Analyzing Social Media Posts for Mental Health Disorder Detection

Submitted by

SOUMYADEEP NANDY (13000121033)

PRITHWISH SARKAR (13000121037)

SAGNIK MUKHOPADHYAY (13000121040)

ARKAPRATIM GHOSH (13000121058)

⟨ Group 29 ⟩

Final Year 7th Semester

⟨ September, 2024 ⟩

Submitted for the partial fulfillment for the degree of
Bachelor of Technology in
Computer Science and Engineering

Techno Main Salt Lake,
EM 4/1, Salt lake, Sector - V, Kolkata - 700091

Department of Computer Science and Engineering
Techno Main Salt Lake
Kolkata - 700 091
West Bengal, India

APPROVAL

This is to certify that the project entitled **"Analyzing Social Media Posts for Mental Health Disorder Detection"** prepared by **SOUMYADEEP NANDY** *(13000121033)*, **PRITHWISH SARKAR** *(13000121037)*, **SAGNIK MUKHOPADHYAY** *(13000121040)* and **ARKAPRATIM GHOSH** *(13000121058)* be accepted in partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

. . . .
(Signature of the Internal Guide)

. . . .
(Signature of the HOD)

. . . .
(Signature of the External Examiner)

. . . .

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

. . . .

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide in the department of Computer Science and Engineering. We are extremely thankful for the keen interest our guide took in advising us, for the books, reference materials and support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staffs for the gracious hospitality they offered us.

Place: Techno Main Salt Lake
Date:

SOUMYADEEP NANDY (13000121033)

PRITHWISH SARKAR (13000121037)

SAGNIK MUKHOPADHYAY (13000121040)

ARKAPRATIM GHOSH (13000121058)

Table of Content

List of Figures

Abstract

This project aims to analyze social media posts for early detection of mental health disorders. Specifically, it focuses on using machine learning and deep learning algorithms to classify social media text data based on potential mental health issues. The problem lies in efficiently detecting patterns that indicate mental health conditions within large, unstructured datasets. Using methods like Logistic Regression , XGboost the project seeks to enhance the accuracy of detecting mental health concern with a specific probability. Expected results include a robust classifier capable of distinguishing between different mental health concerns with high accuracy. This project could provide a valuable tool for mental health monitoring on social platforms.

1 Introduction

1.1 Project Overview

Mental health has become a critical global issue, with millions of people affected by various mental disorders, including depression, anxiety, and others. With the increasing use of social media, these platforms have emerged as spaces where people often express their emotions and struggles, sometimes unknowingly revealing signs of mental health challenges. This project focuses on analyzing social media posts to detect mental health disorders using advanced machine learning techniques. By examining language patterns and contextual usage in text data, this project aims to classify posts that potentially indicate mental health issues. Such detection can facilitate early intervention and help direct individuals to appropriate mental health services.

1.2 Project Purpose

The main goal of this project is to leverage machine learning and deep learning to identify the best model and create a web applicatoin capable of identifying signs of mental health disorders from text, images and social media posts by giving username as input. This aligns with a broader goal of using technology to address public health concerns by enabling early detection through data analysis. Specifically, we use classification models such as Logistic Regression and XGboost to predict mental health issues based on text patterns. The project also addresses the technical challenges of processing large datasets and optimizing algorithms for accurate classification.

1.3 Technical Domain Specifications

This project falls within the intersection of natural language processing (NLP) and machine learning (ML), leveraging techniques such as text vectorization, and classification algorithms. Here are the key technical domain specifications:

- **Hardware** : The project does not require specialized hardware beyond a standard machine with adequate processing power. However, for larger datasets or complex model training, a machine equipped with a GPU (Graphics Processing Unit) could significantly reduce processing time. The project can be run on any system with at least 8GB of RAM and a multi-core processor.
- **Operating System** : The project is cross-platform and can be developed and executed on any modern operating system, including:
  - Windows 10/11
  - macOS
  - Linux distributions (Ubuntu, Linux Mint, etc.) A Linux-based system is often preferred in machine learning projects due to its stability and support for tools like TensorFlow, PyTorch, and other libraries used for model training.
- **Software** :
  - **Programming Languages** : Python 3.x will be the primary programming language, given its extensive libraries for machine learning, data analysis, and NLP.
  - **Libraries / Frameworks** :
    - **Scikit-learn** : Used for machine learning algorithms (k-NN, SVM) and model evaluation.
    - **Pandas** : For data manipulation and preprocessing.
    - **NumPy** : To handle large arrays and matrices, which are crucial for efficient numerical computations.
    - **NLTK and spaCy** : For text preprocessing and natural language understanding.
    - **Matplotlib and Seaborn** : For data visualization.
  - **Development Environment** :
    - **Jupyter Notebook** : For interactive development, experimentation, and visualization.
    - **Anaconda** : A distribution that simplifies package management and deployment.

* **Google Colab** : For cloud-based execution when working with larger datasets or GPU-based model training.

1.4 Business Domain Specifications

From a business perspective, this project holds significant value across various sectors, particularly those that intersect with mental health monitoring, public health awareness, and social media governance. With the increasing prevalence of mental health issues globally, organizations within these industries are searching for innovative solutions to mitigate the growing mental health crisis. Leveraging machine learning for early detection of mental health disorders from social media data can revolutionize how mental health is addressed at both individual and societal levels. Below is a detailed exploration of how this project can impact different business domains:

- **Mental Health Services** : Mental health service providers—such as hospitals, therapy centers, and private practices—can greatly benefit from machine learning models capable of identifying early signs of mental health issues from social media data. In the traditional mental health setting, early detection of disorders like depression or anxiety often relies on self-reporting or clinical assessments, which may come too late in the progression of the disorder. By analyzing patterns in social media posts, these services can adopt a more proactive approach, reaching out to potential patients earlier in their mental health journey.

- **Social Media Platforms** : Social media platforms like Twitter, Facebook, Instagram, and others play an integral role in the public's expression of thoughts and feelings, including mental health struggles. These platforms face increasing pressure to safeguard the well-being of their users. This project's machine learning models can enable these companies to offer valuable services to users while adhering to ethical standards.

- **Public Health Organizations** : Public health organizations are tasked with monitoring and improving the mental well-being of the population on a large scale. For these organizations, access to real-time data from social media can provide a comprehensive view of the mental health landscape, identifying emerging trends and enabling data-driven interventions. Understanding how mental health is being discussed online can help public health organizations create more effective mental health awareness campaigns. Tailored messaging based on the language patterns identified by the model can lead to better engagement with individuals suffering from mental health issues.

1.5 Glossary / Keywords

Term	Definition
Machine Learning (ML)	A subset of artificial intelligence (AI) that enables computers to learn from data and make predictions or decisions without explicit programming.
Natural Language Processing (NLP)	A branch of artificial intelligence focused on the interaction between computers and humans through natural language, including tasks like text analysis.
Support Vector Machines (SVM)	A supervised learning algorithm used for classification or regression tasks, focusing on finding a hyperplane that best separates different classes.
Vectorization	The process of converting textual data into numerical form (such as a vector) so that it can be used as input for machine learning models.
Classifier	A machine learning model or algorithm that categorizes or labels data points into predefined classes.
Mental Health Disorder	A wide range of conditions that affect mood, thinking, and behavior, including depression, anxiety, schizophrenia, etc.
Data Preprocessing	The process of preparing raw data for analysis by cleaning, normalizing, and transforming it into a usable format for machine learning models.
Cross-validation	A model validation technique used to assess how well a model performs by dividing data into training and testing sets multiple times for better accuracy.
Precision	In the context of classification, precision refers to the accuracy of positive predictions, calculated as the ratio of true positives to the sum of true and false positives.
Recall	In classification, recall measures the ability of a model to identify all relevant instances within a dataset, calculated as the ratio of true positives to the sum of true positives and false negatives.
PRAW	PRAW (Python Reddit API Wrapper) is a Python library that provides a simple interface to interact with Reddit's API, allowing developers to easily access, retrieve, and analyze Reddit data, such as posts, comments, and user information.
TesseractOCR	TesseractOCR is an open-source Optical Character Recognition (OCR) engine that extracts text from images with high accuracy; it is widely used for various applications like scanning documents and digitalizing printed text.

Term	Definition
Depression	There is a difference between depression and mood swings or short-lived emotional reactions to daily experiments; A mental state causing painful symptoms adversely disrupts normal activities (e.g., sleeping).
Anxiety	Several behavioral disturbances are associated with anxiety disorders, including excessive fear and worry. Severe symptoms cause significant impairment in functioning cause considerable distress. Anxiety disorders come in many forms, such as social anxiety, generalized anxiety, panic, etc.
Bipolar Disorder	An alternating pattern of depression and manic symptoms is associated with bipolar disorder. An individual experiencing a depressive episode may feel sad, irritable, empty, or lose interest in daily activities. Emotions of euphoria or irritability, excessive energy, and increased talkativeness can all be signs of manic depression. Increased self-esteem, decreased sleep need, disorientation, and reckless behavior may also be signs of manic depression.
Post-Traumatic Stress Disorder (PTSD)	In PTSD, persistent mental and emotional stress can occur after an injury or severe psychological shock, characterized by sleep disturbances, constant vivid memories, and dulled response to others and the outside world. People who re-experience symptoms may have difficulties with their everyday routines and experience significant impairment in their performance.

2 Related Studies

The intersection of social media analytics and mental health research has received increasing attention in recent years, leading to several important studies that highlight the potential for early detection and intervention. This section reviews key findings from various studies, emphasizing the relevance and applicability of social media data for identifying mental health disorders.

One of the seminal works in this domain is by Choudhury et al. (2013), who explored the predictive capabilities of social media content in identifying depression. They analyzed Twitter data and discovered that specific linguistic patterns, such as the use of negative emotion words, correlated strongly with self-reported depressive symptoms. This study demonstrated that social media could serve as a valuable resource for predicting mental health conditions, offering a potential tool for clinicians and researchers alike [2].

Similarly, Guntuku et al. (2017) conducted an integrative review that focused on detecting mental illness through social media. Their work synthesized various approaches and methodologies used in the field, providing insights into the effectiveness of different machine learning algorithms and sentiment analysis techniques. They found that social media platforms are rich sources of data that can reveal critical information about users' mental health, advocating for the development of robust systems to analyze this data effectively [3].

A systematic review by Mathur et al. (2022) further emphasized the significance of mental health classification on social media. They examined various studies that utilized machine learning techniques for mental health detection, highlighting the success of these models in identifying depression and anxiety based on user-generated content. Their findings reinforced the notion that social media can be leveraged not only for individual assessments but also for broader epidemiological studies to understand population mental health trends [4].

In addition, Nadeem (2016) contributed to the discussion by investigating depression identification on Twitter. The study focused on developing algorithms that could discern emotional cues in tweets, indicating a user's mental state. The findings revealed that simple text analysis could lead to significant improvements in identifying at-risk individuals, further validating the potential of social media data in mental health monitoring [5].

Research by AlSagri and Ykhlef (2020) introduced a machine learning-based approach specifically for depression detection on Twitter. Their study incorporated both content and activity features, demonstrating that a combination of linguistic and behavioral analysis could enhance the accuracy of depression identification. This work illustrated the multifaceted nature of social media data and its ability to capture not just what users say but also how they interact online [1].

In a more recent study, Vaishnavi et al. (2022) investigated the application of various machine learning algorithms for predicting mental health illnesses. They found that certain algorithms outperformed others in classifying mental health conditions based on social media posts. This study provided a comparative analysis that could inform future research directions, emphasizing the importance of algorithm selection in the context of mental health detection [7].

Lastly, Safa et al. (2023) presented a roadmap for future development in predicting mental health using social media. Their work highlighted the ongoing challenges in the field, including ethical considerations and the need for improved data privacy measures. They emphasized that while social media offers rich data for mental health analysis, researchers must approach this opportunity with a strong ethical framework to ensure user safety and data security [6].

These studies collectively underscore the growing body of evidence supporting the integration of social media analytics and machine learning for mental health detection. They provide a solid foundation for the current project, which aims to enhance existing methodologies and develop a predictive model for identifying mental health disorders from social media posts.

3 Problem Definition and Preliminaries

3.1 Context and Background

Mental health disorders have become a significant public health concern worldwide. The World Health Organization (WHO) estimates that approximately 1 in 8 people globally experience mental health disorders, which encompass conditions such as depression, anxiety, bipolar disorder, and post-traumatic stress disorder (PTSD). The rise of social media platforms has changed how individuals express their mental health struggles, share experiences, and seek support. Posts on platforms like Reddit and Twitter provide a wealth of data reflecting real-time sentiments, issues, and conversations surrounding mental health. However, this vast amount of unstructured textual data presents challenges in effectively identifying and categorizing specific mental health disorders.

3.2 Objective

The primary objective of this project is to develop a robust system that can automatically analyze social media posts, specifically from Reddit and Twitter, to detect various mental health disorders. By leveraging Natural Language Processing (NLP) techniques and machine learning algorithms, this project aims to:

- **Classify Posts** : Accurately classify social media posts based on the type of mental health disorder mentioned, including but not limited to depression, anxiety, bipolar disorder, and PTSD.
- **Data Driven Insights** : Provide valuable insights into the prevalence and expression of mental health issues on social media, helping researchers, mental health professionals, and policymakers understand trends and patterns.

3.3 Challenges

- **Data Variability** : Social media posts can vary significantly in structure, style, and length. Users may employ slang, abbreviations, and informal language, making it difficult for algorithms to accurately interpret and classify posts.

- **Imbalanced Data** : Certain mental health issues may be underrepresented in social media discussions, leading to an imbalanced dataset. This imbalance can adversely affect model training and performance, making it harder to detect less frequent disorders.
- **Cultural and Contextual Nuances** : Mental health perceptions and discussions can vary across different cultures and contexts. The model needs to account for these nuances to avoid misclassification and provide accurate insights.
- **Privacy and Ethical Considerations** : Analyzing social media data raises ethical concerns regarding user privacy. It is crucial to handle sensitive information responsibly and comply with data protection regulations.

3.4 Scope

The scope of the project ”Analyzing Social Media Posts for Mental Health Disorder Detection” delineates the specific aspects that will be covered, the methodologies employed, and the boundaries within which the research and analysis will occur. The project aims to harness the potential of machine learning and natural language processing (NLP) techniques to analyze social media sentiment and its correlation with mental health disorders, focusing specifically on a Reddit Dataset sourced using Python Reddit API Wrapper. This dataset contains user-generated content that reflects various emotional states, making it a valuable resource for this analysis.

- **Dataset Selection and Characteristics**
  The primary data source for this project is the top textual posts from Reddit. This dataset includes posts that are labeled with mental health issues (Normal, Anxiety, Depression, Bipolar, PTSD). The selection of this dataset is pivotal, as it encapsulates a wide range of mental health-related discussions expressed by individuals on social media. Key characteristics of the dataset include:
  - **User Anonymity** : To respect user privacy and adhere to ethical standards, the dataset does not contain personally identifiable information (PII) about the Twitter users. This ensures compliance with data protection regulations while allowing for robust analysis.
- **Analysis Objectives**
  The project will focus on several key objectives:
  - **Correlation with Mental Health Issues** : The analysis will explore the correlation between identified sentiments and specific mental health disorders, thereby contributing to the understanding of how social media discourse reflects mental health challenges.

