'messages' columns from the original 'dataset'. This means that dt_transformed now contains only the 'label' and 'messages' columns from the original dataset. The next line of code y = dt_transformed.iloc[:, :-1].values selects all rows of the 'label' column of dt_transformed using the .iloc() method and stores them as a NumPy array in the variable y. The [:, :-1] syntax selects all rows and all columns except for the last column, which in this case is the 'message' column. This means that y will contain the values of the 'label' column as a NumPy array.

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique is employed to convert a collection of text documents into a numerical representation that is suitable for machine learning tasks. This code utilizes the TfidfVectorizer class from scikit-learn to perform TF-IDF feature extraction on a dataset of processed messages. Below is an explanation for the TF-IDF feature extraction using the provided code in Figure 3.15.

1. The TfidfVectorizer is configured with the following parameters:
   ngram_range = (1, 2) : Considers both unigrams and bigrams in the text, capturing single words and two-word combinations.

   max_df = 0.75: Ignores terms that appear in more than 75% of the documents, helping filter out common and less informative words.

   min_df =5: Ignores terms that appear in fewer than 5 documents, eliminating rare words that may not contribute significantly.

   max_features = 10000: Limits the vocabulary size to the top 10,000 features, controlling the dimensionality of the resulting TF-IDF matrix.

2. TF-IDF Feature Matrix:

The `fit_transform` method is applied to the 'processed_messages' column of the dataset, transforming the text data into a TF-IDF feature matrix. The resulting `tfidf` variable represents a sparse matrix where each row corresponds to a message, and each column corresponds to a unique word or word combination in the vocabulary.

3. Conversion to Pandas DataFrame:

The TF-IDF sparse matrix is converted to a Pandas DataFrame (`p`) using DataFrame(tfidf). This DataFrame provides a more structured and user-friendly representation of the TFIDF features, allowing for easier inspection and analysis.

4. Preview of the TF-IDF DataFrame:

The `head()` method is used to display the first few rows of the TF-IDF DataFrame (`p`), providing an insight into the numerical representation of the processed messages.

```python
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),max_df=0.75, min_df=5, max_features=10000)

# TF-IDF feature matrix
tfidf = tfidf_vectorizer.fit_transform(df['processed_messages'] )
tfidf
```

```
<13724x3995 sparse matrix of type '<class 'numpy.float64'>'
        with 104478 stored elements in Compressed Sparse Row format>
```

```python
[ ] from pandas.core.frame import DataFrame
    p=DataFrame(tfidf)

[ ] p.head()
```

	0
0	(0, 2782)\t0.4214115762478841\n (0, 1401)\t...
1	(0, 2776)\t1.0
2	(0, 2277)\t0.20749105380938423\n (0, 2997)\...
3	(0, 30)\t0.39820971779869035\n (0, 2264)\t0...
4	(0, 1401)\t0.26651672757548955\n (0, 3490)\...

Figure 3.15 Conversion of Texts

3.3.   DATA VISUALZATION

Data visualization serves as a fundamental technique in data analysis, offering a graphical representation of information through charts, graphs, and maps. Its significance lies in transforming complex datasets into visually intuitive formats, allowing for the identification of trends, patterns, and outliers. This visual representation enhances data exploration, enabling easy comparison between data points and facilitating effective communication of insights. The diverse types of visualizations, such as charts for trends, tables for structured data, graphs for relationships, maps for geographic data, and dashboards for comprehensive displays, cater to various analytical needs. Utilizing tools like Tableau, Excel, or programming libraries like Matplotlib, data visualization aids in making data driven decisions by providing accessible insights into massive amounts of information.

The code in Figure 3.16 creates a bar chart using Matplotlib in Python. The data being plotted is the number of messages in a dataset that falls into each of two categories: "0", and "1". The resulting bar chart will have two bars, one for each category. The height of each bar will correspond to the count of messages in that category. The labels on the x-axis will be the class, and the y-axis will display the number of messages.

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Count the occurrences of each class
class_counts = df['label'].value_counts()

# Create the plot
plt.figure(figsize=(10, 6))  # Adjust figure size for better visualization
sns.countplot(x='label', data=df, palette='viridis') # Use a visually appealing palette

# Customize the plot
plt.title('Number of Messages per Class', fontsize=16)
plt.xlabel('Class', fontsize=14)
plt.ylabel('Number of Messages', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Add value annotations to each bar
for p in plt.gca().patches:
    height = p.get_height()
    plt.gca().text(p.get_x() + p.get_width() / 2., height + 1, height, ha='center', fontsize=12)