  - **Trend Analysis** : By analyzing sentiment trends over time, the project seeks to identify patterns in public discourse surrounding mental health, including any potential spikes in negative sentiments during particular events or crises.

- **Methodologies** : The project will employ various methodologies to achieve its objectives, including:

  - **Data Processing** : Cleaning and preparing the dataset to ensure that it is suitable for analysis. This includes tasks such as removing noise (e.g., URLs, hashtags), tokenization, and normalization of text.
  - **Feature Extraction** : Utilizing techniques like Term Frequency-Inverse Document Frequency (TF-IDF) to convert textual data into numerical representations suitable for machine learning algorithms.
  - **Machine Learning Techniques** : Implementing various machine learning algorithms, including Logistic Regression and XGboost to classify posts and evaluate their performance based on accuracy, precision, recall, and F1 score.
  - **Data Visualizations** : Employing visualization tools to present findings clearly, including heat maps for confusion matrices and ROC AUC Curve.

3.5 Exclusions

In delineating the boundaries of the project "Analyzing Social Media Posts for Mental Health Disorder Detection," it is crucial to specify what is excluded from the scope of this research to maintain a clear focus on the primary objectives. This project will not encompass the direct collection or real-time monitoring of Twitter data via the Twitter API, as it is solely reliant on the pre-existing Twitter sentiment dataset obtained from Kaggle. Therefore, any analysis involving the dynamic aspects of social media engagement, such as real-time sentiment shifts in response to current events or trending topics, will be outside the project's purview. Furthermore, the study will not address the technicalities of Reddit's platform-specific features, such as hashtags, user mentions, or reposts, subreddits in detail, as these elements are not central to the primary research focus on mental issue classification of individual users. While the project aims to analyze posts surrounding mental health, it will also include mapping the mental issue with mental wellbeing. Additionally, the research will not explore the ethical implications of data ownership or the responsibilities of social media platforms regarding user-generated content, as the focus will be primarily on data analysis techniques and outcomes rather than the broader ethical landscape.

3.6 Assumptions

In the context of the project ”Analyzing Social Media Posts for Mental Health Disorder Detection,” several key assumptions underpin the research framework and methodologies employed. Firstly, it is assumed that the Reddit dataset obtained using PRAW is representative of broader social media discourse regarding mental health issues, capturing a diverse range of sentiments expressed by users on the platform. This assumption is critical as it establishes the foundation for analyzing sentiment trends and their potential correlations with various mental health disorders. Secondly, it is presumed that the posts recorded in the dataset accurately reflect the users' true emotions and perspectives at the time of posting, thereby providing valid data for analysis. Furthermore, it is assumed that the textual data within the dataset can be effectively processed and interpreted through natural language processing (NLP) techniques, allowing for the accurate classification of sentiments and identification of patterns. Another assumption is that the selected machine learning algorithms, including Logistic Regression and XGboost, will perform optimally with the provided data, leading to reliable and interpretable results regarding sentiment analysis and mental health correlations. Additionally, it is assumed that the sentiments expressed in social media posts can serve as a valid proxy for understanding public perceptions of mental health, enabling insights into societal attitudes and the potential stigmatization associated with these disorders. The project also assumes that the cleaning and preprocessing steps applied to the data will sufficiently prepare the dataset for analysis, minimizing noise and irrelevant information that could skew the results. Lastly, it is presumed that the ethical considerations surrounding the use of publicly available social media data have been adequately addressed, ensuring that the research adheres to relevant ethical standards and does not compromise user privacy or data integrity. These assumptions serve as the bedrock for the project's analytical framework, guiding the research processes and interpretations that follow.

4 Proposed Solution

The proposed work centers around ”Analyzing Social Media Posts for Mental Health Disorder Detection,” leveraging advanced data analytics and machine learning techniques to provide insights into the sentiments expressed in social media discussions related to mental health issues. This research is significant due to the increasing prevalence of mental health disorders globally and the role social media plays in shaping public perception and discourse around these issues. The core objective of this project is to develop a systematic approach to classify sentiments in tweets, thereby enabling better understanding and awareness of mental health conditions through social media analysis. Below, I outline the specific contributions and reusable components deployed in this project.

4.1 Special Contributions

- **Dataset Acquisition and Preparation** : The initial step involved sourcing a high-quality dataset from Kaggle, specifically the Twitter sentiment dataset. This dataset comprises user-generated tweets containing sentiments related to various mental health issues, serving as the primary data source for analysis. The preparation phase included extensive data cleaning and preprocessing, wherein missing values were handled, duplicate entries removed, and text normalization performed. This crucial step ensured the dataset's integrity and suitability for subsequent analysis, allowing for a more accurate representation of sentiments.

- **Text Vectorization** : To enable machine learning models to interpret textual data, TF-IDF was implemented. This approach involved converting posts into a numerical format by creating a matrix representation of word frequencies across the dataset. The Scikit-learn library was instrumental in this process, offering functions for text vectorization and feature extraction. The reusable components for text preprocessing and vectorization were packaged into functions, allowing for easy application to future datasets or similar projects.

- **Implementation of Machine Learning Algorithms** : The project employed several machine learning algorithms, focusing primarily on Logistic Regression and XGboost for smental health classification. The Logistic Regression algorithm was chosen for its simplicity and effectiveness in classifying data points based on proximity in the feature space. In addition to Logistic Regression, XGboost was implemented due to its robustness in handling high-dimensional data and multi class classification tasks.

- **Model Evaluation** : Comprehensive evaluation metrics were employed to assess the performance of the machine learning models. Metrics such as accuracy, precision, recall, and F1-score were calculated to provide a holistic view of model effectiveness in classifying sentiments related to mental health. This evaluation process not only demonstrated the models' capabilities but also highlighted areas for improvement, providing a foundation for future iterations of the project. The evaluation framework, including metrics calculations and visualization, was designed as reusable components to streamline future model assessments.

- **Insights and Recommendations** : A critical aspect of this project is the generation of actionable insights based on the analysis of social media sentiments. The findings from the classification can inform mental health professionals, researchers, and policymakers about public sentiment trends, potential stigma associated with mental health issues, and the effectiveness of awareness campaigns. Recommendations for mental health awareness strategies can be derived from understanding how sentiments vary across different demographics and regions. This interpretative analysis, combined with quantitative results, contributes valuable knowledge to the ongoing conversation about mental health in society.

- **Documentation and Reproducibility** : To enhance the usability and impact of the project, thorough documentation was maintained throughout the research process. This documentation includes detailed explanations of the methodologies employed, code snippets, and instructions for reproducing the results. The aim is to ensure that the components developed in this project can be easily utilized by other researchers and practitioners in the field. By documenting the code and methodologies, I am contributing to the open-source community, allowing for collaborative improvements and innovations in mental health sentiment analysis.

4.2 Reusable Components

- **Data Collection Functions** : Modular functions designed for data collection, which can be reused across different platforms.
- **Data preprocessing Module** : A component that does data cleaning to remove the duplicates and empty rows and add a seperate column for cleaned texts. This formatted dataset is then used further with TF-IDF to create a numerical matrix to be fed to the machine leanring algorithms.
- **Machine and Deep Learning Model Functions** : Functions for implementing Logistic Regression, Naive Bayes, Support Vector Machine, Random Forest, XGboost, Long Short Term Memory algorithm allowing for easy retraining on varying datasets. These also features various evaluation metrics, making it easy to assess different models' performances.
- **Deployment Function** : A seperate function that has the main python file for creating web based application on Streamlit Cloud. This also includes the requirements and package dependencies for deploying the application.

5 Project Planning

5.1 Software Life Cycle Model

In developing the project, I adopted an iterative approach to the Waterfall model, enabling a structured yet flexible framework for managing the various phases of the project. The project plan outlines specific tasks, dependencies, timelines, and milestones to ensure a systematic progression towards the final goal of analyzing social media posts for mental health disorder detection.

The Iterative Waterfall model was chosen for this project due to its structured approach while allowing for iterative revisions and refinements. Unlike the traditional Waterfall model, which emphasizes a linear progression through distinct phases, the iterative variant permits revisiting earlier stages based on findings and feedback. This flexibility is particularly beneficial in data-driven projects where insights gained during the analysis may necessitate adjustments to earlier stages, such as refining requirements or enhancing data preparation techniques.

In this project, the iterative nature of the Waterfall model facilitated ongoing improvement and adaptation throughout the development process. For example, initial results from the model evaluation phase may prompt a revisit to data preprocessing to enhance data quality or to explore alternative modeling techniques. This approach ultimately fosters a more robust final product, ensuring that the developed system meets the dynamic needs of mental health disorder detection in social media posts.

The key feature of the iterative approach is the feedback loop that exists between the phases. For instance, after completing the testing phase, if certain models do not meet performance expectations, the project can loop back to the implementation phase. This allows for modifications to the models, preprocessing techniques, or even revisiting the requirements to ensure alignment with the project's objectives.

The project is divided into distinct phases:

- **Requirement Gathering and Analysis** : This initial phase involved understanding the project's goals, objectives, and stakeholder expectations. It spanned approximately two weeks, culminating in a detailed requirements document that guided the subsequent stages.
- **Data Collection and Preparation** : Utilizing a Twitter sentiment dataset from Kaggle and Reddit API, the data collection phase was executed over a week. This included downloading the dataset, examining its structure, and performing data cleaning and prepossessing to ensure its suitability for analysis.
- **Model Development** : This phase, lasting about three weeks, included the creation of a Bag of Words model, splitting the dataset into training and test sets, and implementing various machine learning algorithms such as k-Nearest Neighbors (k-NN) and Support Vector Machines (SVM) for sentiment classification.

- **Model Evaluation** : Following model development, a week was allocated for rigorous testing and validation of the models, ensuring they met the required accuracy benchmarks. This phase involved using performance metrics such as accuracy, precision, recall, and F1-score to evaluate the models' effectiveness.

- **Final Deployment and Documentation** : The last phase, spanning two weeks, focused on deploying the best-performing model and creating comprehensive documentation. This included user manuals and technical documentation to facilitate future maintenance and enhancements.

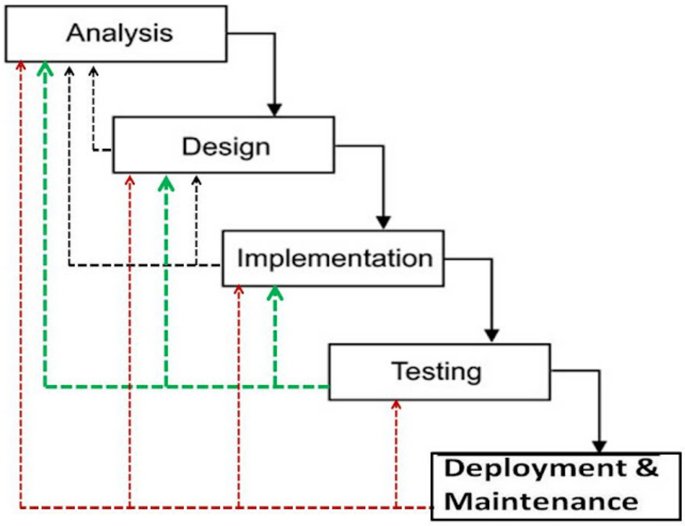

Figure 1: Iterative Waterfall Model

5.2 Dependencies and Milestones

Key dependencies were identified for successful project progression. For instance, completion of the data preparation phase was critical before proceeding to model development. Milestones were established at the end of each phase to ensure accountability and track progress. The successful completion of the requirement gathering phase marked the first milestone, followed by the data preparation phase, and so on.

5.3 Scheduling

Effective scheduling is crucial to the success of any project, as it establishes a clear timeline for tasks, milestones, and dependencies. In the context of our project on detecting mental health disorders through social media analysis, a detailed schedule has been developed to guide the project from inception to completion. This schedule includes specific tasks such as requirement gathering, data preprocessing, model implementation, testing, and deployment, each with clearly defined deadlines. The iterative nature of our chosen methodology allows for flexibility within the schedule, enabling adjustments based on testing outcomes and stakeholder feedback. Key milestones, such as the completion of data analysis, model validation, and user acceptance testing, have been identified to monitor progress and ensure timely delivery of the final product. By utilizing project management tools, such as Microsoft Project, we can visualize and track the progress of tasks, manage resources effectively, and maintain open communication among team members, ensuring that the project stays on schedule and meets its objectives. This proactive approach to scheduling enhances our ability to deliver a high-quality solution that aligns with our goals and stakeholder expectations.

Task Mode	Task Name	Duration	Start	Finish	Predecessors
	Analyzing Social Media Posts for Mental Health Disorder Detection	**109 days**	**Wed 03-07-24**	**Sat 30-11-24**	
	Feasibility Study	6 days	Wed 03-07-24	Wed 10-07-24	
	Requirements Analysis	**15 days**	**Thu 11-07-24**	**Wed 31-07-24**	
	Requirements Gathering	9 days	Thu 11-07-24	Tue 23-07-24	2
	Analysis Of Requirements	6 days	Wed 24-07-24	Wed 31-07-24	2
	Design	**15 days**	**Thu 01-08-24**	**Wed 21-08-24**	
	High Level Design	8 days	Thu 01-08-24	Mon 12-08-24	5
	Low Level Design	7 days	Tue 13-08-24	Wed 21-08-24	7
	Coding	**30 days**	**Thu 22-08-24**	**Wed 02-10-24**	
	Data Collection and Preprocessing	10 days	Thu 22-08-24	Wed 04-09-24	8
	Model Development	15 days	Thu 05-09-24	Wed 25-09-24	10
	Refining Models	1 day	Thu 26-09-24	Thu 26-09-24	11
	Testing	**25 days**	**Thu 03-10-24**	**Wed 06-11-24**	
	Unit Testing	5 days	Thu 03-10-24	Wed 09-10-24	12
	Integration Testing	5 days	Thu 10-10-24	Wed 16-10-24	14
	System Testing	9 days	Thu 17-10-24	Tue 29-10-24	15
	Acceptance Testing	6 days	Thu 17-10-24	Thu 24-10-24	15
	Deployment and Documentation	**13 days**	**Thu 07-11-24**	**Mon 25-11-24**	
	Deployment	8 days	Thu 07-11-24	Mon 18-11-24	17
	Documentation	5 days	Tue 19-11-24	Mon 25-11-24	19
	Delivery	1 day	Tue 26-11-24	Tue 26-11-24	20

Figure 2: Project Plan

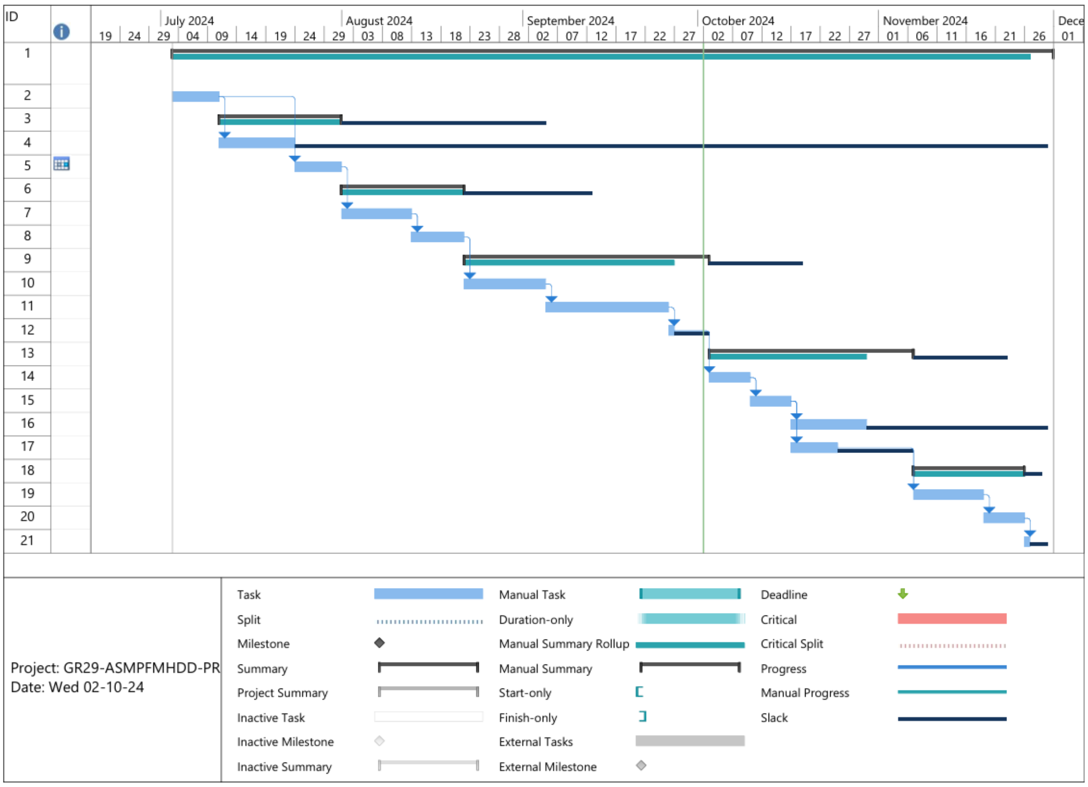

Figure 3: Gantt Chart

Requirement ID	Requirement Description	Priority	Category	Source	Status	Dependencies	Assigned To	Verification
FR-001	Collect and preprocess social media data from Kaggle and Reddit.	High	Functional	Stakeholder	Completed	FR-002	Data Team	Data Analysis Review
FR-002	Implement data cleaning and feature extraction for NLP.	High	Functional	Project Docs	Completed	FR-001	Development Team	Feature Verification
FR-003	Train machine learning and deep learning models for mental health classification.	High	Functional	Stakeholder	In Progress	FR-001, FR-002	ML Team	Model Accuracy Testing
FR-004	Evaluate models using performance metrics (accuracy, recall, F1).	High	Functional	Stakeholder	In Progress	FR-003	ML Team	Performance Report
NFR-001	Document all technical and user aspects of the final solution.	Medium	Non Functional	Stakeholder	In Progress	FR-004	Documentation Team	Peer Review
NFR-002	Ensure model deployment is done on a scalable platform.	Medium	Non Functional	Project Docs	Not Started	FR-003, FR-004	Development Team	System Testing

Figure 4: Requirement Matrix

6 Requirement Analysis

6.1 Requirement Matrix

The Requirement Matrix is a comprehensive tool used to track and manage the key requirements of a project. It systematically organizes each requirement with a unique identifier, description, priority, and category (such as functional or non-functional). The matrix also records the source of the requirement, its current status (e.g., in progress, completed), any dependencies on other requirements, and the team or individual responsible for fulfilling it. Additionally, it includes a verification method to ensure the requirement is met, such as through testing or review. This structured format helps ensure that all requirements are clearly defined, prioritized, and tracked, enabling effective project management and ensuring alignment with stakeholder expectations.

6.2 Requirement Elaboration

6.2.1 Functional Requirements

Requirement ID: FR-001
Description: Data Collection
Priority: High
Category: Functional
The system requires an ability to collect and ingest a large dataset of Reddit posts using Python Reddit API Wrapper. The data should include text content from tweets, associated sentiment labels, and other metadata such as timestamp and user details. This will serve as the primary source of information for mental health disorder detection. The system must ensure that the dataset is loaded correctly into the machine learning environment, and any discrepancies in the structure should be handled with pre-processing steps like cleaning, normalization.

Requirement ID: FR-002
Description: Data Cleaning and Preprocessing
Priority: High
Category: Functional
The system must include modules to clean the raw data, such as removing irrelevant characters, handling missing data, and tokenizing text. For the social media posts, it is essential to remove URLs, stopwords, and unnecessary punctuation. The pre-processing pipeline should also convert the text into a suitable numerical format using Term Frequency-Inverse Document Frequency (TF-IDF) for further analysis. Proper pre-processing ensures that the data is in a form that can be efficiently used by machine learning models.

Requirement ID: FR-003
Description: Sentiment and Disorder Detection Model
Priority: High
Category: Functional
The system needs to implement machine learning algorithms such as Logistic Regression and XGboost to classify social media posts based on text and detect potential signs of mental health disorders. The system must be able to train these models on historical data and then apply them

to predict the sentiment and detect mental health-related issues in new posts.

Requirement ID: FR-004
Description: Model Validation and Evaluation
Priority: High
Category: Functional
The system must evaluate the performance of the trained models by splitting the dataset into training and test sets. Various performance metrics like accuracy, precision, recall, and F1-score should be computed to assess the model's effectiveness in detecting mental health disorders. Based on the evaluation, the system should allow for model fine-tuning, such as adjusting hyperparameters, to improve the overall performance.

6.2.2 Non Functional Requirements

Requirement ID: NFR-001
Description: Documentation
Priority: Medium
Category: Non Functional
The addition of documentation and maintenance manuals as a non-functional requirement ensures that the system is not only usable in the short term but also maintainable and extensible over time. This guarantees that future updates and improvements to the system can be implemented without disrupting existing functionality or requiring a steep learning curve for new developers or users.

Requirement ID: NFR-002
Description: Scalability
Priority: Medium
Category: Non Functional
The system must be designed to efficiently process large volumes of data, considering the potential growth in the amount of social media posts that need to be analyzed. The data processing pipeline should be scalable to handle increasing data size without significant degradation in performance. This could involve implementing parallel processing techniques or leveraging cloud-based infrastructure to ensure that processing large datasets remains feasible even as the dataset scales.

7 Design

7.1 Technical Environment

The technical environment for the project ”Analyzing Social Media Posts for Mental Health Disorder Detection” comprises a combination of hardware, software, and tools that enable smooth data analysis, machine learning model training, and deployment. Below is a detailed overview of the minimum hardware configuration, software tools, and package details necessary to carry out this project effectively.

Minimum Hardware Configuration
Given the nature of the project, which involves processing textual data and training machine learning models, the hardware requirements are modest but significant enough to ensure optimal performance. The minimum configuration needed is:

- **Processor** : Intel Core i5 (or equivalent) with a base clock speed of at least 2.5 GHz. A multi-core processor is preferred as it helps in parallel processing, which is essential during model training and data preprocessing steps.
- **RAM** : 8 GB of RAM is recommended to handle the operations of data loading, cleaning, and transformation. Large datasets, like those used in this project, may require more memory to prevent memory overflow errors and reduce delays during processing. For larger datasets, 16 GB of RAM would be ideal.
- **Storage** : At least 256 GB of SSD storage is recommended. Faster storage access significantly impacts loading time for datasets and dependencies. SSD is preferred over traditional HDD because of its faster read/write speeds, which benefit large datasets like the Reddit-based social media posts used in this project.
- **Graphics Processing Unit (GPU)** : For basic machine learning tasks like Logistic Regression or SVM, a dedicated GPU is not necessary. However, if deep learning models or more complex neural networks were introduced later, a GPU like NVIDIA GTX 1060 with 4 GB VRAM or higher would be advantageous.
- **Operating System** : Windows 10 (64-bit) or higher, macOS 10.13 (High Sierra) or higher, or any stable Linux distribution (e.g., Ubuntu 18.04 or higher). The operating system should support all necessary machine learning libraries and be compatible with the tools required for the project.

Software Tools and Packages

For the software stack, the project leverages a set of well-established tools, platforms, and programming libraries to ensure smooth execution from data preprocessing to model deployment:

- **Python** : The primary programming language used for data processing, model training, and evaluation. Python is chosen due to its rich ecosystem of libraries and frameworks tailored for machine learning and data science.
- **Pandas** : A powerful library for data manipulation and analysis, essential for data preprocessing tasks, such as handling missing values and restructuring datasets.
- **scikit-learn** : A comprehensive machine learning library in Python used for implementing and comparing algorithms in classification, regression, and clustering, along with various model evaluation tools.
- **Streamlit** : An open-source Python framework that facilitates the deployment of machine learning models and interactive web applications.
- **Pyngrok** : A Python wrapper for ngrok, used to create secure tunnels to locally deployed applications, which is particularly useful for sharing Streamlit applications over the web.
- **Google Colab** : A cloud-based platform used for writing, executing, and sharing Python code, with access to GPU and TPU resources, beneficial for model training. It integrates seamlessly with libraries like TensorFlow and PyTorch.
- **PRAW (Python Reddit API Wrapper)** : A Python library that allows for easy interaction with the Reddit API to access, retrieve, and analyze Reddit data, such as posts, comments, and user information.
- **pytesseract** : A Python wrapper for Tesseract OCR, used to extract text from images. It's essential for converting image-based text data into a format suitable for processing.
- **Pillow** : A Python Imaging Library that adds support for image processing, which aids in handling image files for text recognition tasks with pytesseract.
- **joblib** : A library for efficient serialization and deserialization of Python objects, especially useful for saving and loading machine learning models during deployment.
- **protobuf** : A protocol buffer library by Google used for serializing structured data, helpful in efficient data exchange between applications.
- **deep-translator** : A library that facilitates easy translation across different languages, enabling multilingual processing of text data.

- **Requests** : A user-friendly library for making HTTP requests, used to retrieve data from APIs or web resources as part of data collection.
- **google-generativeai** : A Python client library for Google's generative AI models, providing tools to integrate and utilize AI functionalities within the project.

7.2 Hierarchy of Modules

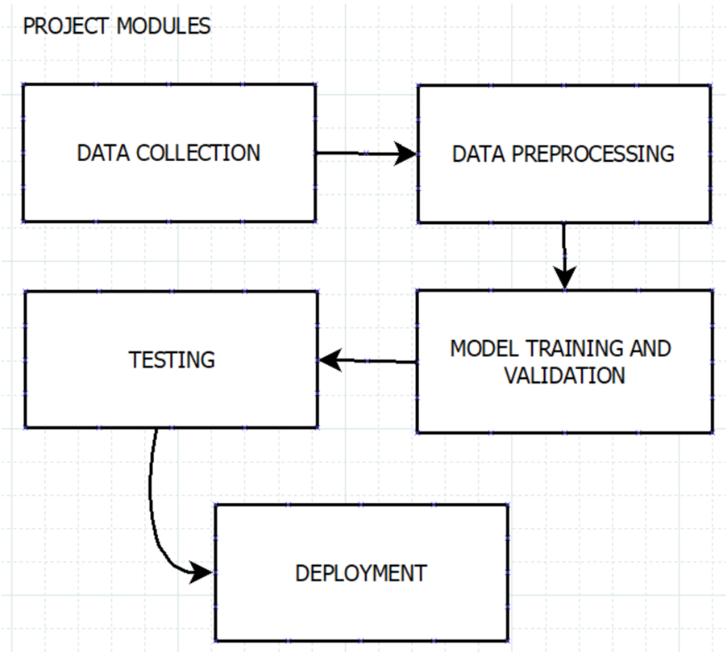

Figure 5: Project Modules

In this project, the system is structured into key modules to classify mental health issues based on text input effectively. The **Data Collection Module** gathers relevant text data, building a comprehensive dataset from sources like CSV files or platforms such as Reddit via PRAW. Next, the **Data Preprocessing Module** loads and cleans this data through tokenization, stop-word removal, and lemmatization, preparing it for analysis. Following this, the **Model Training and Validation Module** converts the text into numerical features using techniques like TF-IDF, splitting the data into training and validation sets to test various machine learning models, including Logistic Regression, Naive Bayes, SVM, Random Forest, XGboost and LSTM. Finally, the **Testing and Deployment Module** allows real-time predictions by deploying the model on platforms like Streamlit Cloud, providing an accessible solution for practical applications.