# Improve the layout and display the plot
plt.tight_layout()
plt.show()
```

Figure 3.16 Code for Data Visualization
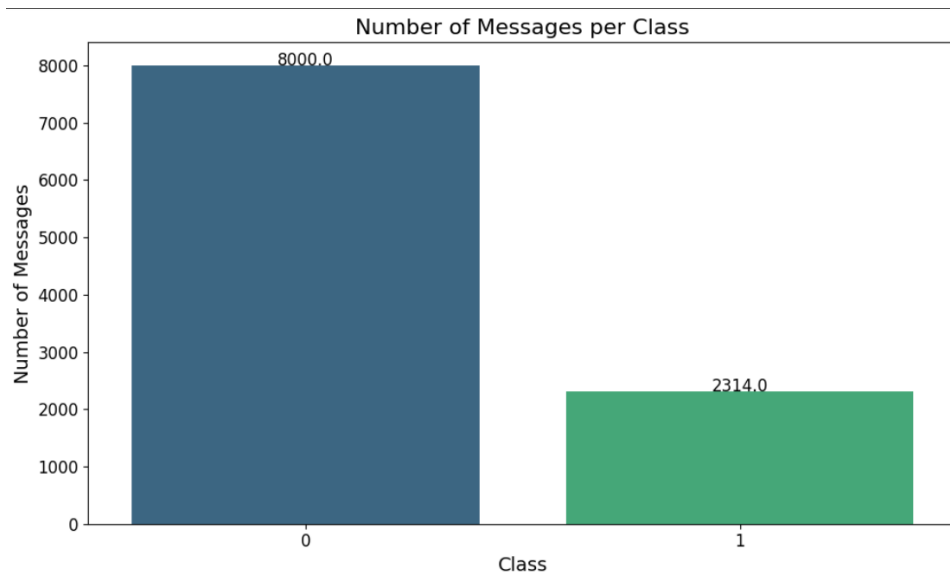
Figure 3.17 Visualization Before Dataset Augmentation

Figure 3.17 shows the value counts before Data Augmentation.
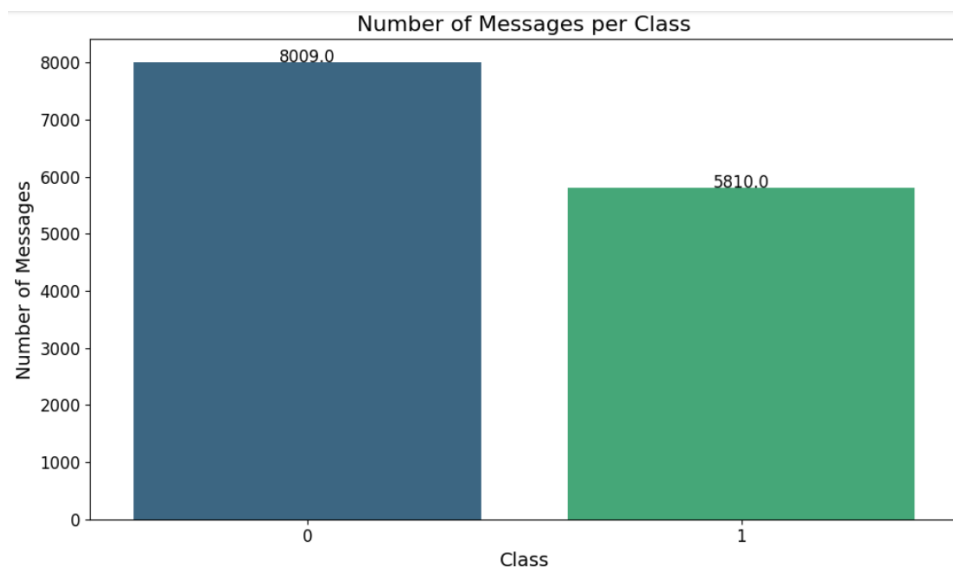
Figure 3.18 Visualization After Dataset Augmentation

Figure 3.18 shows the value counts after Data Augmentation.

3.4. ANALYSIS OF ALGORITHMS

Algorithms used in 'Detection of Depression from Textual Data in Social Media Using Machine Learning' are Random Forest, Decision Tree and Multilayer Perceptron.

Random Forest

Random Forest is a supervised learning technique used for both Classification and Regression problems. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. Instead of relying on one decision tree, the random forest takes the output from each tree and based on the majority votes of outputs, and it takes the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. Figure 3.19 gives a brief image of Random Forest.

Figure 3.19 Random Forest

Working:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Pseudocode:

1. Input:

  - Training dataset with N samples and M features.

  - Number of trees to create: T.

  - Number of features to randomly select at each split: F.

2. Initialize an empty list called 'forest' to store the trees.

3. For each tree (from 1 to T):

  a. Create a random sample of the training data (some samples may repeat).

  b. Build a decision tree:

    i. At each decision point in the tree:

      - Randomly pick F features.

      - Find the best feature and split point among the F features to split the node.

      - Split the node into two child nodes based on the selected split point.

    ii. Keep splitting until the tree is fully grown or meets some stopping condition (like a max depth or minimum samples).

  c. Add this decision tree to the 'forest'.

4. To make a prediction for a new sample x:

  a. Let each tree in the forest make a prediction.

  b. The final prediction is the one that most trees agree on (majority vote).
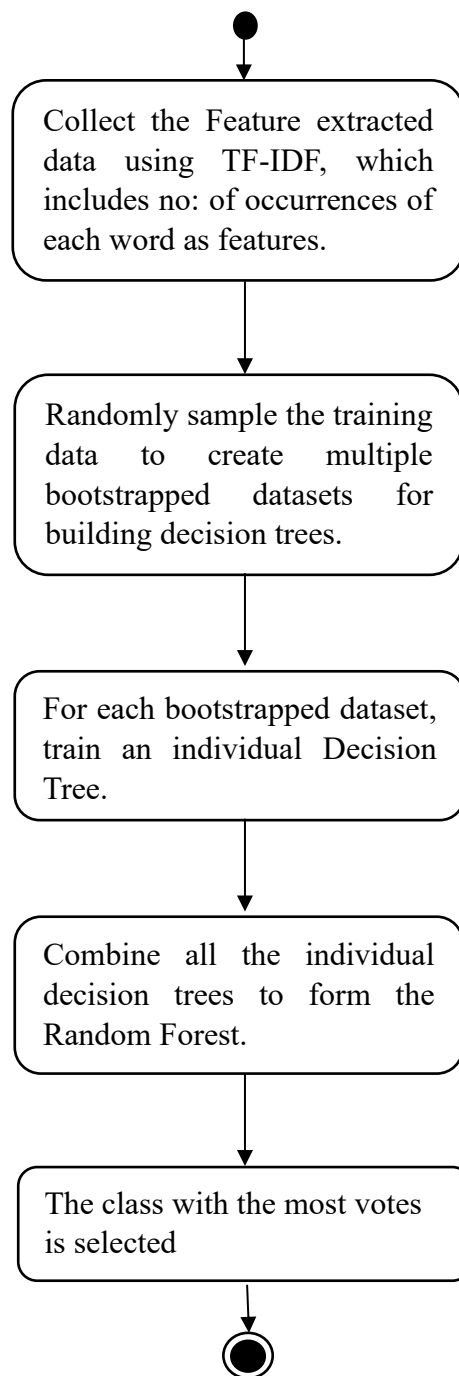
Figure 3.20 Activity Diagram of Random Forest

Decision Tree

Decision trees are a type of machine-learning algorithm that can be used for both classification and regression tasks. They work by learning simple decision rules inferred from the data features. These rules can then be used to predict the value of the target variable for new data samples. Example of Decision Tree shows in Figure 3.20. In the Decision Tree, the major challenge is the identification of the attribute for the root node at each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain: When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

2. Gini Index: Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with a lower Gini index should be preferred.
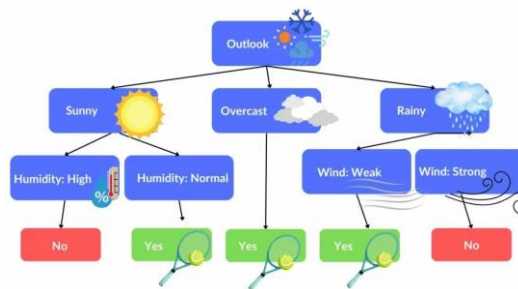
Figure 3.21 Decision Tree

Working:

Step 1: Start with the entire dataset at the root node.

Step 2: Select the best feature to split the data at the root node using the attribute selection measure.

Step 3: Split the dataset into subsets based on the chosen feature and split point.

Step 4: Repeat Steps 2 & 3 for each subset to build the tree.

Step 5: Assign a final prediction to each leaf node.

Step 6: For new data points, pass them down the tree to reach a leaf node and output the prediction.

Pseudocode:

1. Input:

   - Training dataset with N samples and M features.

   - Stopping criteria, such as maximum depth or minimum number of samples per leaf.

2. Initialize the root of the tree with the entire dataset.

3. For each node in the tree:

   a. If all samples at the node belong to the same class (classification) or the maximum depth is reached, make the node a leaf node and assign the class label or average value.

   b. Else:

     i. Calculate the best feature and split point based on a criterion (like Gini Impurity, Information Gain for classification, or Variance Reduction for regression).

     ii. Split the dataset into two subsets based on the best feature and split point.

     iii. Create child nodes for each subset.

     iv. Recursively apply steps 3a and 3b to the child nodes.

4. Output: A fully grown decision tree.

5. To make a prediction for a new sample x:

   a. Start at the root node.

   b. Traverse the tree by following the decision rules at each internal node.

   c. Continue until a leaf node is reached.

   d. The prediction is the value or class label assigned to the leaf node.
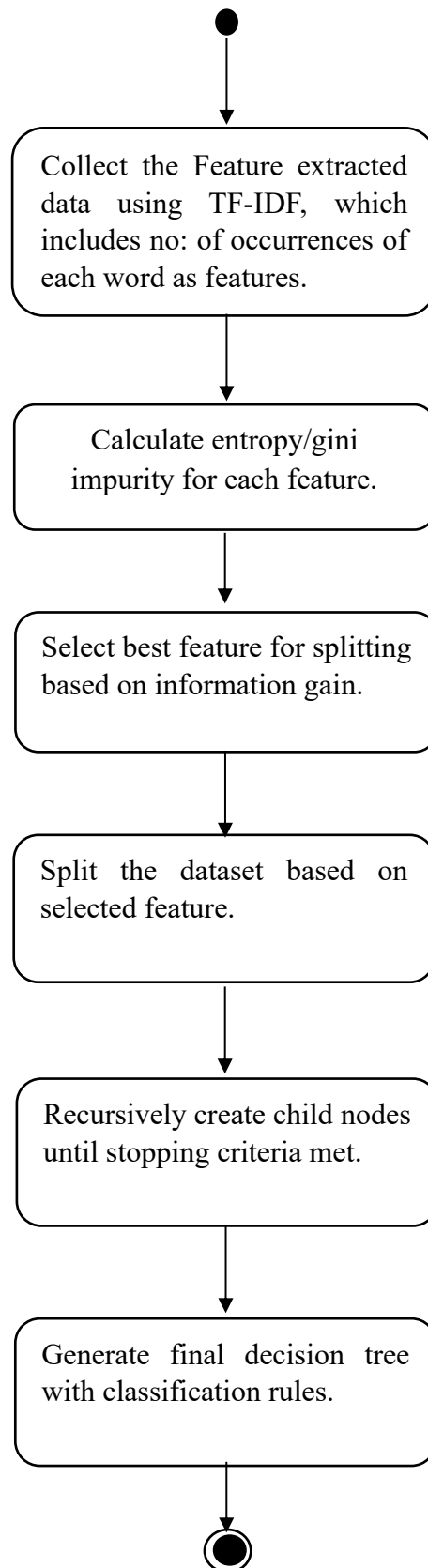
Figure 3.22 Activity Diagram of Decision Tree

Multilayer Perceptron

A Multilayer Perceptron (MLP) is a type of artificial neural network used for supervised learning tasks, such as classification and regression. It consists of an input layer, one or more hidden layers, and an output layer, where each layer is made up of neurons. These neurons are fully connected between layers and use activation functions like ReLU or Sigmoid to learn complex, non-linear relationships in the data. The network is trained using backpropagation, where the weights are adjusted iteratively to minimize the error between predicted and actual outputs. MLPs are versatile and can be applied to a wide range of problems.
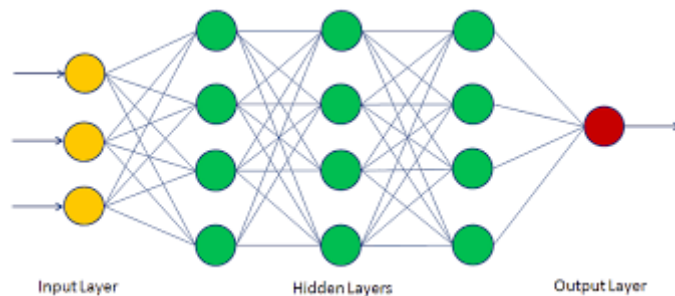
Figure 3.23 Multilayer Perceptron

Working:

Step 1: Input Layer

- The input features are fed into the input layer, with each feature represented by a neuron.

Step 2: Forward Propagation

- Hidden Layers: Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.
- This process repeats through all hidden layers.

Step 3: Output Layer

- The final layer computes the output, applying an activation function if needed (e.g., Softmax for classification).

Step 4: Loss Calculation

- The difference between the predicted and actual output is calculated using a loss function.

Step 5: Backpropagation

- The error is backpropagated to update the weights using gradient descent.

Step 6: Training

- Repeat forward propagation, loss calculation, and backpropagation for several iterations to minimize the loss.

Step 7: Prediction

- After training, the MLP makes predictions on new data using forward propagation.

Pseudocode:

1. Input:

   - Training dataset with N samples and M features.

   - Number of hidden layers: L.

   - Number of neurons in each hidden layer.

   - Activation function (e.g., ReLU, Sigmoid).

   - Learning rate (α).

   - Number of epochs (iterations).

2. Initialize weights and biases for all layers.

3. For each epoch:

  a. For each sample in the training dataset:

    i. Forward Propagation:

     - For each layer from 1 to L:

      1. Compute the weighted sum: z = (weights * input) + bias.

      2. Apply activation function: a = activation(z).

      3. Set the output of the current layer as input for the next layer.

     - Compute the final output in the output layer.

    ii. Compute the loss using the loss function (e.g., Mean Squared Error, Cross-Entropy).

    iii. Backward Propagation:

     - Calculate the gradient of the loss with respect to each weight and bias.

     - Update weights and biases using gradient descent:

      weight = weight - α * (gradient of loss with respect to weight)

      bias = bias - α * (gradient of loss with respect to bias)

4. After training, use the learned weights and biases to make predictions on new data by performing forward propagation.

5. Output:

   - Trained model that can make predictions on new data.
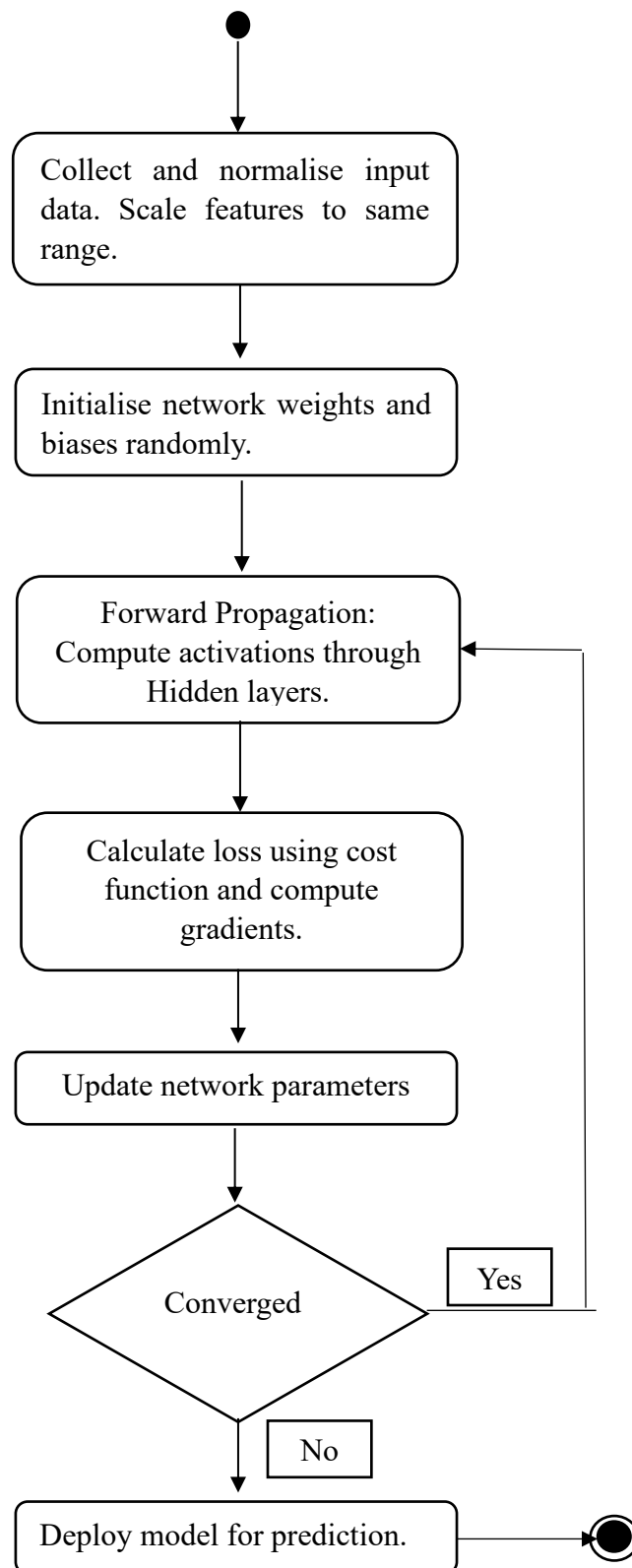
Figure 3.24 Activity Diagram of Multilayer Perceptron

3.5. PROJECT PLAN

3.5.1. Project Pipeline
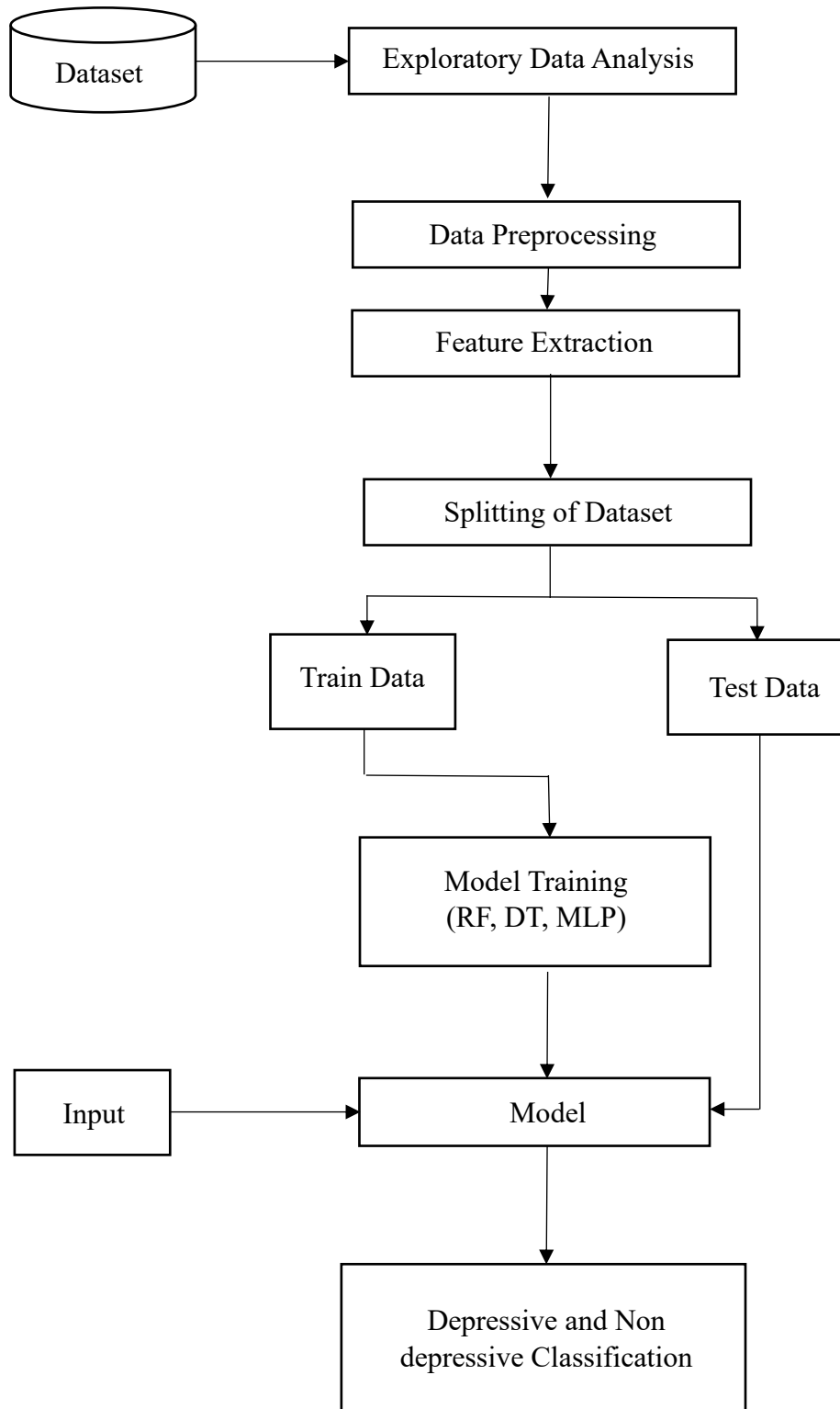
Figure 3.25 Project Pipeline

3.6.   FEASIBILITY ANALYSIS

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing system or proposed system, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success.

Evaluated the feasibility of the system in terms of the following categories:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

3.6.1. Technical Feasibility

Technical feasibility assesses whether the technology required for your project is readily available, and if your team has the expertise to implement it effectively.

Data Processing and Analysis:

Feasibility: This project involves processing and analyzing textual data from social media posts to detect depressive content. The workflow includes extensive preprocessing steps, such as stopword removal and stemming using NLTK's PorterStemmer, and cleaning with regular expressions (re). Feature extraction is handled by TFIDF vectorization through scikit-learn, which enables effective numerical representation of text data. These tools are widely available, well-supported, and commonly used for data manipulation and analysis, ensuring strong technical feasibility.

Expertise: Familiarity with these tools, as evidenced by their usage in the project, indicates technical competency.

Machine Learning Algorithms:

Feasibility: Using Random Forest, Decision Tree, and Multilayer Perceptron for depression detection is both practical and effective. These algorithms work well with the features created by TF-IDF and N-gram vectorization, making them suitable for identifying depressive content in text data. Their adaptability and strong performance ensure that the model can reliably classify text data from social media.

Expertise: The implementation of these algorithms suggests a technical understanding of machine learning concepts.

Data Visualization:

Feasibility: Matplotlib and Seaborn are used for data visualization, indicating the project's reliance on established and technically feasible visualization tools.

Expertise: The team's ability to visualize data trends suggests proficiency in using these tools.

File Handling:

Feasibility: Reading and writing data to CSV files using Pandas is a standard practice, ensuring technical feasibility.

Expertise: File handling is a fundamental skill, and your team's successful utilization of it implies technical competence.

3.6.2. Economic Feasibility

Economic feasibility assesses whether the project is financially viable, considering costs and potential benefits.

Development Costs:

Feasibility: Open-source tools and libraries are used, reducing software acquisition costs. However, consider potential costs related to personnel, training, and hardware.

Benefits: A well-designed depression detection system for social media can provide significant benefits, including early identification of depressive content, better support options, and insights for mental health interventions.

Maintenance Costs:

Feasibility: Open-source tools often have lower maintenance costs. Consider ongoing costs related to data updates, algorithm improvements, and support.

Benefits: The long-term benefits of a depression detection system, such as improved accuracy in identifying depressive content, should outweigh maintenance costs.

3.6.3. Operational Feasibility

Operational feasibility evaluates whether the project aligns with operational requirements and whether it can be smoothly integrated into existing processes.

User Acceptance:

Feasibility: The project aims to improve the accuracy of detecting depressive content in social media, which can enhance user experience and satisfaction for mental health professionals. By providing timely insights, the system supports better mental health care.

Integration: If the depression detection system can be seamlessly integrated with existing mental health platforms or tools, it will significantly enhance operational feasibility, making it easier for professionals to utilize the system in their workflows.

Scalability:

Feasibility: Consideration should be given to scalability as the volume of text data increases. Implementing machine learning algorithms like RF, DT, and MLP on larger datasets might present challenges, particularly regarding processing time and resource allocation. Ensuring that the system can handle increased data loads efficiently will be crucial for its long-term success.

Mitigation: To address scalability concerns, strategies like distributed computing, cloud-based deployment, or optimization techniques can be explored to maintain model performance and manage high data loads as usage grows.

Ease of Use:

Feasibility: If the system is designed with a user-friendly interface and easy navigation, it enhances operational feasibility.

Training: Assess the ease with which users can understand and interact with the system.

The software environment leverages the Python programming language and key libraries for data processing, machine learning, and visualization. Development can be facilitated through Colab Notebooks. The hardware environment requires a standard computing setup with ample RAM and storage, and optional utilization of cloud services for scalability. This combination ensures a robust and flexible system environment for the development, testing, and potential deployment of the detection system.

3.7.   SYSTEM ENVIRONMENT

3.7.1. Software Environment

The software environment encompasses the tools, frameworks, and platforms utilized in the development and execution of the detection system.

Programming Languages:

Python: Python serves as the primary programming language for this project due to its simplicity, flexibility, and extensive support for data science and machine learning. Python's readability and widespread adoption in scientific computing make it an ideal choice, especially with its wide range of libraries designed for data analysis, model training, and evaluation.

Data Processing and Analysis:

Pandas: Pandas is used for loading, managing, and transforming the dataset, particularly with CSV files. It provides easy-to-use data structures, such as DataFrames, allowing for the intuitive handling of both numerical and categorical data.

NLTK: NLTK is used for natural language processing tasks. In your project, it assists with stopword removal and stemming (using PorterStemmer), crucial for cleaning and preparing the text data for analysis.

Regular Expressions (re): help clean and preprocess text by removing special characters, URLs, and other unwanted elements, ensuring that only relevant data is passed on to feature extraction.

Machine Learning Libraries:

scikit-learn: Used for implementing and training machine learning models and is also used for feature extraction. In this project, it enables TF-IDF and N-gram vectorization to convert text into numerical features. Employed for implementing machine learning algorithms, particularly Random Forest, Decision tree and MLP.

Data Visualization:

Matplotlib and Seaborn: Chosen for creating insightful visualizations, aiding in the exploration and communication of data patterns. It is employed to generate standard charts, such as line plots, bar charts, histograms, and scatter plots, which provide a clear understanding of the underlying data distribution and relationships among features.

File Handling:

Pandas: Applied for reading and writing data to CSV files, ensuring seamless data management.

Development Environment:

Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. We can write and execute code in Python. Colab supports many machine learning libraries which can be easily loaded in the colab notebook.

Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tool a developer needs for a quick code-build-debug cycle and leaves more complex work flows to fuller featured IDEs, such as Visual Studio IDE.

HTML and CSS

Hyper Text Markup Language is used for creating web pages.HTML describes the structure of the web page. Here, the user interface of my project is done using HTML. This ensures that users can interact with the system smoothly and that all functionalities are accessible and straightforward.

CSS

CSS adds the styling layer to the project interface, enhancing its visual appeal and usability. Cascading Style Sheet is used with HTML to style the web pages. CSS also enables responsive design, allowing the interface to adjust seamlessly across different devices, including desktops, tablets, and mobile phones.

Flask

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's having a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features. It does have many cool features like URL routing, template engine. It is a WSGI web app framework.

GitHub

Git is an open-source version control system that was started by Linus Torvalds. Git is similar to other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better.

The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward.

3.7.2. Hardware Environment

The hardware environment refers to the physical infrastructure necessary to support the development and deployment of the detection system.

Computing Resources:

Processor: 2 GHz, A powerful multi-core processor is recommended to handle the computational demands efficiently. Consider a processor with at least quadcore architecture for optimal performance.

Storage:

Sufficient Disk Space: 512 GB SSD

Memory (RAM):

8 GB RAM: Sufficient RAM is crucial for efficiently handling large datasets and performing machine learning tasks.

Internet Connectivity:

High-Speed Internet: A stable and high-speed internet connection is essential for accessing online resources, libraries, and datasets during development.

4. SYSTEM DESIGN

4.1. MODEL BUILDING

4.1.1. Model Planning

The model planning phase for depression detection in social media focuses on designing a structured approach to identify depressive content using machine learning techniques. This involves selecting suitable algorithms (RF, DT, and MLP), and outlining strategies for accurately classifying text data based on linguistic and emotional features extracted through TF-IDF and N-gram vectorization.

1. Objective:
    Develop a system to detect depression from textual data on social media using machine learning techniques. The goal is to analyse posts to identify patterns indicative of depression, without relying on explicit keywords such as 'depressed.' This tool aims to assist in early detection of depressive conditions and provide support to those at risk.

2. Approach:
    Conduct a comparative study of three machine learning algorithms: Multilayer Perceptron (MLP), Random Forest (RF), and Decision Tree (DT). Evaluate these models based on their accuracy, precision, and recall in detecting depression from social media text data. Feature extraction methods such as N-grams and TF-IDF will be employed to convert textual data into machine-readable format.

3. Data Preparation:
    Utilize a dataset from Kaggle consisting of posts labelled as depressive or non-depressive. Preprocess the dataset by handling missing values, removing duplicates, and performing text cleaning operations such as tokenization, stemming, and stopword removal. Split the dataset into training and testing sets for evaluation.

4. Exploratory Data Analysis (EDA):
    Perform EDA to understand the distribution of depressive and non-depressive posts in the dataset. Analyse features such as text length, frequent terms, and emotional tone to identify patterns that could aid in the classification of depressive posts.

5. Model Building:
    Implement the machine learning models: Multilayer Perceptron (MLP), Random Forest (RF), and Decision Tree (DT). Use the pre-processed text data with features extracted through TF-IDF and N-grams to train the models. Evaluate their performance on the test data using metrics like accuracy, confusion matrix, and classification reports.

6. Model Comparison:
    Compare the performance of the three algorithms based on accuracy, F1 score, and speed. The model that best generalizes to unseen data will be selected for deployment, and suggestions for remedies will be provided for users identified as experiencing depression.

4.1.2. **Model Training**

The dataset was divided into two parts. X representing the input features, and y representing the target variable. The training set consists of 80% of the data and is used to train the model. Figure 4.1 shows the split of dataset.