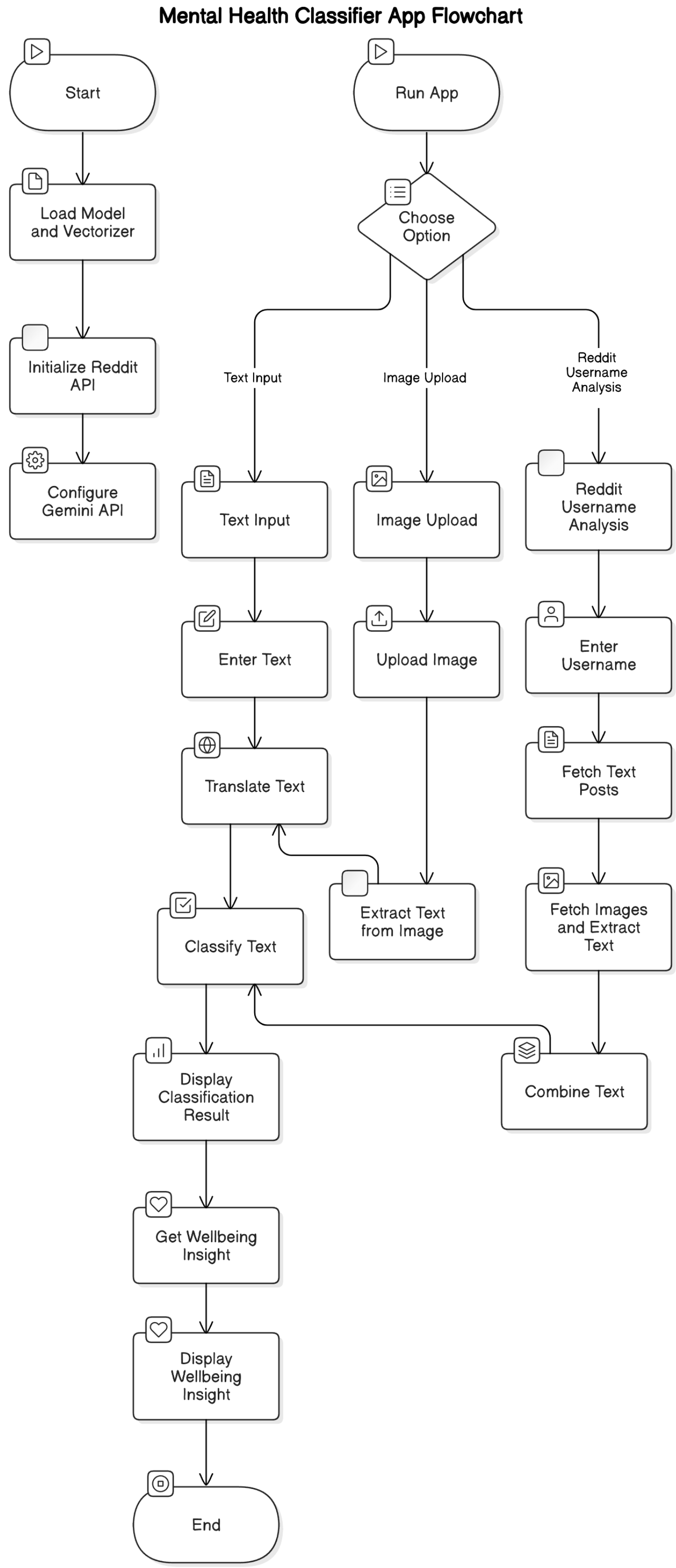

Figure 6: System Overview

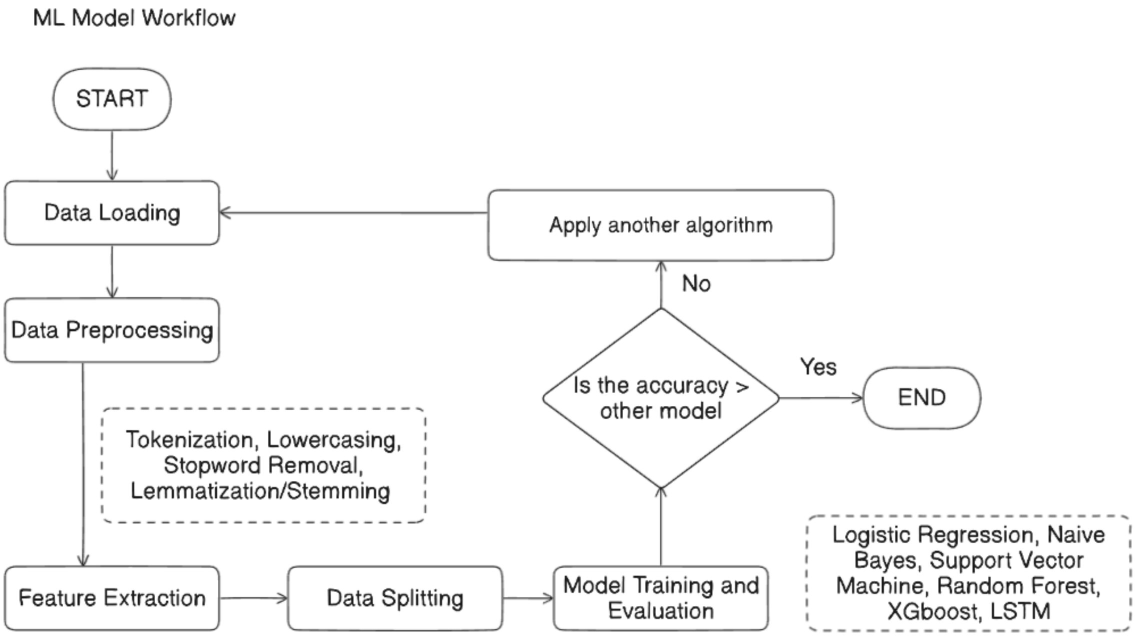

Figure 7: Model Workflow

7.3 Detailed Design

7.3.1 Data Loading and Preprocessing

The Data Loading and Preprocessing Module is the foundation of the system, responsible for ingesting and preparing the text data for analysis. This module begins by loading the dataset from the preprocessed_mental_health_text.csv file, which contains various mental health-related text entries. Once the data is loaded, a series of preprocessing steps are conducted to ensure the text is clean and ready for feature extraction. This includes tokenization, where the text is split into individual words or tokens, and lowercasing to maintain uniformity across the dataset. Additionally, stop-word removal is performed to eliminate common words that do not contribute to the meaning, such as ”and,” ”the,” and ”is.” Finally, lemmatization or stemming is applied to reduce words to their base or root forms. These preprocessing techniques are crucial as they help improve the quality of the input data, ultimately leading to better model performance.

7.3.2 Feature Extraction

In the Feature Extraction Module, the preprocessed text data is transformed into a numerical format that machine learning algorithms can process. This module allows for the selection between two primary feature extraction methods: Term Frequency-Inverse Document Frequency (TF-IDF). The TF-IDF approach evaluates the importance of words in the dataset by considering their frequency in individual documents relative to their overall occurrence across all documents. This helps in highlighting the most informative words. By converting text into numerical features, this module prepares the data for the subsequent training and validation stages, ensuring that the classification models can effectively interpret the input.

7.3.3 Model Training and Validation

The Model Training and Validation Module is critical to developing a robust classification system. In this module, the dataset is split into training and testing sets to evaluate the performance of the models accurately. Various classification algorithms are employed, including Logistic Regression, Naive Bayes, Support Vector Machines, Random Forest, XGboost and LSTM. Each model is trained on the training set, which involves adjusting the model parameters based on the input features and their corresponding labels. Following training, the models undergo validation to assess their performance using various metrics such as accuracy, precision, recall, and F1-score. A decision point is included to determine if the achieved accuracy meets the project requirements. If the accuracy is deemed acceptable, the model is accepted.

7.3.4 Prediction

The Prediction Module is designed to provide real-time classification of new input text related to mental health issues. Upon receiving user input, this module initiates a preprocessing workflow that mirrors the steps applied during the training phase, including tokenization, lowercasing, stop-word removal, and lemmatization or stemming. Once the input text is preprocessed, it is fed into the trained classification models to generate predictions. Each model may provide a classification result, allowing for a comprehensive analysis of the input. This module not only delivers the predicted mental health issue but also ensures that users receive an informative output that reflects the confidence level of each prediction, enabling them to understand the model's reasoning. The seamless integration of this module into the overall system enhances the user experience by providing instant and relevant feedback.

7.3.5 Testing and Deployment

The module focuses on making the trained models accessible for real-time predictions. Once the models have been validated and selected based on their performance, this module prepares them for deployment on suitable platforms like Streamlit Cloud. This involves packaging the models and creating a user interface where users can input text and receive predictions. The deployment process also includes considerations for scaling, ensuring that the system can handle multiple requests simultaneously while maintaining responsiveness. By providing a free and efficient deployment solution, this module enables users to access the mental health classification service easily. The deployment of the models ensures that the insights generated from the analysis can be utilized effectively in real-world applications.

8 Implementation

8.1 Features From RM

For the initial prototype development, a subset of the requirements from the Requirement Matrix (RM) was carefully selected to focus on implementing core functionalities and demonstrating proof of concept. The selection was based on a combination of high-priority functional requirements that form the backbone of the system, ensuring that critical features are built and validated before expanding the scope.

The requirements chosen for the prototype primarily involve data preprocessing, model training, and evaluation processes. These requirements were selected because they are fundamental to the project's success, ensuring that the data pipeline and model implementation work seamlessly together. This subset of features lays the groundwork for later integration with additional components and more complex functionality.

The filtered part of the RM focuses on the following high-priority requirements: data cleaning and feature extraction, training of machine learning models, and evaluation of model performance using standard metrics. These requirements were identified as crucial because they directly impact the system's ability to handle data, learn patterns, and provide meaningful outputs. Without successfully implementing these core features, the overall effectiveness of the solution would be significantly reduced.

Furthermore, these selected features align with the project's goals and provide a clear pathway for incremental development. By narrowing down the requirements to these foundational aspects, the development team can ensure that the prototype is not only functional but also extensible, providing a robust framework for future enhancements.

	A	B	C	D
1	**Requirement ID**	**Requirement Description**	**Priority**	**Category**
2	FR-001	Collect and preprocess social media data from Kaggle and Reddit.	High	Functional
3	FR-002	Implement data cleaning and feature extraction for NLP.	High	Functional
4	FR-003	Train machine learning and deep learning models (k-NN, SVM) for sentiment analysis.	High	Functional
5	FR-004	Evaluate models using performance metrics (accuracy, recall, F1).	High	Functional

Figure 8: Features from Requirement Matrix

8.2 Code Details and Output

8.2.1 Data Collection

Collecting Reddit Posts

```
import praw
import pandas as pd
import time

# Initialize Reddit API
reddit = praw.Reddit(client_id='YOUR_CLIENT_ID',
                     client_secret='YOUR_CLIENT_SECRET',
                     user_agent='Mental␣Health␣Classifier')

# Define subreddits and post types
subreddits = {'normal': ['news', 'AskReddit'],
              'depression': ['depression'],
              'ptsd': ['PTSD'],
              'anxiety': ['Anxiety'],
              'bipolar': ['BipolarReddit']}
post_types = ['hot', 'new', 'top']
posts_per_type = 100

# Collect and save posts
data = []
for label, subs in subreddits.items():
    for sub in subs:
        for post_type in post_types:
            posts = getattr(reddit.subreddit(sub), post_type
                )(limit=posts_per_type)
            for post in posts:
                data.append([post.title + "␣" + post.
                    selftext, label])
            time.sleep(1)

df = pd.DataFrame(data, columns=['text', 'label'])
df.to_csv(f'{label}_dataset.csv', index=False)
```

Combining Collected Datasets

```
import pandas as pd
from sklearn.utils import shuffle

# Load datasets
bipolar_df = pd.read_csv("bipolar_dataset.csv")
depression_df = pd.read_csv("depression_dataset.csv")
normal_df = pd.read_csv("normal_dataset.csv")
anxiety_df = pd.read_csv("anxiety_dataset.csv")
ptsd_df = pd.read_csv("ptsd_dataset.csv")

# Minimum length for balanced classes
min_length = min(len(bipolar_df), len(depression_df), len(
   normal_df) // 6, len(anxiety_df), len(ptsd_df))

# Create balanced pattern
pattern_data = []
for i in range(min_length):
    pattern_data.extend([bipolar_df.iloc[i], depression_df.
       iloc[i]] +
                        normal_df.iloc[i*6:(i+1)*6].to_dict(
                           'records') +
                        [anxiety_df.iloc[i], ptsd_df.iloc[i
                           ]])

pattern_df = pd.DataFrame(pattern_data)
remaining_data = shuffle(pd.concat([bipolar_df[min_length:],
    depression_df[min_length:],
                                    normal_df[min_length
                                       *6:], anxiety_df[
                                       min_length:], ptsd_df
                                       [min_length:]]))

final_df = pd.concat([pattern_df, remaining_data]).
   reset_index(drop=True)
final_df.to_csv("mental_health_combined.csv", index=False)
```

The first code snippet collects Reddit posts by connecting to specific subreddits associated with various mental health topics. Using the Reddit API, posts are retrieved from ‘hot‘, ‘new‘, and ‘top‘ categories for each mental health type, including general subreddits (for the ”normal” category). Each post’s title and content are combined, labeled, and saved into separate CSV files. The second code snippet merges these CSV files to form a combined dataset. It creates a balanced sample by selecting the minimum number of rows across each category and organizes the data in a structured pattern. Remaining rows are shuffled and appended, producing a final dataset suitable for machine learning applications in mental health classification.

8.2.2 Data Preprocessing

Text Preprocessing

```
import pandas as pd
import re
from sklearn.feature_extraction.text import
   TfidfVectorizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk

# Download stopwords (if you haven't already)
nltk.download('stopwords')
nltk.download('punkt')

# Load the dataset
df = pd.read_csv('mental_health.csv')

# 1. Handling Missing Values
df.dropna(subset=['text'], inplace=True)

# 2. Removing duplicates (if any)
df.drop_duplicates(subset=['text'], inplace=True)

# 3. Text Preprocessing
negative_words = {"not", "no", "nor", "never", "nothing"
   , "nowhere", "neither", "cannot", "n't", "without", "
   barely", "hardly", "scarcely"}

def clean_text(text):
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'@\w+', '', text)  # Remove mentions
       (@username)
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove
       special characters, numbers, and punctuations
    text = text.lower()  # Convert text to lowercase
    tokens = word_tokenize(text)  # Tokenize the text
    tokens = [word for word in tokens if word not in
       stopwords.words('english') or word in
       negative_words]  # Remove stopwords, but keep
       negative words
    clean_text = ' '.join(tokens)  # Join the tokens
       back into a single string
    return clean_text

df['cleaned_text'] = df['text'].apply(clean_text)  #
   Apply the cleaning function to the 'text' column
```

Text Preprocessing

```
# 4. Feature Extraction using TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=10000)  #
    Adjust the max_features
X = vectorizer.fit_transform(df['cleaned_text'])  # Fit
    and transform the cleaned text data
X_df = pd.DataFrame(X.toarray(), columns=vectorizer.
    get_feature_names_out())  # Convert the result to a
    DataFrame for easier understanding

# Save the preprocessed dataset (optional)
df.to_csv('preprocessed_mental_health.csv', index=False)
```

The code snippet above demonstrates the preprocessing steps for a mental health dataset. It handles missing values and duplicates in the text data, performs text cleaning by removing URLs, mentions, and special characters, and applies tokenization. The cleaned text is then vectorized using TF-IDF to convert the textual data into a format suitable for machine learning models. Finally, the preprocessed dataset is saved as a new CSV file for future use in classification tasks.

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
   aaa  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa   ab  abandon  abandoned  \
0  0.0                                 0.0  0.0      0.0        0.0
1  0.0                                 0.0  0.0      0.0        0.0
2  0.0                                 0.0  0.0      0.0        0.0
3  0.0                                 0.0  0.0      0.0        0.0
4  0.0                                 0.0  0.0      0.0        0.0

   abandoning  abandonment  abc  abilities  ability  ...  zemeckis  zemlja  \
0         0.0          0.0  0.0        0.0  0.00000  ...       0.0     0.0
1         0.0          0.0  0.0        0.0  0.03726  ...       0.0     0.0
2         0.0          0.0  0.0        0.0  0.00000  ...       0.0     0.0
3         0.0          0.0  0.0        0.0  0.00000  ...       0.0     0.0
4         0.0          0.0  0.0        0.0  0.00000  ...       0.0     0.0

   zero  zoloft  zombie  zombieland  zombies  zone  zoom  zuckerberg
0   0.0     0.0     0.0         0.0      0.0   0.0   0.0         0.0
1   0.0     0.0     0.0         0.0      0.0   0.0   0.0         0.0
2   0.0     0.0     0.0         0.0      0.0   0.0   0.0         0.0
3   0.0     0.0     0.0         0.0      0.0   0.0   0.0         0.0
4   0.0     0.0     0.0         0.0      0.0   0.0   0.0         0.0

[5 rows x 10000 columns]
```

Figure 9: Data Collection and Preprocessing

8.2.3 Logistic Regression Model for Classification

Logistic Regression for Mental Health Classification

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,
    classification_report, confusion_matrix

# Load the preprocessed dataset
dataset = pd.read_csv('preprocessed_mental_health.csv')

# Check if 'cleaned_text' and 'mental_health_issue' columns
    exist
if 'cleaned_text' not in dataset.columns or '
    mental_health_issue' not in dataset.columns:
     raise ValueError("The dataset must have 'cleaned_text'
        and 'mental_health_issue' columns.")

# Remove rows with missing values in 'cleaned_text' column
dataset.dropna(subset=['cleaned_text'], inplace=True)

# Initialize the CountVectorizer and fit/transform the
    cleaned text
LRvectorizer = CountVectorizer()
X = LRvectorizer.fit_transform(dataset['cleaned_text'])

# Prepare the target variable
y = dataset['mental_health_issue']

# Split the dataset into Training and Test Sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

# Initialize the Logistic Regression model
LRmodel = LogisticRegression(multi_class='ovr', max_iter
    =10000)

# Train the model
LRmodel.fit(X_train, y_train)

# Make predictions on the test set
y_pred = LRmodel.predict(X_test)
```