```
[ ] from sklearn.model_selection import train_test_split # Import the train_test_split function from sklearn.model_selection

    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=42)
```

Figure 4.1 Dataset Splitting

Random Forest

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report, accuracy_score

    # Initialize Random Forest Classifier
    clf_rf = RandomForestClassifier(max_depth= None, n_estimators=500, random_state=42)
    clf_rf.fit(X_train, y_train)

    # Predict on test data
    y_pred_rf = clf_rf.predict(X_test)

    # Evaluate
    print(clf_rf.score(X_train, y_train))
    print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
    print(classification_report(y_test, y_pred_rf))
```

Figure 4.2 Random Forest training

A Random Forest Classifier, which is an ensemble learning method based on multiple decision trees is used to train the model. The model consists of 300 decision trees. The maximum depth of each tree is none and random_state is set to 42 to ensure reproducibility. The classifier is fit to the training data (X_train, y_train), and predictions are made on the test data (X_test).

```
print(clf_rf.score(X_train, y_train))
0.9949026033133078
```

Figure 4.3 Random Forest training accuracy

The model achieved an accuracy of 99.49% on the training data as shown in Figure 4.3.

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score

# Initialize Decision Tree Classifier
clf_tree = DecisionTreeClassifier(random_state=42)
clf_tree.fit(X_train, y_train)

# Predict on test data
y_pred_tree = clf_tree.predict(X_test)

# Evaluate
print(clf_tree.score(X_train, y_train))
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_tree))
print(classification_report(y_test, y_pred_tree))
```

Figure 4.4 Decision Tree training

The model is initialized with a set random_state of 42, which ensures that the results are reproducible by controlling the randomness in the training process. The classifier is fit to the training data (X_train, y_train), and predictions are made on the test data (X_test).

```
print(clf_tree.score(X_train, y_train))
0.9949026033133078
```

Figure 4.5 Decision Tree training accuracy

The model achieved an accuracy of 99.49% on the training data as shown in Figure 4.5.

Multilayer Perceptron

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report

# Initialize MLP Classifier
clf_mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
clf_mlp.fit(X_train, y_train)

# Predict on test data
y_pred_mlp = clf_mlp.predict(X_test)

# Evaluate
print(clf_mlp.score(X_train, y_train))
print("MLP Accuracy:", accuracy_score(y_test, y_pred_mlp))
print(classification_report(y_test, y_pred_mlp))
```

Figure 4.6 Multilayer Perceptron training

The model consists of 100 neurons in the hidden layer. The maximum number of iterations is set to 300 for training, and the random_state is set to 42 to ensure reproducibility. The classifier is fit to the training data (X_train, y_train), and predictions are made on the test data (X_test).

```
print(clf_mlp.score(X_train, y_train))
0.9910795557982888
```

Figure 4.7 Multilayer Perceptron training accuracy

The model achieved an accuracy of 99.10% on the training data as shown in Figure 4.7.

4.1.3. Model Testing

```
Random Forest Accuracy: 0.9078995267564616
              precision    recall  f1-score   support

           0       0.91      0.94      0.92      1592
           1       0.91      0.87      0.89      1155

    accuracy                           0.91      2747
   macro avg       0.91      0.90      0.90      2747
weighted avg       0.91      0.91      0.91      2747
```

Figure 4.8 Random Forest testing accuracy

Random forest obtained an accuracy of 90.78% as shown in Figure 4.8.

```
Decision Tree Accuracy: 0.8693119767018566
              precision    recall  f1-score   support

           0       0.89      0.88      0.89      1592
           1       0.84      0.86      0.85      1155

    accuracy                           0.87      2747
   macro avg       0.87      0.87      0.87      2747
weighted avg       0.87      0.87      0.87      2747
```

Figure 4.9 Decision Tree testing accuracy

Decision Tree obtained an accuracy of 86.93% as shown in Figure 4.9.

```
MLP Accuracy: 0.8776847469967237
              precision    recall  f1-score   support

           0       0.90      0.89      0.89      1592
           1       0.85      0.86      0.86      1155

    accuracy                           0.88      2747
   macro avg       0.87      0.88      0.87      2747
weighted avg       0.88      0.88      0.88      2747
```

Figure 4.10 Multilayer Perceptron testing accuracy

Multilayer Perceptron obtained an accuracy of 87.76% as shown in Figure 4.10.

5. RESULTS AND DISCUSSIONS

The algorithms are compared using various performance metrics. Random Forest algorithm has the highest accuracy of any algorithm. After Random Forest, Multilayer Perceptron followed by Decision Tree.

Table 5.1 Evaluation of Algorithms

Algorithm	Accuracy	Precision	Recall	F1-score
Random Forest Classifier	91	91	91	91
Decision Tree	87	87	87	87
Multilayer Perceptron	88	87	88	87

From the confusion matrix, as shown in Figure 5.1, we can also say that Random Forest gives us the best prediction, and Decision Tree gives us the poorest prediction in this case. As a result, we can conclude that for our chosen dataset, Random Forest is the best classification algorithm. Overall, RF is better as RF has fewer wrong predictions compared to DT and MLP and RF gives more correct predictions.

(a) RF

(b) DT
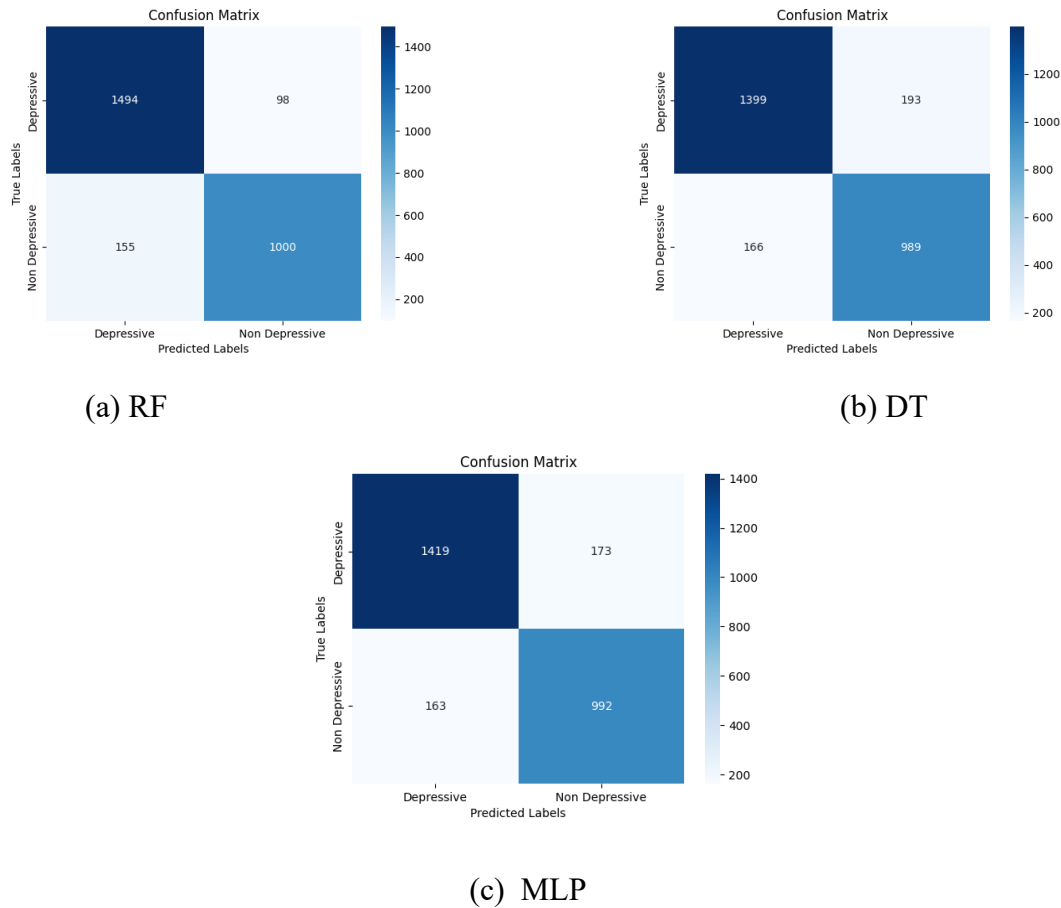
(c) MLP

Figure 5.1 Confusion matrix of different algorithms a) RF, b) DT and c) MLP

Validation with Unseen Data

Validating the Depression detection model on unseen data is essential to ensure its accuracy and reliability in real-world applications. Testing with unseen data provides insight into the model's performance with new, independent cases, helping to prevent overfitting and confirming its generalizability.

```python
# Import necessary libraries
from sklearn.feature_extraction.text import TfidfVectorizer
import re
import pandas as pd
import pickle
import nltk
import string
import pandas as pd
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import seaborn as sns


df=pd.read_csv("/content/drive/MyDrive/depression/send/preprocessedsec.csv")

processed_messages= df['processed_messages'].astype(str).tolist()

# Sample input text
sample=input(" enter input text")

# Preprocess the input text
def preprocess_input_text(sample):
    stemmer = PorterStemmer()
```

```python
# Define the list of extended stopwords
extended_stopwords = nltk.corpus.stopwords.words("english")
other_exclusions = ["#ff", "ff", "rt"]
extended_stopwords.extend(other_exclusions)

# Preprocessing steps
# Step 1: Removal of extra spaces
tweet_space = re.sub(r'\s+', ' ', sample)

# Step 2: Removal of @name[mention]
tweet_name = re.sub(r'@[\w\-]+', '', tweet_space)

# Step 3: Removal of links [https://abc.com]
tweet_no_links = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', tweet_name)

# Step 4: Removal of punctuations and numbers
tweet_no_punctuation = re.sub("[^a-zA-Z]", " ", tweet_no_links)

# Step 5: Remove leading and trailing whitespace
tweet_stripped = tweet_no_punctuation.strip()

# Step 6: Replace normal numbers with 'numbr'
tweet_no_numbers = re.sub(r'\d+(\.\d+)?', 'numbr', tweet_stripped)

# Step 7: Convert to lowercase
tweet_lower = tweet_no_numbers.lower()
```

```
    # Step 8: Tokenize
    tokenized_tweet = tweet_lower.split()

    # Step 9: Remove stopwords and stem the tokens
    tokenized_tweet = [stemmer.stem(token) for token in tokenized_tweet if token not in extended_stopwords]

    # Step 10: Join the tokens back into a string
    processed_input_text = ' '.join(tokenized_tweet)

    return processed_input_text

# Preprocess the input text
processed_input_text = preprocess_input_text(sample)

# Initialize TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2), max_df=0.75, min_df=5, max_features=10000)

# Fit the TF-IDF vectorizer on your dataset
tfidf = tfidf_vectorizer.fit_transform(processed_messages)  # Assuming processed_messages is your preprocessed dataset

# Transform the processed input text
tfidf_sample = tfidf_vectorizer.transform([processed_input_text])
```

```python
with open('/content/drive/MyDrive/depression/send/RF_model.pkl', 'rb') as model_file:
    rf = pickle.load(model_file)

# Make predictions
predicted_label = rf.predict(tfidf_sample)[0]

# Map class labels to their meanings
class_labels = {
    0: "Non Depressive",
    1: "Depressive"
}

# Get the label for the predicted class
predicted_label_text = class_labels[predicted_label]

# Print the result
print("Input Text:", sample)
print("Predicted Label:", predicted_label_text)
```

Figure 5.2 Validation with unseen data.

Figure 5.2 shows the implementation code for validation with unseen data

For validating with unseen data first we have to enter a input text that is not in the dataset as shown in Figure 5.3.

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
 enter input text[                                    ]
```

Figure 5.3 User input text