Logistic Regression for Mental Health Classification

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Print classification report
print("Classification Report:\n", classification_report(
    y_test, y_pred))

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred
    ))
```

The provided code demonstrates how to apply Logistic Regression for mental health classification using a preprocessed dataset. First, the dataset is loaded, and a check is performed to ensure that it contains the necessary columns, specifically `cleaned_text` and `mental_health_issue`. If these columns are missing, an error is raised. The dataset is then cleaned by removing rows that have missing values in the `cleaned_text` column using the `dropna()` function. Next, the `CountVectorizer` is initialized and applied to the `cleaned_text` column to convert the text data into a numerical format suitable for machine learning. This is accomplished by transforming the text into a document-term matrix where each row represents a document (post) and each column represents a unique term (word). The target variable, `mental_health_issue`, is also extracted from the dataset.The dataset is then split into training and testing sets using `train_test_split()`, where 80% of the data is used for training, and 20% is reserved for testing. The `random_state` is set to ensure reproducibility of the results. A Logistic Regression model is initialized with a multi-class strategy (`multi_class='ovr'`) and a high number of iterations (`max_iter=10000`) to allow the model to converge. The model is then trained on the training data using the `fit()` method. After training, predictions are made on the test set using the `predict()` method. The model's performance is evaluated by calculating the accuracy score, which is printed as a percentage. Additionally, a detailed classification report is generated, which includes metrics such as precision, recall, and F1-score for each class. Lastly, a confusion matrix is printed to visualize the model's classification performance and to understand how well the model distinguishes between different mental health issues. This end-to-end pipeline is crucial for applying Logistic Regression to textual data, allowing for effective classification of mental health issues based on Reddit posts.

8.2.4 Naive Bayes for Classification

Naive Bayes for Mental Health Classification

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
   CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,
   classification_report, confusion_matrix

# Load the preprocessed dataset
dataset = pd.read_csv('preprocessed_mental_health.csv')

# Check if 'cleaned_text' and 'mental_health_issue'
   columns exist
if 'cleaned_text' not in dataset.columns or '
   mental_health_issue' not in dataset.columns:
    raise ValueError("The␣dataset␣must␣have␣'
       cleaned_text'␣and␣'mental_health_issue'␣columns."
       )

# Remove rows with missing values in 'cleaned_text'
   column
dataset.dropna(subset=['cleaned_text'], inplace=True)

# Initialize the CountVectorizer and fit/transform the
   cleaned text
NBvectorizer = CountVectorizer()
X = NBvectorizer.fit_transform(dataset['cleaned_text'])

# Prepare the target variable
y = dataset['mental_health_issue']

# Split the dataset into Training and Test Sets
X_train, X_test, y_train, y_test = train_test_split(X, y
   , test_size=0.2, random_state=42)

# Initialize the Naive Bayes classifier
NBmodel = MultinomialNB()

# Fit the model
NBmodel.fit(X_train, y_train)
```

Naive Bayes for Mental Health Classification

```
# Make predictions
y_pred = NBmodel.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Print classification report
print("Classification Report:\n", classification_report(
   y_test, y_pred))

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test,
   y_pred))
```

The provided code demonstrates how to apply the Naive Bayes classifier (MultinomialNB) for mental health classification using a preprocessed dataset. First, the dataset is loaded, and a check is performed to ensure that it contains the necessary columns, specifically `cleaned_text` and `mental_health_issue`. If these columns are missing, an error is raised. The dataset is then cleaned by removing rows that have missing values in the `cleaned_text` column using the `dropna()` function. Next, the `CountVectorizer` is initialized and applied to the `cleaned_text` column to convert the text data into a numerical format suitable for machine learning. This is accomplished by transforming the text into a document-term matrix where each row represents a document (post) and each column represents a unique term (word). The target variable, `mental_health_issue`, is also extracted from the dataset. The dataset is then split into training and testing sets using `train_test_split()`, where 80% of the data is used for training, and 20% is reserved for testing. The `random_state` is set to ensure reproducibility of the results. A Naive Bayes model (`MultinomialNB`) is initialized and trained on the training data using the `fit()` method. After training, predictions are made on the test set using the `predict()` method. The model's performance is evaluated by calculating the accuracy score, which is printed as a percentage. Additionally, a detailed classification report is generated, which includes metrics such as precision, recall, and F1-score for each class. Lastly, a confusion matrix is printed to visualize the model's classification performance and to understand how well the model distinguishes between different mental health issues. This end-to-end pipeline is crucial for applying the Naive Bayes algorithm to textual data, allowing for effective classification of mental health issues based on Reddit posts.

8.2.5 Support Vector Machine for Classification

Support Vector Classifier Implementation

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,
   classification_report, confusion_matrix

# Load the preprocessed dataset
dataset = pd.read_csv('preprocessed_mental_health.csv')

# Check if 'cleaned_text' and 'mental_health_issue' columns
   exist
if 'cleaned_text' not in dataset.columns or '
   mental_health_issue' not in dataset.columns:
    raise ValueError("The␣dataset␣must␣have␣'cleaned_text'␣
       and␣'mental_health_issue'␣columns.")

# Remove rows with missing values in 'cleaned_text' column
dataset.dropna(subset=['cleaned_text'], inplace=True)

# Initialize the CountVectorizer and fit/transform the
   cleaned text
SVMvectorizer = CountVectorizer()
X = SVMvectorizer.fit_transform(dataset['cleaned_text'])

# Prepare the target variable
y = dataset['mental_health_issue']

# Split the dataset into Training and Test Sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)

# Initialize the Support Vector Classifier
SVMmodel = SVC(kernel='linear', C=1, random_state=42,
   probability=True)  # You can adjust parameters as needed

# Train the model
SVMmodel.fit(X_train, y_train)

# Make predictions on the test set
y_pred = SVMmodel.predict(X_test)
```

Support Vector Classifier Implementation

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy:␣{accuracy␣*␣100:.2f}%')

# Print classification report
print("Classification␣Report:\n", classification_report(
   y_test, y_pred))

# Print confusion matrix
print("Confusion␣Matrix:\n", confusion_matrix(y_test, y_pred
   ))
```

The provided code demonstrates how to apply the Support Vector Classifier (SVC) for mental health classification using a preprocessed dataset. First, the dataset is loaded, and a check is performed to ensure that it contains the necessary columns, specifically `cleaned_text` and `mental_health_issue`. If these columns are missing, an error is raised. The dataset is then cleaned by removing rows that have missing values in the `cleaned_text` column using the `dropna()` function.Next, the `CountVectorizer` is initialized and applied to the `cleaned_text` column to convert the text data into a numerical format suitable for machine learning. This is accomplished by transforming the text into a document-term matrix where each row represents a document (post) and each column represents a unique term (word). The target variable, `mental_health_issue`, is also extracted from the dataset. The dataset is then split into training and testing sets using `train_test_split()`, where 80% of the data is used for training, and 20% is reserved for testing. The `random_state` is set to ensure reproducibility of the results. A Support Vector Classifier model (`SVC`) is initialized with a linear kernel (`kernel='linear'`), regularization parameter `C=1`, and the probability flag set to `True` to enable probability estimates. The model is then trained on the training data using the `fit()` method. After training, predictions are made on the test set using the `predict()` method. The model's performance is evaluated by calculating the accuracy score, which is printed as a percentage. Additionally, a detailed classification report is generated, which includes metrics such as precision, recall, and F1-score for each class. Lastly, a confusion matrix is printed to visualize the model's classification performance and to understand how well the model distinguishes between different mental health issues. This end-to-end pipeline is crucial for applying the Support Vector Machine (SVM) algorithm to textual data, allowing for effective classification of mental health issues based on Reddit posts.

8.2.6 Random Forest for Classification

Random Forest Classifier Implementation

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,
    classification_report, confusion_matrix

# Load the preprocessed dataset
dataset = pd.read_csv('preprocessed_mental_health.csv')

# Check if 'cleaned_text' and 'mental_health_issue' columns
    exist
if 'cleaned_text' not in dataset.columns or '
    mental_health_issue' not in dataset.columns:
    raise ValueError("The dataset must have 'cleaned_text'
        and 'mental_health_issue' columns.")

# Remove rows with missing values in 'cleaned_text' column
dataset.dropna(subset=['cleaned_text'], inplace=True)

# Initialize the CountVectorizer and fit/transform the
    cleaned text
RFvectorizer = CountVectorizer()
X = RFvectorizer.fit_transform(dataset['cleaned_text'])

# Prepare the target variable
y = dataset['mental_health_issue']

# Split the dataset into Training and Test Sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

# Initialize the Random Forest Classifier with added
    parameters
RFmodel = RandomForestClassifier(
    n_estimators=3000,          # Number of trees
    max_depth=None,             # Maximum depth of each tree
    min_samples_split=20,       # Minimum number of samples
        to split a node
```

Random Forest Classifier Implementation

```
    min_samples_leaf=1,         # Minimum number of samples
       in a leaf node
    max_features='sqrt',        # Number of features to
       consider at each split
    bootstrap=False,             # Whether to use
       bootstrapping

    random_state=42             # For reproducibility
)

# Train the model
RFmodel.fit(X_train, y_train)

# Make predictions on the test set
y_pred = RFmodel.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Print classification report
print("Classification Report:\n", classification_report(
   y_test, y_pred))

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred
   ))
```

The provided code demonstrates how to apply the Random Forest Classifier for mental health classification using a preprocessed dataset. First, the dataset is loaded, and a check is performed to ensure that it contains the necessary columns, specifically `cleaned_text` and `mental_health_issue`. If these columns are missing, an error is raised. The dataset is then cleaned by removing rows that have missing values in the `cleaned_text` column using the `dropna()` function. Next, the `CountVectorizer` is initialized and applied to the `cleaned_text` column to convert the text data into a numerical format suitable for machine learning. This is accomplished by transforming the text into a document-term matrix where each row represents a document (post) and each column represents a unique term (word). The target variable, `mental_health_issue`, is also extracted from the dataset. The dataset is then split into training and testing sets using `train_test_split()`, where 80% of the data is used for training, and 20% is reserved for testing. The `random_state` is set to ensure reproducibility of the results. A Random Forest Classifier model is initialized with several hyperparameters. Specifically, `n_estimators=3000` defines the number of decision trees in the forest. The `max_depth=None` means the trees are allowed to grow until all leaves are pure or contain fewer than the minimum samples required to split a node. The `min_samples_split=20` and `min_samples_leaf=1` specify the minimum number of samples required to split an internal node and the minimum number of samples required to be at a leaf node, respectively. The `max_features='sqrt'` sets the maximum number of features to consider for the best split at each node to be the square root of the total number of features. `bootstrap=False` disables bootstrapping, meaning the entire dataset is used to build each tree. After initialization, the model is trained using the `fit()` method on the training data. After training, predictions are made on the test set using the `predict()` method. The model's performance is evaluated by calculating the accuracy score, which is printed as a percentage. Additionally, a detailed classification report is generated, which includes metrics such as precision, recall, and F1-score for each class. Lastly, a confusion matrix is printed to visualize the model's classification performance and to understand how well the model distinguishes between different mental health issues. This end-to-end pipeline is essential for applying the Random Forest algorithm to textual data, allowing for effective classification of mental health issues based on Reddit posts.

8.2.7 XGBoost for Classification

XGBoost Classifier Implementation

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,
   classification_report, confusion_matrix
import xgboost as xgb

# Load the dataset
data = pd.read_csv('preprocessed_mental_health.csv')

# Separate features and target
X = data['text']
y = data['mental_health_issue']

# Encode target labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)
```

XGBoost Classifier Implementation

```
# Convert text data to numerical data using TF-IDF
   Vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=5000)
X_train = tfidf.fit_transform(X_train)
X_test = tfidf.transform(X_test)

# Define the XGBoost classifier
xgb_clf = xgb.XGBClassifier(objective='multi:softmax',
   num_class=5, eval_metric='mlogloss', use_label_encoder=
   False)

# Train the model
xgb_clf.fit(X_train, y_train)

# Predict on the test set
y_pred = xgb_clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=
   label_encoder.classes_)

print(f"Accuracy: {accuracy * 100:.2f}%")
print("Classification Report:\n", report)

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred
   ))
```

The provided code demonstrates how to apply XGBoost for mental health classification using a preprocessed dataset. First, the dataset is loaded, and the target variable (`mental_health_issue`) is separated from the feature variable (`text`). The target variable is then encoded using `LabelEncoder()` to convert the categorical labels into numerical values, which are required for machine learning algorithms. Next, the dataset is split into training and testing sets using `train_test_split()`, where 80% of the data is used for training, and 20% is reserved for testing. The `random_state` is set to ensure reproducibility of the results. The text data is then transformed into numerical features using the `TfidfVectorizer()`. This vectorizer converts the raw text into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features, which are more suitable for training a machine learning model. The `max_features=5000` parameter limits the number of features to the 5000 most frequent terms in the dataset. The XGBoost classifier is then initialized with several hyperparameters.

The `objective='multi:softmax'` indicates that the task is a multi-class classification problem. The `num_class=5` parameter specifies the number of classes for the classification task, and `eval_metric='mlogloss'` is set to use the log loss metric for evaluation. `use_label_encoder=False` disables the automatic label encoding of target labels, as we have already manually encoded the labels. After initialization, the model is trained on the training data using the `fit()` method. After training, predictions are made on the test set using the `predict()` method. The model's performance is evaluated by calculating the accuracy score, which is printed as a percentage. Additionally, a detailed classification report is generated, which includes metrics such as precision, recall, and F1-score for each class. Lastly, a confusion matrix is printed to visualize the model's classification performance and to understand how well the model distinguishes between different mental health issues. This pipeline demonstrates the use of XGBoost in combination with TF-IDF vectorization for text classification, enabling effective classification of mental health issues based on Reddit posts.