```
Input Text: I'm exhausted from pretending to be okay when all I want to do is disappear for a while.
Predicted Label: Depressive
```

Figure 5.4 Prediction of unseen data

As shown in Figure 5.4 the model predicts unseen input data correctly.

6. MODEL DEPLOYMENT

The principal objective of model deployment is to seamlessly integrate the models developed during the project into a production environment. This transition empowers end-users to access and leverage the system for making informed decisions. Model deployment is a critical phase that bridges the gap between theoretical model development and practical, real-world application. By deploying the models, the project aims to provide a tangible and accessible solution that fulfils the needs and preferences of the target audience.

To facilitate the deployment process, the project utilizes the python framework Flask and libraries. These tools contribute to the efficiency and effectiveness of model integration into the production environment.

User Interface (UI):

The depression detection system is integrated into the user interface, creating a cohesive and user-friendly experience. Users can interact with the system through the web interface.

The following figures shows user interface of this application. This interface is simple and easy to understand. This application's user interface is designed for straightforward interaction beginning with a clean introduction page of 'Depression Detection System'. The get started button leads you to page in which you need to enter the input text in the text column.
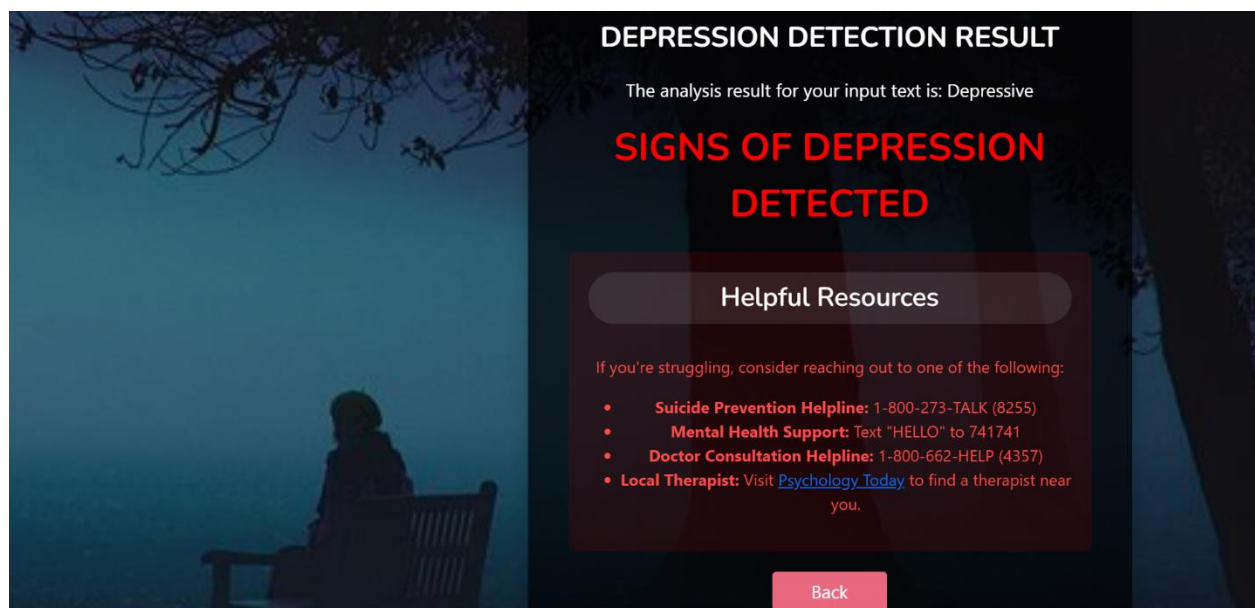
After clicking detect button, you will be routed to result page where you will be provided the classification result whether the text is depressive or not.
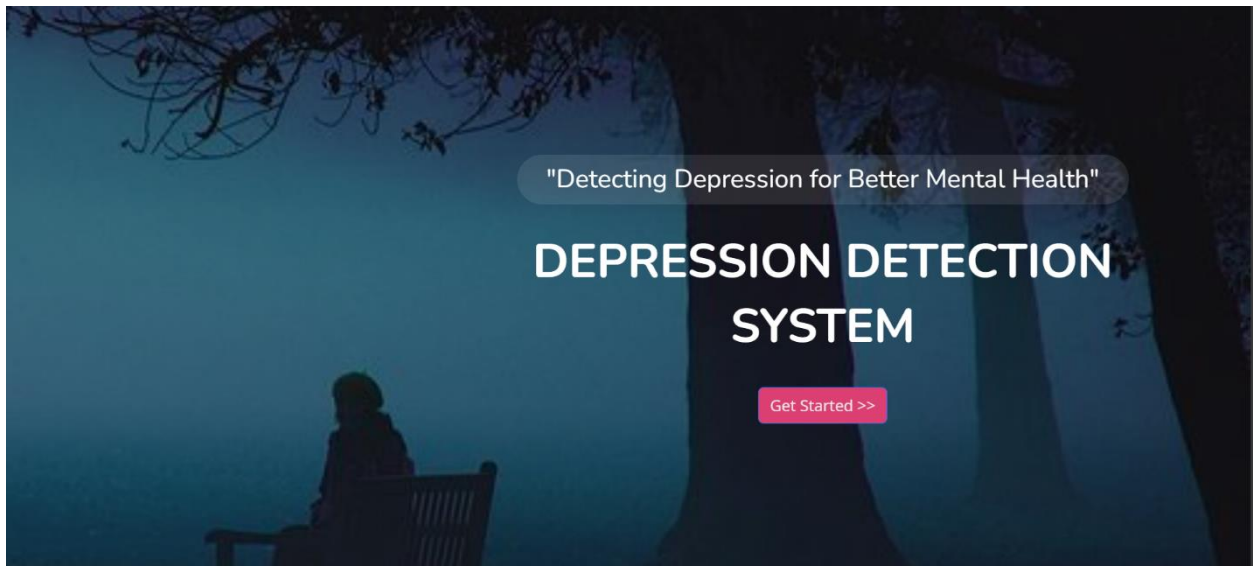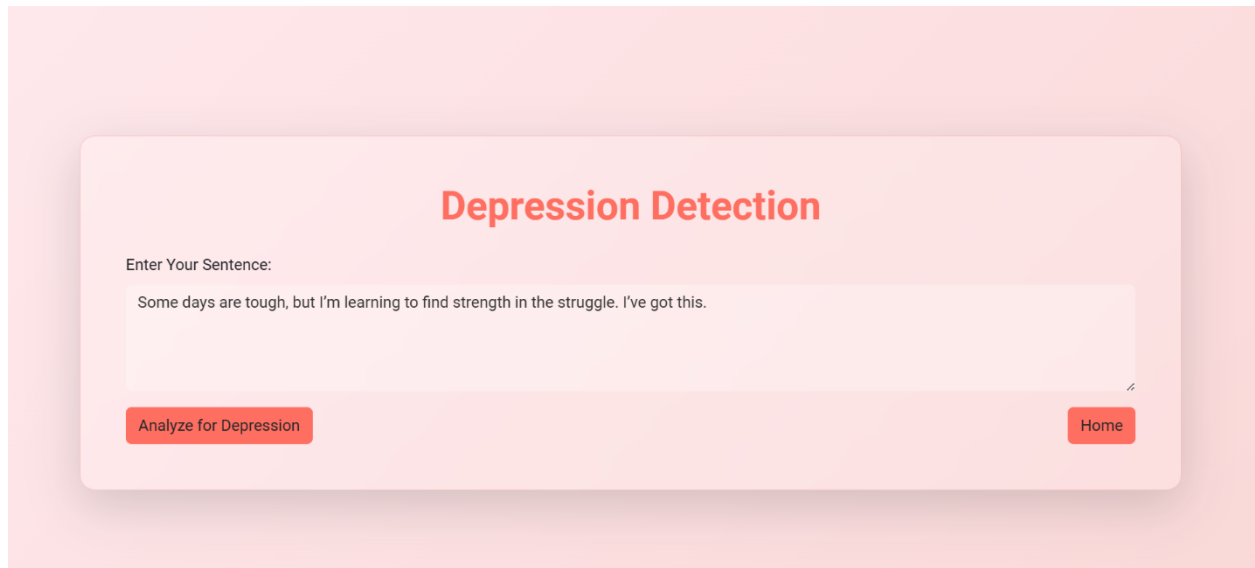

Figure 6.1 Homepage

Figure 6.2 Input Page of Depression Detection with a depressive text

Figure 6.3 Result Page of Depression Detection

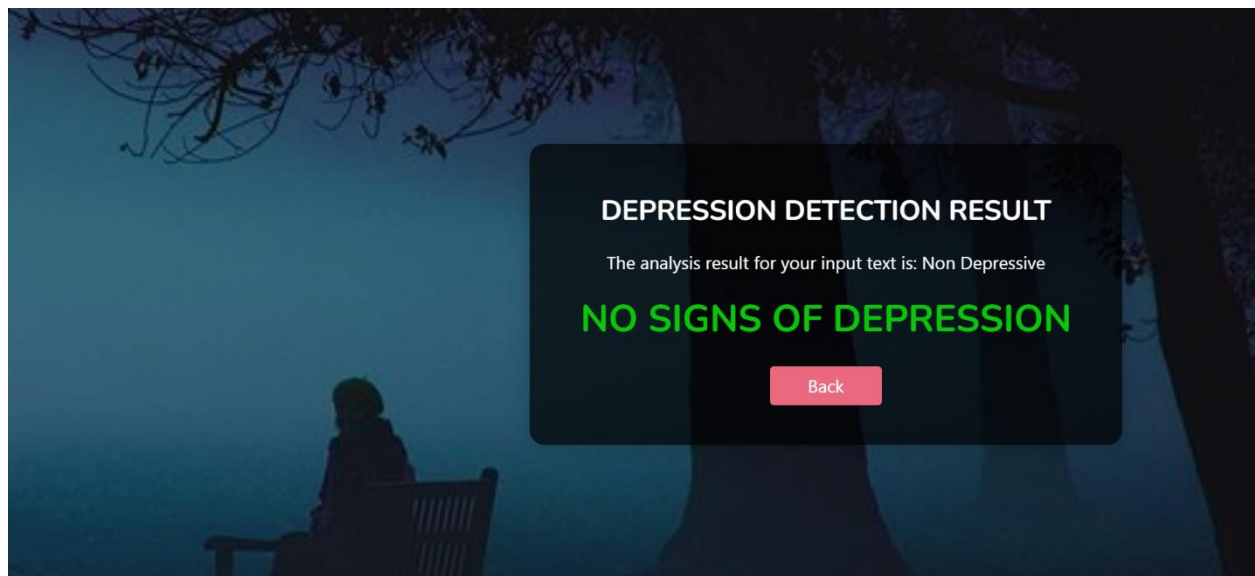Figure 6.3 depicts the output if a depressive text is given.

Figure 6.4 Input Page with non-depressive text

Figure 6.5 Result Page with Non depressive text

Figure 6.5 shows the result page with non-depressive text.

7. GIT HISTORY

Git Repository MINIPROJ-DEPRESSION-DETECTION-FROM-TEXTUAL-DATA-USING-ML contains all colab files, py files, html files and three related research papers. It is maintained for systematic way of project presentation and mainly for future reference. The colab files are created separately for three sprint releases during the project.
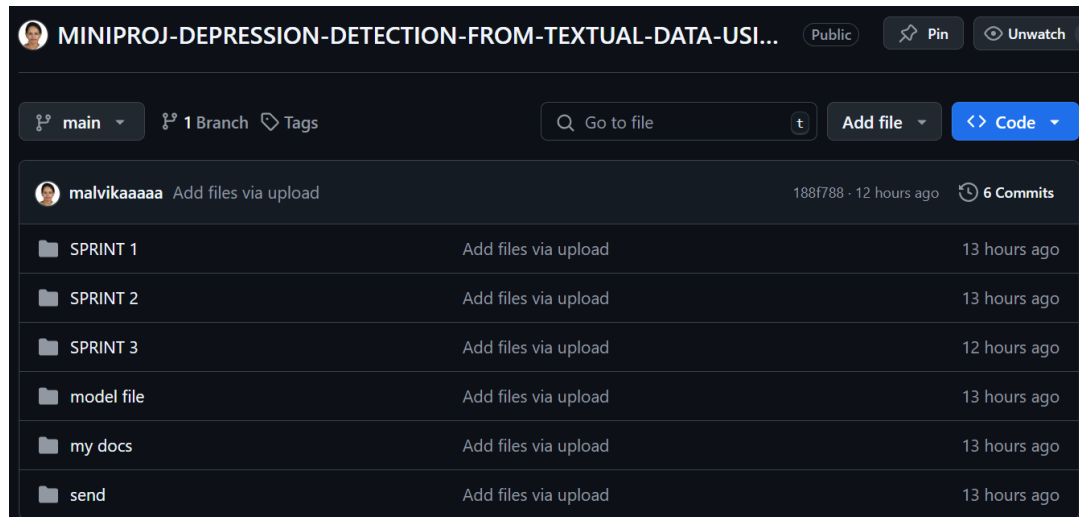
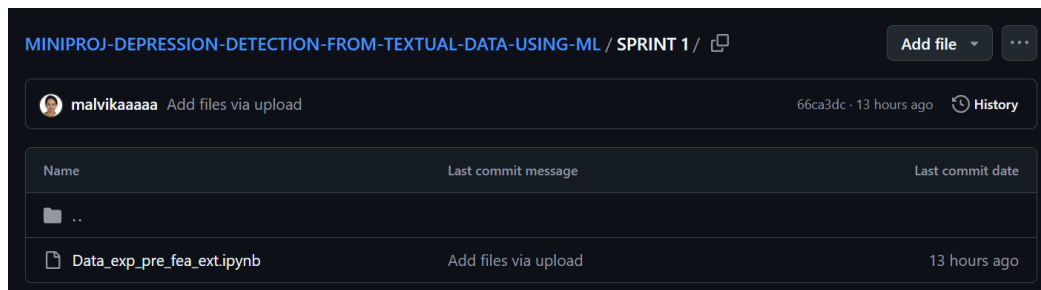

Figure 7.1 Git History of 3 Sprints
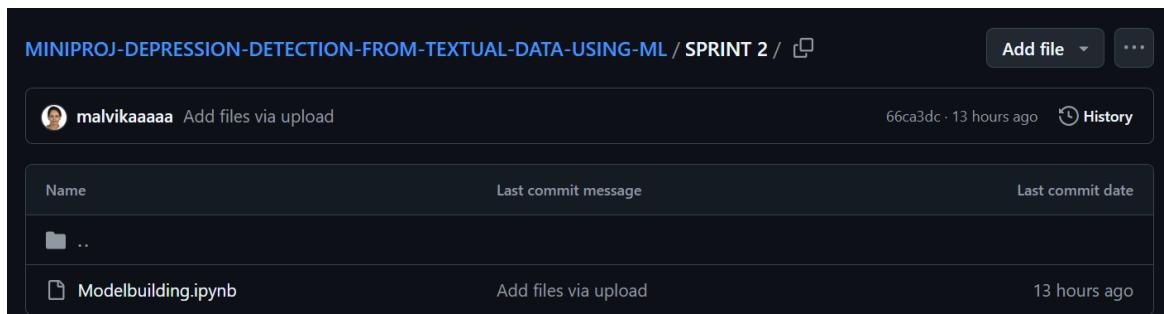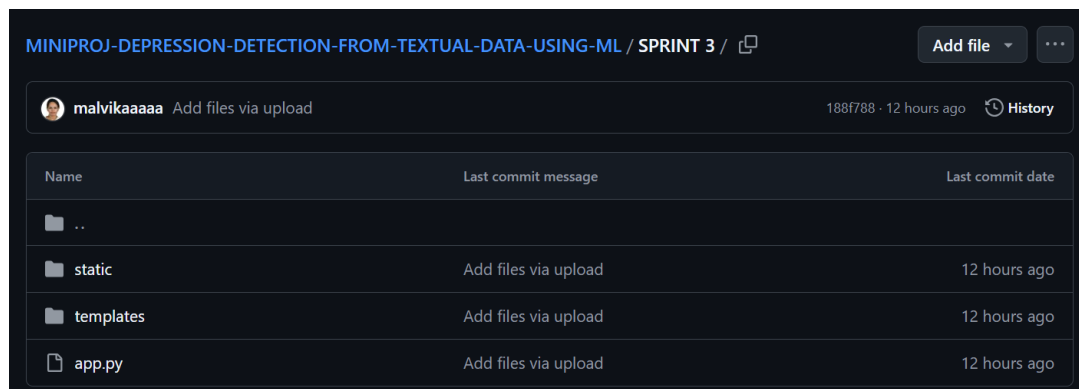

Figure 7.2 Git History of Sprint 1

Figure 7.3 Git History of Sprint 2

Figure 7.4 Git History of Sprint 3

8. CONCLUSION

In conclusion, the Depression Detection from Social Media project is positioned to contribute significantly to the early identification and support of individuals experiencing depression, leveraging the ubiquitous nature of social media as a platform for sharing emotions. Social media provides an informal yet powerful means for individuals to express their mental state, often reflecting deeper emotional struggles. By analysing these textual expressions with advanced machine learning techniques, this project seeks to build a reliable, accessible tool that can detect signs of depression, regardless of explicit keywords.

The system employs robust classifiers, including Multi-Layer Perceptron (MLP), Random Forest (RF), and Decision Trees (DT), and incorporates feature extraction methods such as N-gram and TF-IDF to identify depressive content with high accuracy. This approach goes beyond traditional keyword detection, synthesizing insights from literature on emotion detection to focus on four key types of features: emotional process, linguistic style, temporal process, and a combined feature set. This framework enables the system to detect subtle tones of sadness or depressive indicators, even in nuanced or indirect language, making it adaptable to the varied ways users may express themselves online.

To ensure accessibility and ease of use, the project will feature a web interface built using Flask. This interface will allow users to input social media text for evaluation, displaying results that indicate whether the content suggests depressive tendencies. Additionally, for users identified as exhibiting signs of depression, the system will provide suggested remedies or coping strategies in an adjacent text field. This feature aims to offer a practical form of support, encouraging users to seek help or engage in positive coping mechanisms.

Ultimately, this project represents a meaningful intersection of technology and mental health, offering a proactive solution to the growing need for mental health awareness and intervention in digital spaces. By combining machine learning with nuanced feature extraction methods, the Depression Detection system has the potential to foster early recognition and support for depression, contributing to improved mental well-being on both an individual and societal level. This system not only seeks to increase the accessibility of mental health support but also to promote a more compassionate, data-informed approach to mental health within the realm of social media.