8.2.8 Long Short Term Memory based Classification

LSTM Model Implementation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix,
   classification_report
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
   pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense,
   Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

# Load the dataset
data = pd.read_csv('preprocessed_mental_health.csv')

# Separate features and target
X = data['text']
y = data['mental_health_issue']
```

LSTM Model Implementation

```
# Encode target labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
y = to_categorical(y)  # Convert labels to one-hot encoded
    format for multi-class classification

# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

# Tokenize and pad the text sequences
vocab_size = 10000  # Set a vocabulary size
max_length = 100    # Set a max length for padding

tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>
    ")
tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

X_train_padded = pad_sequences(X_train_seq, maxlen=
    max_length, padding='post', truncating='post')
X_test_padded = pad_sequences(X_test_seq, maxlen=max_length,
    padding='post', truncating='post')

# Build the LSTM model
model = Sequential([
    Embedding(vocab_size, 128, input_length=max_length),
    LSTM(128, return_sequences=True),
    Dropout(0.2),
    LSTM(64),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dense(y.shape[1], activation='softmax')  # Output layer
        with softmax for multi-class classification
])

model.compile(optimizer='adam', loss='
    categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train_padded, y_train, epochs=10,
    batch_size=32, validation_data=(X_test_padded, y_test))
```

LSTM Model Implementation

```
# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training␣
    Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation␣
    Accuracy')
plt.title('Model␣Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training␣Loss')
plt.plot(history.history['val_loss'], label='Validation␣Loss
    ')
plt.title('Model␣Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(X_test_padded,
    y_test)
print(f"Test␣Accuracy:␣{test_accuracy␣*␣100:.2f}%")

# Generate predictions and convert back from one-hot
    encoding
y_pred = model.predict(X_test_padded)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)

# Output classification report
print("Classification␣Report:\n", classification_report(
    y_test_classes, y_pred_classes, target_names=
    label_encoder.classes_))

# Plot Confusion Matrix
cm = confusion_matrix(y_test_classes, y_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=label_encoder.classes_, yticklabels=
    label_encoder.classes_)
```

LSTM Model Implementation

```
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

This code demonstrates how to apply a Long Short-Term Memory (LSTM) neural network for text classification, specifically for predicting mental health issues based on preprocessed Reddit data. The dataset is loaded and the target variable (`mental_health_issue`) is encoded into a one-hot format for multi-class classification. The dataset is split into training and testing sets. Text data is tokenized and padded to ensure uniform input lengths for the LSTM model. The model consists of embedding layers, LSTM layers with dropout for regularization, and dense layers for classification. The model is trained on the training data, and performance is monitored using validation data. The training and validation accuracy and loss are plotted. The model is evaluated on the test data, and performance metrics such as accuracy and classification report are displayed. A confusion matrix is plotted to visualize model performance.

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
465/465 ━━━━━━━━━━━━━━━━━━━━ 135s 280ms/step - accuracy: 0.6046 - loss: 1.1142 - val_accuracy: 0.6841 - val_loss: 0.7390
Epoch 2/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 131s 281ms/step - accuracy: 0.6681 - loss: 0.7686 - val_accuracy: 0.6634 - val_loss: 0.8092
Epoch 3/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 144s 286ms/step - accuracy: 0.6499 - loss: 0.8130 - val_accuracy: 0.6680 - val_loss: 0.7538
Epoch 4/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 139s 280ms/step - accuracy: 0.6368 - loss: 0.8657 - val_accuracy: 0.6605 - val_loss: 0.7481
Epoch 5/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 130s 280ms/step - accuracy: 0.6949 - loss: 0.6574 - val_accuracy: 0.6922 - val_loss: 0.6711
Epoch 6/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 141s 278ms/step - accuracy: 0.7132 - loss: 0.6056 - val_accuracy: 0.6933 - val_loss: 0.6697
Epoch 7/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 147s 289ms/step - accuracy: 0.7304 - loss: 0.5676 - val_accuracy: 0.7081 - val_loss: 0.6606
Epoch 8/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 131s 282ms/step - accuracy: 0.7483 - loss: 0.5433 - val_accuracy: 0.7516 - val_loss: 0.6319
Epoch 9/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 142s 282ms/step - accuracy: 0.8079 - loss: 0.4523 - val_accuracy: 0.7484 - val_loss: 0.6037
Epoch 10/10
465/465 ━━━━━━━━━━━━━━━━━━━━ 142s 282ms/step - accuracy: 0.8564 - loss: 0.3823 - val_accuracy: 0.8355 - val_loss: 0.5368
```

Figure 10: Output for LSTM Epochs

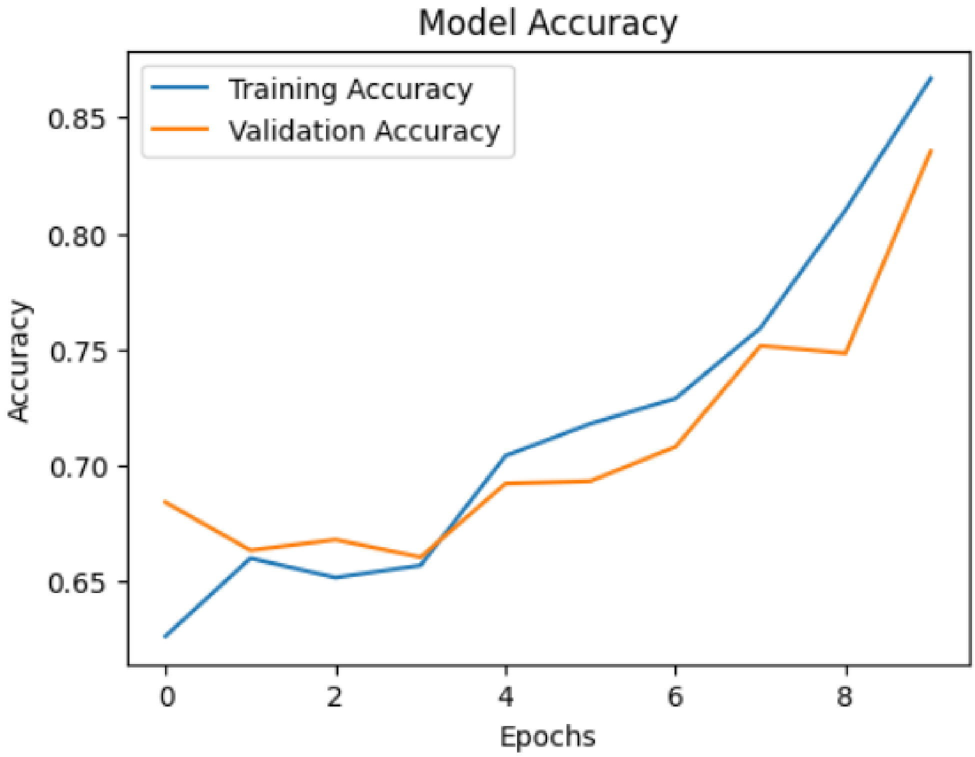

Figure 11: Accuracy v/s Epoch

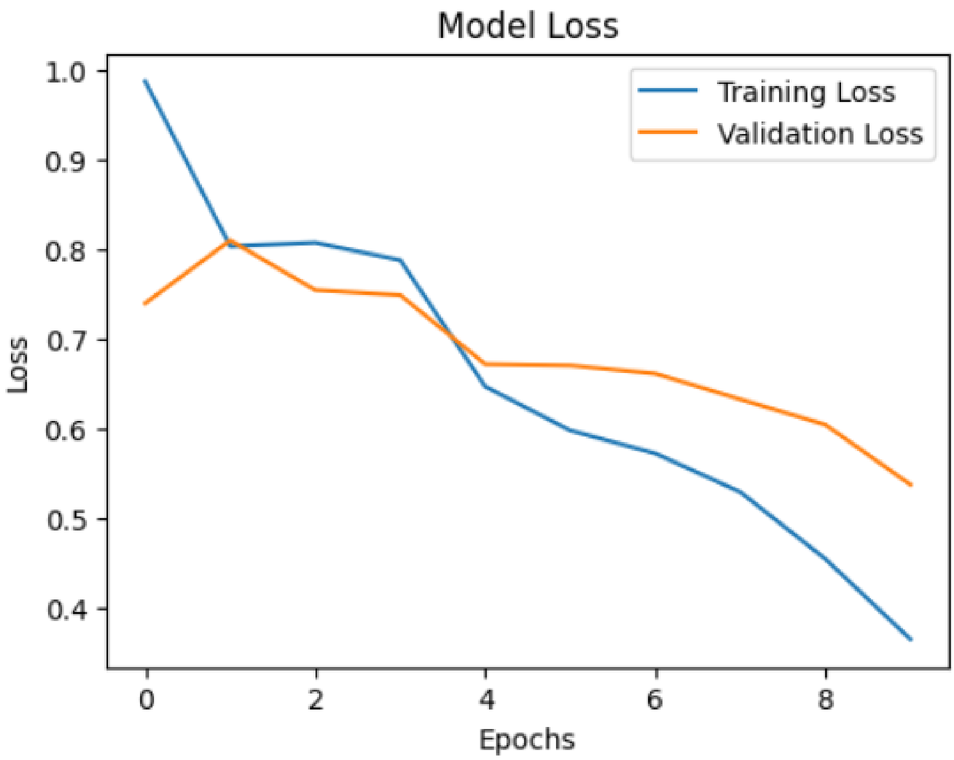

Figure 12: Loss v/s Epoch

9 Results and Analysis

The metrics used for evaluating the performance of the classification models include Precision, Recall, F1-Score, and Support, along with the Confusion Matrix. These metrics are crucial for assessing how well the models are able to differentiate between various classes, providing insight into their accuracy, ability to capture relevant instances, and the overall balance between precision and recall. By analyzing these metrics, a comprehensive understanding of the model's performance can be obtained, enabling informed decisions for further optimization and tuning.

9.1 Classification Metrics and Confusion Matrix

- **Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positives. It can be calculated as:

  $$\text{Precision} = \frac{TP}{TP + FP}$$

  where TP represents True Positives, and FP represents False Positives.

- **Recall:** Recall is the ratio of correctly predicted positive observations to all observations in the actual class:

  $$\text{Recall} = \frac{TP}{TP + FN}$$

  where TP is True Positives, and FN is False Negatives.

- **F1-Score:** F1-Score is the harmonic mean of Precision and Recall, calculated as:

  $$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Support:** Support refers to the number of actual occurrences of each class in the dataset:

  $$\text{Support} = \text{Number of samples in the true class}$$

- **Confusion Matrix:** A confusion matrix is used to evaluate the performance of a classification model. It is structured as follows:

  $$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

  where:

  - TP = True Positives
  - FP = False Positives
  - FN = False Negatives
  - TN = True Negatives

9.2 Results of Logistic Regression

Logistic Regression Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.83	0.77	0.80	379
Bipolar	0.74	0.55	0.63	384
Depression	0.76	0.76	0.76	373
Normal	0.92	0.99	0.95	2183
PTSD	0.87	0.77	0.82	394
Accuracy	87.66%			
Macro Avg	0.82	0.77	0.79	3713
Weighted Avg	0.87	0.88	0.87	3713

Confusion Matrix for Logistic Regression Model

	Anxiety	Bipolar	Depression	Normal	PTSD
Anxiety	291	18	27	24	19
Bipolar	7	213	29	125	10
Depression	26	31	283	18	15
Normal	3	10	4	2165	1
PTSD	24	16	29	22	303

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.95
Bipolar	0.92
Depression	0.96
Normal	0.99
PTSD	0.95

The Logistic Regression model performed well with an overall accuracy of 87.66%, indicating that the model correctly classified the majority of the instances. The classification report shows high precision, recall, and F1-scores for the 'Normal' class, which was expected due to its large number of instances. However, the 'Bipolar' and 'Anxiety' classes have lower recall and F1-scores, suggesting that the model struggles more with these classes. The confusion matrix highlights the misclassifications. For example, 'Anxiety' is often confused with 'Depression' and 'PTSD,' while the 'Normal' class is well-separated from the other classes. The large number of instances in the 'Normal' class could have contributed to the high accuracy but also to the imbalance in performance across other classes. The ROC curve AUC scores indicate that the model performs well in distinguishing between the classes. The 'Normal' class has the highest AUC (0.99), which is expected due to the large proportion of 'Normal' instances. Other classes, like 'Anxiety' and 'Depression,' also have high AUC scores (0.95 and 0.96, respectively), indicating that the model is capable of distinguishing them effectively.

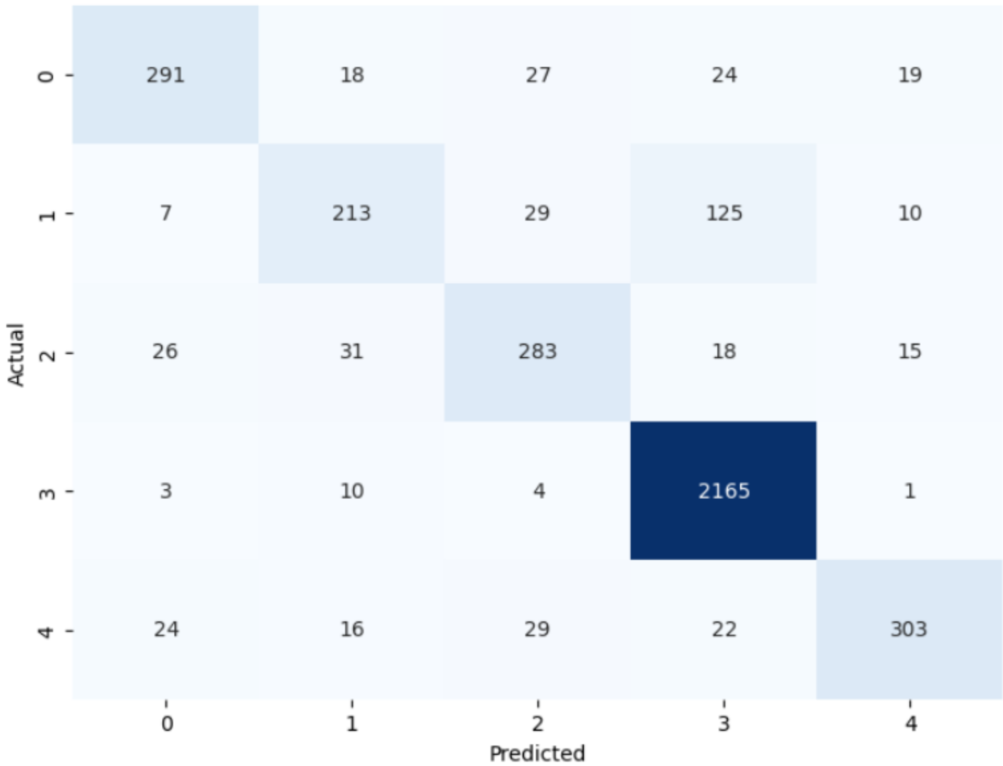

Figure 13: Confusion Matrix (Logistic Regression)

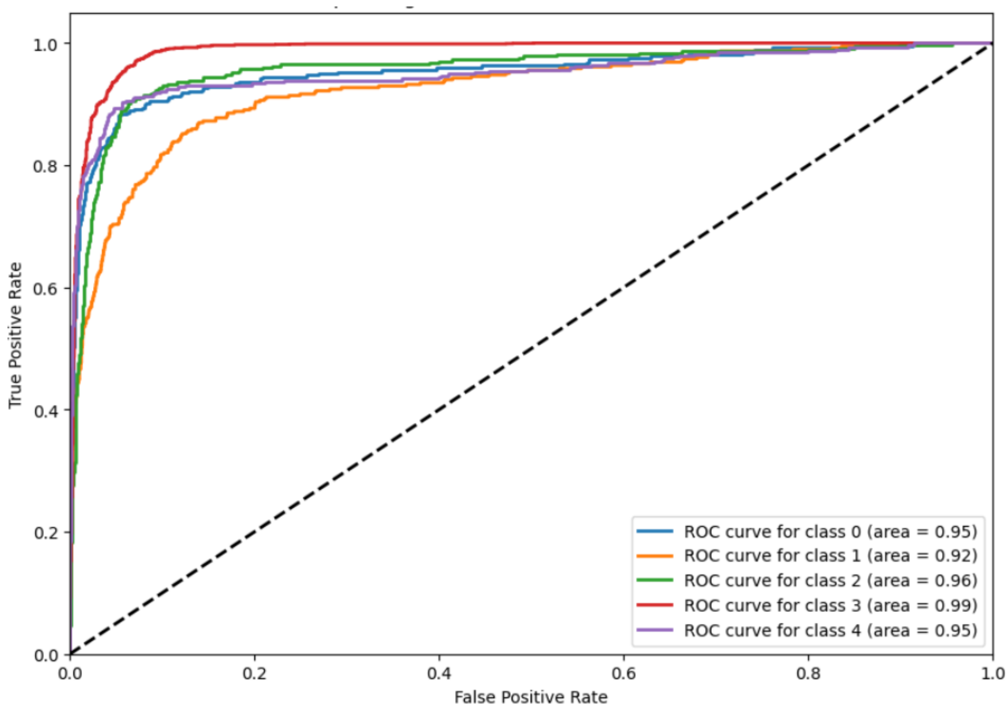

Figure 14: ROC AUC (Logistic Regression)

9.3 Results of Naive Bayes

Naive Bayes Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.70	0.73	0.72	379
Bipolar	0.83	0.45	0.58	384
Depression	0.59	0.87	0.70	373
Normal	0.96	0.92	0.94	2183
PTSD	0.71	0.83	0.76	394
Accuracy	83.63%			
Macro Avg	0.76	0.76	0.74	3713
Weighted Avg	0.85	0.84	0.84	3713

Confusion Matrix for Naive Bayes Model

	Anxiety	Bipolar	Depression	Normal	PTSD
Anxiety	278	4	63	3	31
Bipolar	30	171	62	86	35
Depression	26	6	323	0	18
Normal	38	21	68	2006	50
PTSD	24	5	34	4	327

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.92
Bipolar	0.89
Depression	0.94
Normal	0.99
PTSD	0.94

The Naive Bayes model achieved an overall accuracy of 83.63%, indicating a reasonable performance. The classification report shows strong results for the Normal class, with a precision of 0.96 and a recall of 0.92, resulting in an F1-score of 0.94. However, the Bipolar class exhibits much lower recall (0.45) and F1-score (0.58), suggesting that the model struggles to accurately identify Bipolar instances. Misclassifications are more frequent for Bipolar, often being confused with Depression and PTSD. The Anxiety class, although having decent precision (0.70), shows a lower recall (0.73), indicating that it also faces challenges in classification. The Confusion Matrix highlights that the model has difficulty distinguishing Bipolar and Depression from each other, with a large number of misclassifications across these classes. On the other hand, the Normal class is well-separated and correctly classified, with 2006 true positives, which likely contributes to the high overall accuracy. PTSD also shows relatively good classification results, with misclassifications being less frequent. The ROC AUC Curve Areas show that the model performs well for most classes, with the Normal class having the highest AUC score (0.99), followed by Depression and PTSD with AUCs of 0.94. While the model performs well for distinguishing between classes like Normal and Depression, the lower AUC for Bipolar (0.89) indicates that there may still be room for improvement in distinguishing this class from others.

Figure 15: Confusion Matrix (Naive Bayes)

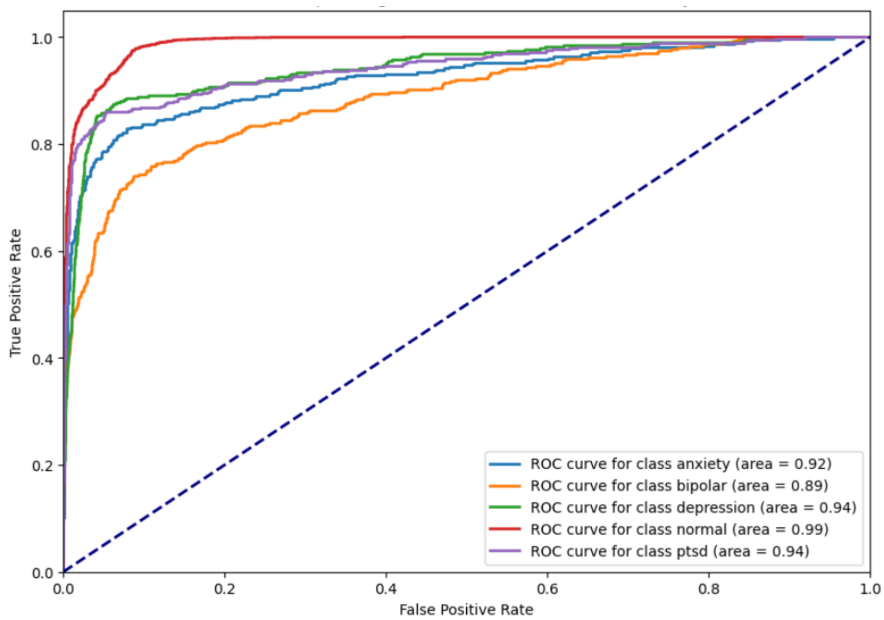

Figure 16: ROC AUC (Naive Bayes)

9.4 Results of Support Vector Machine

SVM Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.72	0.76	0.74	379
Bipolar	0.62	0.61	0.61	384
Depression	0.74	0.71	0.72	373
Normal	0.94	0.95	0.95	2183
PTSD	0.78	0.74	0.76	394
Accuracy	85.13%			
Macro Avg	0.76	0.75	0.76	3713
Weighted Avg	0.85	0.85	0.85	3713

Confusion Matrix for SVM Model

	Anxiety	Bipolar	Depression	Normal	PTSD
Anxiety	287	22	29	13	28
Bipolar	20	233	29	88	14
Depression	37	31	265	11	29
Normal	20	62	8	2084	9
PTSD	34	26	29	13	292

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.96
Bipolar	0.90
Depression	0.96
Normal	0.98
PTSD	0.96

The Support Vector Machine (SVM) model achieved an accuracy of 85.13%, demonstrating strong performance in classifying the various mental health conditions. The classification report indicates that the Normal class has the highest precision (0.94) and recall (0.95), resulting in an F1-score of 0.95, showing that the model is particularly good at identifying instances of normal behavior. However, the Bipolar class shows lower performance, with a precision of 0.62 and recall of 0.61, suggesting that the model struggles more with identifying bipolar disorder instances. The recall for the Anxiety class is relatively high (0.76), though precision is somewhat lower (0.72), indicating a balanced classification performance for this condition. The confusion matrix reveals that the model is generally accurate, with the Normal class being correctly classified most of the time (2084 true positives). However, some misclassifications occur for classes like Bipolar, Depression, and PTSD, with notable misclassifications of Bipolar as Depression and Normal. These misclassifications are likely affecting the overall recall for certain classes. The ROC AUC curve areas demonstrate that the model has excellent performance in distinguishing between most classes. The Normal class has the highest AUC of 0.98, followed by Anxiety, Depression, and PTSD with AUCs of 0.96. Bipolar, although still high at 0.90, lags behind the other classes, reflecting the model's challenge in distinguishing Bipolar from other conditions.

Figure 17: Confusion Matrix (SVM)

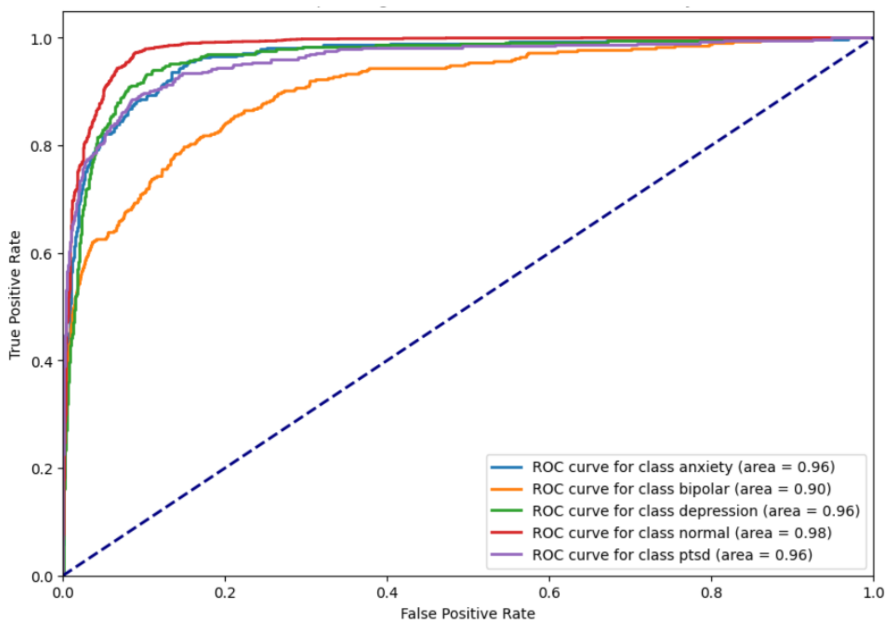

Figure 18: ROC AUC (SVM)

9.5 Results of Random Forest

Random Forest Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.81	0.70	0.75	379
Bipolar	0.93	0.47	0.62	384
Depression	0.72	0.77	0.74	373
Normal	0.88	1.00	0.93	2183
PTSD	0.92	0.74	0.82	394
Accuracy	86.00%			
Macro Avg	0.85	0.73	0.77	3713
Weighted Avg	0.86	0.86	0.85	3713

Confusion Matrix for Random Forest Model