9. FUTURE SCOPE

The future scope of the Depression Detection system includes expanding its capabilities, improving predictive accuracy, and enhancing its integration within real life situations.

- Incorporation of Additional Data Sources: Expanding the system to include supplementary data such as user activity patterns, frequency of posts, and engagement metrics could enhance diagnostic depth. By integrating these behavioural insights alongside textual analysis, the system could better assess nuanced indicators of mental health, potentially improving the detection of early or complex depression cases.

- Advanced Algorithms and Model Optimization: Exploring advanced algorithms like deep learning or recurrent neural networks (RNNs) could boost the model's ability to identify more intricate depressive patterns in text. Regular retraining on updated social media datasets and hyperparameter optimization would help maintain high accuracy, ensuring the model stays relevant amid evolving language trends and user behaviour.

- Real-Time Social Media Data Integration: With real-time integration capabilities, the system could analyze user posts as they are created, allowing for more immediate assessment and intervention. This feature would make the model useful for ongoing mental health monitoring, supporting early intervention and potentially alerting users to high-risk periods based on their social media activity.

- Scalability and Cloud Deployment: Cloud-based deployment would enable scalability, allowing the system to handle large volumes of social media data and reach users across various platforms. This approach would make the system accessible to a broader audience, including mental health organizations, clinics, and individuals in remote or underserved areas.

- Enhanced User Interface and Mobile Access: Developing a mobile-friendly interface with a focus on usability would improve access for both individuals and mental health professionals. Incorporating real-time feedback, intuitive navigation, and data visualization features could enhance user engagement, making the tool more effective for ongoing monitoring and personal mental health management.

10.APPENDIX

10.1. MINIMUM SOFTWARE REQUIREMENTS

To deploy and run the Thyroid Detection System, the following minimum software requirements must be met:

- Python:
    Version 3.6 or above is required to execute the project code and its dependencies.
- NumPy:
    Essential for numerical operations and array manipulations in Python.
- Pandas:
    Necessary for data manipulation, cleaning, and analysis.
- Scikit-learn:
    The scikit-learn library is used for machine learning functionalities, including the implementation of the Random Forest, Logistic Regression and Support Vector Machine.
- Matplotlib and Seaborn:
    Required for data visualization and generating plots and charts.
- Pickle:
    Used for saving and loading machine learning models.
- Colab Notebook:
    If using Colab notebooks for development and testing, having Colab installed is recommended.

10.2. MINIMUM HARDWARE REQUIREMENTS

The minimum hardware requirements to ensure smooth execution are as follows:

- Processor (CPU):
    A modern multi-core processor (e.g., Intel Core i3 or above) is recommended for optimal performance in thyroid detection system.
- RAM:
    A minimum of 8 GB RAM is recommended to manage the dataset and perform machine learning operations smoothly.
- Storage:
    Adequate storage space for storing datasets, code, and any additional resources. While the system itself doesn't require extensive storage, having sufficient space for datasets and potential model persistence is essential.
- Internet Connection:
    An internet connection is necessary for downloading datasets, libraries, and dependencies during the setup phase.

11.REFERENCES

1. Tadesse MM, Lin H, Xu B, Yang L. "Detection of depression-related posts in reddit social media forum." Ieee Access. 2019 Apr 4

2. Chiong R, Budhi GS, Dhakal S, Chiong F. "A textual-based featuring approach for depression detection using machine learning classifiers and social media texts." Computers in Biology and Medicine. 2021 Aug 1

3. Islam MR, Kabir MA, Ahmed A, Kamal AR, Wang H, Ulhaq A. "Depression detection from social network data using machine learning techniques." Health information science and systems. 2018 Dec

4. https://www.kaggle.com/datasets/hyunkic/twitter-depression-dataset

5. https://www.kaggle.com/datasets/bababullseye/depression-analysis

6. https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/

7. https://www.geeksforgeeks.org/decision-tree-introduction-example/

8. https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/?ref=header_ind

9. https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/

10. Text book on Machine Learning with R- Brett Lantz

11. Text book on Data Science-Concepts and Practice-Vijay Kotu and Bala Deshpande