	Anxiety	Bipolar	Depression	Normal	PTSD
Anxiety	264	3	34	68	10
Bipolar	12	180	46	141	5
Depression	26	3	286	48	10
Normal	3	6	1	2173	0
PTSD	19	2	30	53	290

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.96
Bipolar	0.89
Depression	0.97
Normal	0.97
PTSD	0.97

The Random Forest model achieved an accuracy of 86.00%, indicating a strong performance in classifying the mental health conditions. The classification report shows that the Normal class achieved the highest recall (1.00) and a precision of 0.88, resulting in a high F1-score of 0.93. This reflects the model's ability to correctly classify the majority of the Normal instances. The Bipolar class, however, shows a much lower recall (0.47), which indicates that the model has difficulty identifying instances of Bipolar disorder, as reflected by its F1-score of 0.62. The Anxiety and PTSD classes have relatively balanced performance, with moderate precision and recall values. The confusion matrix illustrates the distribution of misclassifications. The Normal class is correctly classified almost entirely (2173 true positives), while other classes like Bipolar and PTSD exhibit significant misclassifications, particularly Bipolar, which is often misclassified as Depression and Normal. The ROC AUC scores for all classes are quite high, indicating that the model is effective at distinguishing between the classes. The Anxiety, Depression, Normal, and PTSD classes each have AUC values above 0.96, with the Normal and Depression classes achieving 0.97. Bipolar has the lowest AUC at 0.89, which corresponds to its lower classification performance. These AUC values suggest that the Random Forest model is proficient in distinguishing between most of the classes, though it faces challenges in identifying Bipolar disorder.

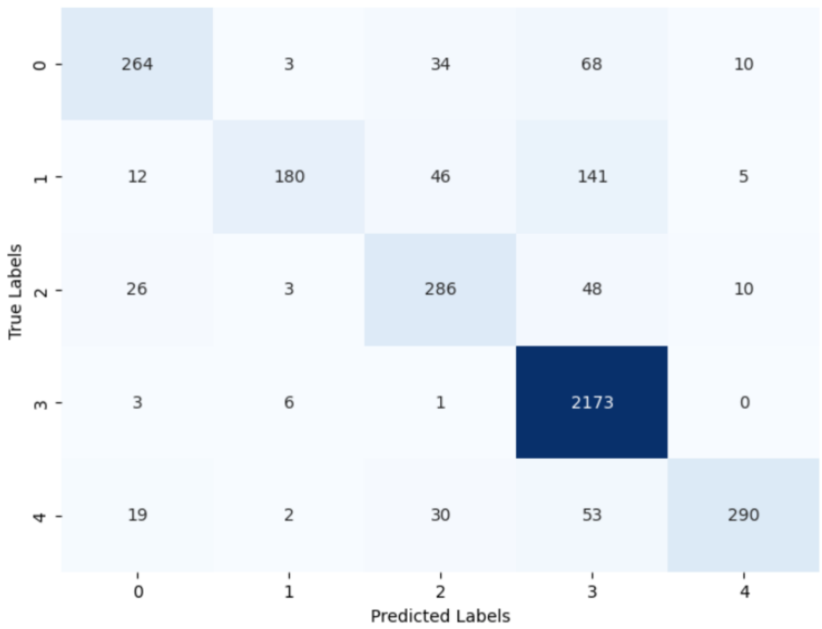

Figure 19: Confusion Matrix (Random Forest)

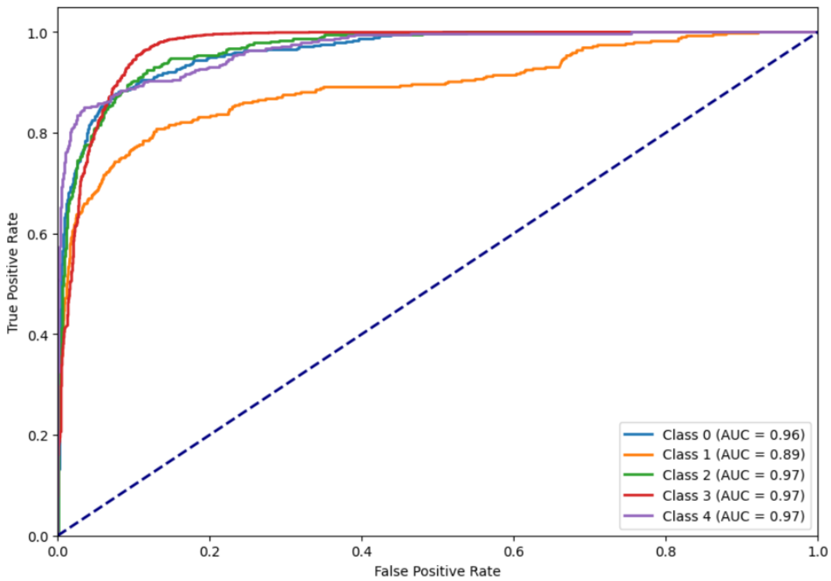

Figure 20: ROC AUC (Random Forest)

9.6 Results of XGBoost

XGBoost Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.81	0.74	0.77	403
Bipolar	0.77	0.62	0.69	397
Depression	0.72	0.81	0.76	387
Normal	0.93	0.98	0.95	2137
PTSD	0.86	0.75	0.80	396
Accuracy	87.39%			
Macro Avg	0.82	0.78	0.80	3720
Weighted Avg	0.87	0.87	0.87	3720

Confusion Matrix for XGBoost Model

	Anxiety	Bipolar	Depression	Normal	PTSD
Anxiety	297	20	39	24	23
Bipolar	13	248	36	92	8
Depression	30	13	313	17	14
Normal	3	28	8	2096	2
PTSD	22	12	36	29	297

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.97
Bipolar	0.95
Depression	0.97
Normal	0.99
PTSD	0.97

The XGBoost model achieved an accuracy of 87.39%, demonstrating strong overall performance. The classification report indicates high precision and recall for the ’Normal’ class, which is likely due to its substantial representation in the dataset. The ’Anxiety’ and ’PTSD’ classes also showed reasonable results, with the ’Anxiety’ class achieving a precision of 0.81 and a recall of 0.74. However, the ’Bipolar’ class exhibited a lower recall (0.62), suggesting that the model struggles more with this class. The confusion matrix highlights a well-separated ’Normal’ class, while the ’Bipolar’ and ’PTSD’ classes experience more confusion with other categories. The ROC AUC scores further emphasize the model’s good discriminatory capability, with AUC values of 0.97 for ’Anxiety,’ ’Depression,’ and ’PTSD,’ and 0.99 for ’Normal.’ These scores indicate that the model is proficient at distinguishing between different mental health conditions in the dataset.

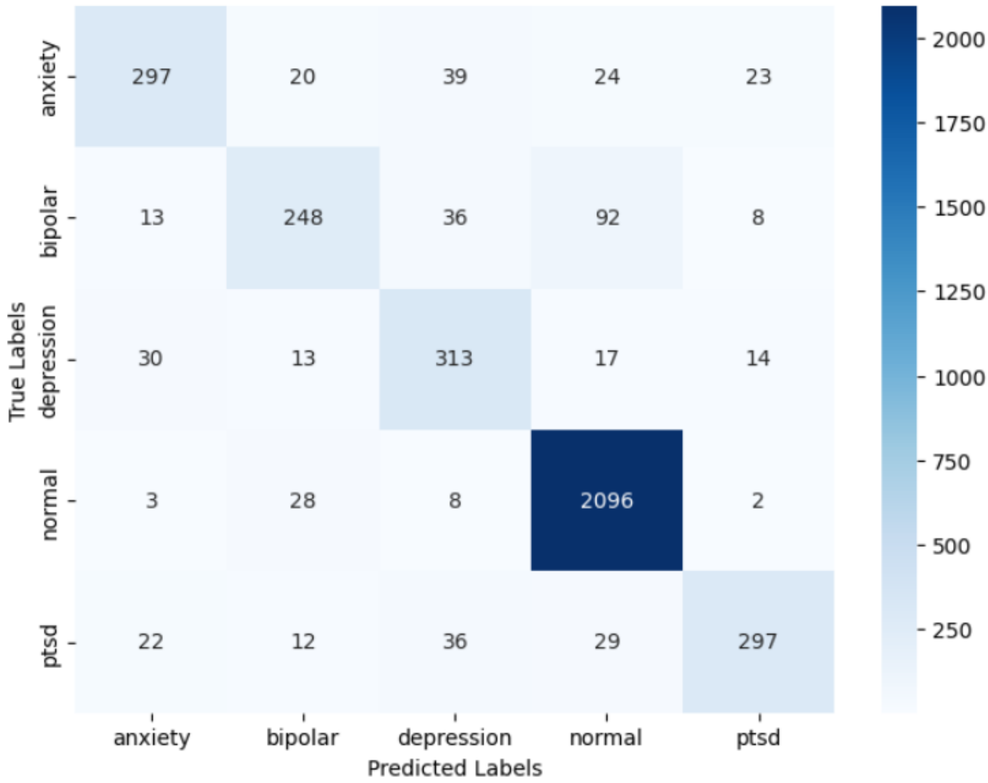

Figure 21: Confusion Matrix (XGBoost)

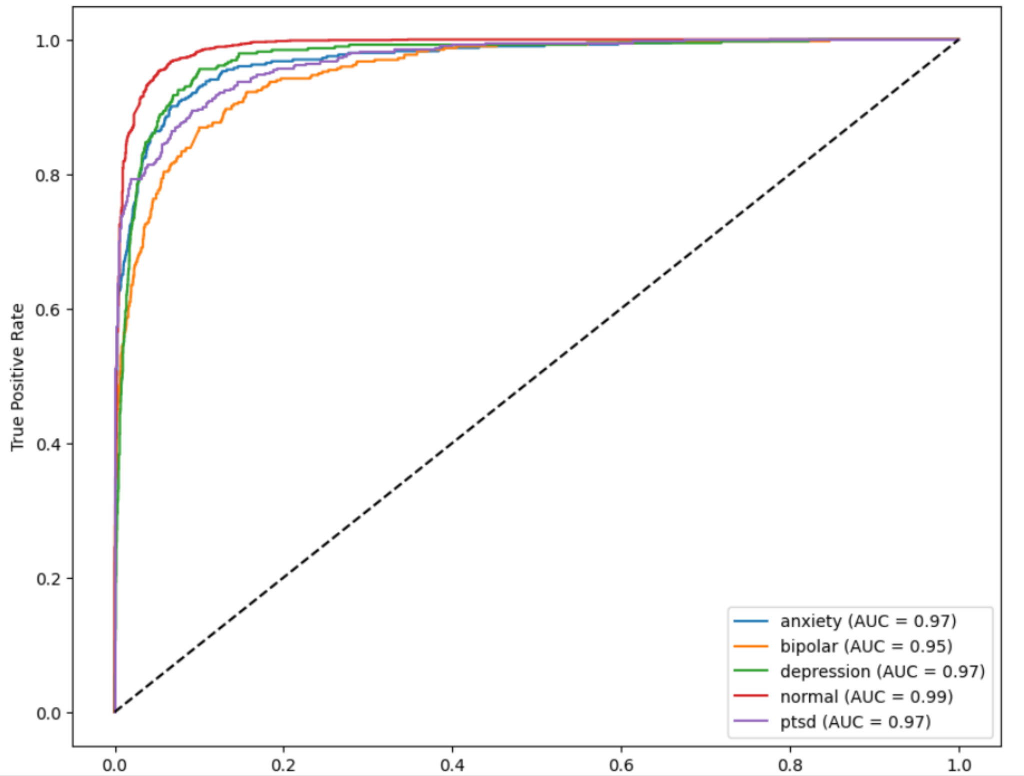

Figure 22: ROC AUC (XGBoost)

9.7 Results of LSTM

LSTM Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.81	0.68	0.74	403
Bipolar	0.60	0.59	0.60	397
Depression	0.56	0.77	0.65	387
Normal	0.96	0.96	0.96	2137
PTSD	0.76	0.61	0.68	396
Accuracy	83.55%			
Macro Avg	0.74	0.72	0.73	3720
Weighted Avg	0.84	0.84	0.84	3720

Confusion Matrix for LSTM Model

	Anxiety	Bipolar	Depression	Normal	PTSD
Anxiety	274	36	32	28	33
Bipolar	26	233	42	85	11
Depression	25	31	299	19	13
Normal	6	18	11	2043	9
PTSD	31	33	33	35	264

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.95
Bipolar	0.92
Depression	0.94
Normal	0.99
PTSD	0.95

The LSTM model achieved an accuracy of 83.55%, indicating strong performance, although it lags slightly behind other models such as XGBoost and SVM. The classification report shows high precision, recall, and F1-scores for the 'Normal' class, which dominates the dataset. The 'Anxiety' class has reasonable performance, with a precision of 0.81 and recall of 0.68, but the 'Bipolar' and 'PTSD' classes have lower precision and recall, especially for 'Bipolar' (0.60 in both precision and recall). The confusion matrix reflects this, with 'Bipolar' and 'PTSD' misclassified with other classes, while 'Normal' is well-separated. The ROC AUC scores highlight the model's ability to distinguish between classes effectively, with 'Normal' scoring the highest AUC of 0.99, followed by other classes such as 'Anxiety' and 'Depression,' both with AUCs above 0.90.

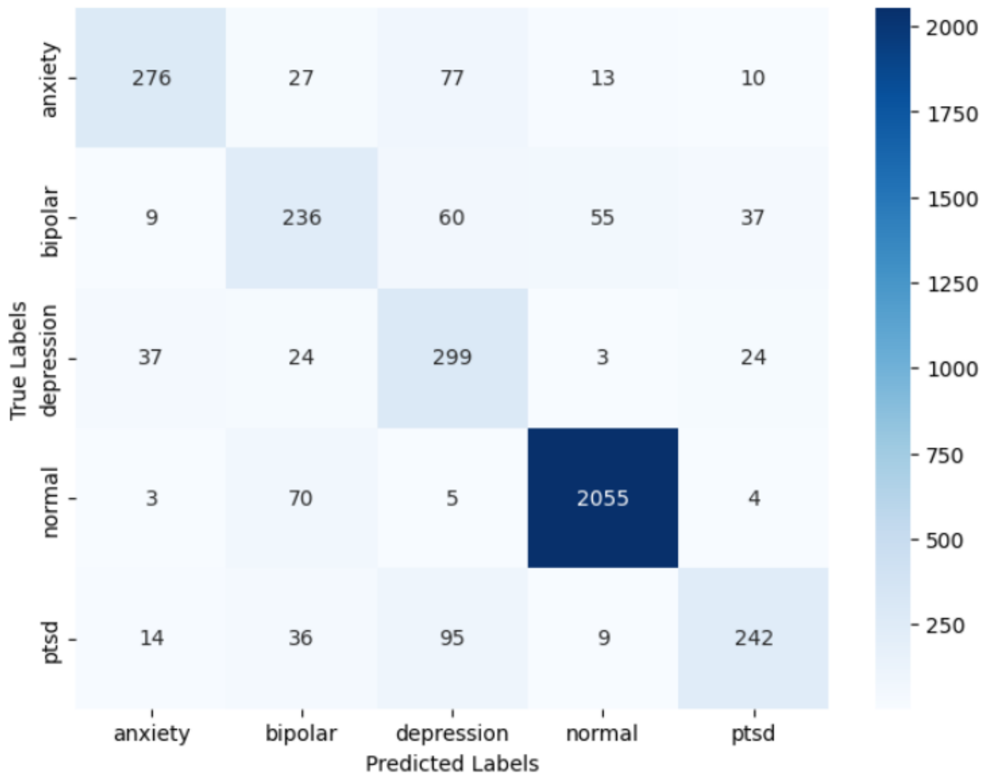

Figure 23: Confusion Matrix (LSTM)

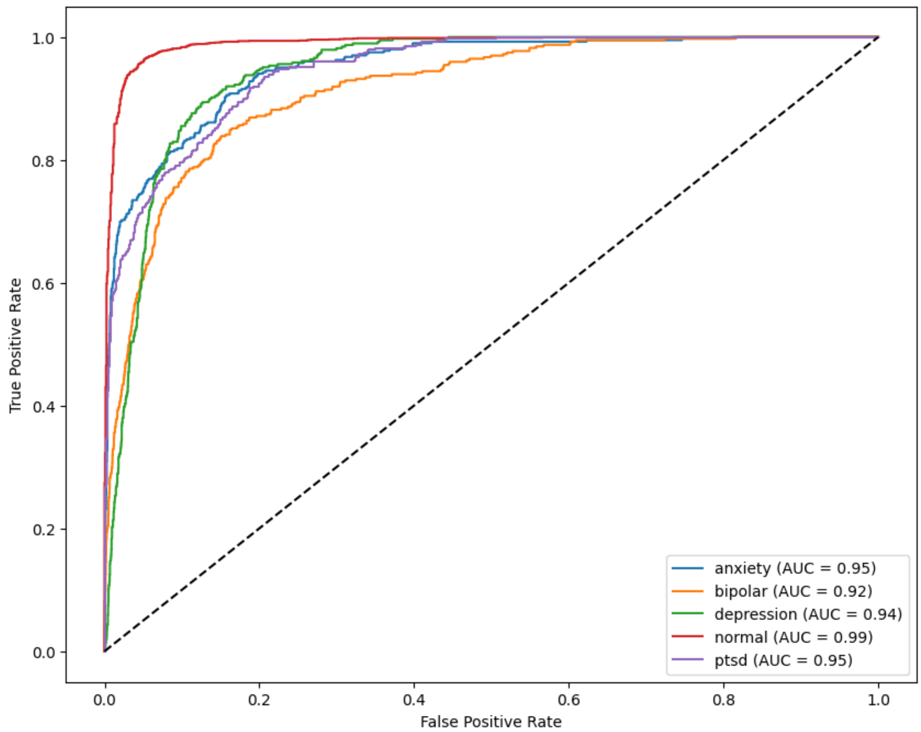

Figure 24: ROC AUC (LSTM)

Among the various machine learning algorithms tested, Logistic Regression emerged as the best performing model. With an accuracy of 87.66%, it outperformed the other models in terms of overall accuracy, precision, recall, and F1-score. The classification report for Logistic Regression shows strong performance across all classes, especially the 'Normal' class, which had high precision (0.92) and recall (0.99). Although the model faced challenges with the 'Bipolar' class, which had lower recall and F1-score, its overall ability to differentiate between other mental health conditions was impressive. The confusion matrix and ROC AUC scores further highlight the model's robustness, with an AUC of 0.99 for the 'Normal' class and strong values for other classes as well. This indicates that Logistic Regression not only achieves high accuracy but also performs well in distinguishing between different mental health categories, making it the most reliable choice for this classification task.

10 Conclusion

10.1 Project Benefits

The project on detecting mental health disorders through social media analysis offers a wide array of significant benefits, both immediate and long-term, across multiple dimensions. First and foremost, it addresses a critical issue in mental health care—early detection and intervention. Social media has become a ubiquitous platform where people express their thoughts, feelings, and emotional states, often unconsciously. By leveraging the vast amounts of data available on social media platforms, our project seeks to tap into this resource to identify early signs of mental health disorders such as anxiety, depression, and more severe conditions like bipolar disorder or schizophrenia. The ability to detect mental health issues through real-time social media data is a game-changer for public health systems, mental health practitioners, and even individuals who may not realize they are at risk. Early detection enables timely intervention, reducing the overall burden of mental health disorders on society by preventing escalation into more severe conditions that often lead to hospitalization, self-harm, or even suicide. In this sense, the project aligns with global health initiatives that emphasize early diagnosis and preventive care.

Moreover, this project holds significant potential for improving the accuracy and efficiency of mental health diagnostics. Traditional diagnostic methods are often time-consuming, subjective, and reliant on self-reporting, which can lead to underdiagnosis or misdiagnosis. By utilizing machine learning algorithms and natural language processing techniques, our project automates the process of sentiment and behavioral analysis on social media platforms, offering a more objective and data-driven approach. This automated system can process large volumes of data much faster than human professionals, providing insights that would be impossible to glean from manual analysis. The algorithms developed as part of this project can be easily scaled to analyze millions of social media posts, enabling a broader reach in monitoring public mental health trends. Additionally, the project offers practical benefits for mental health professionals, allowing them to focus on treatment and intervention rather than diagnosis. It provides a tool that can be integrated into telehealth systems, offering mental health screening at scale, which is particularly valuable in underserved or rural areas where access to mental health professionals is limited.

From a technological standpoint, the project offers a host of reusable components and methodologies. The machine learning models developed, the sentiment analysis tools, and the overall data pipeline are designed to be scalable and modular. These components can be adapted and extended to other domains beyond mental health, such as market sentiment analysis, public opinion monitoring, or even detecting harmful behavior like cyberbullying and harassment online. By advancing the state of the art in social media analytics, this project contributes to the growing field of AI-driven health care solutions. Furthermore, it provides a blueprint for future interdisciplinary work that integrates data science, psychology, and public health.

10.2 Future Scope for Improvements

While this project offers numerous immediate benefits, there is substantial room for future enhancements that can broaden its applicability, accuracy, and effectiveness. Currently, the project focuses on analyzing Reddit data using PRAW, which is limited to a specific social media platform and dataset. In the future, incorporating data from other platforms like Facebook, Instagram, Twitter, and even niche forums could provide a more comprehensive understanding of an individual's mental health status. Different platforms cater to different demographics and social behaviors, and expanding the dataset will allow for a more holistic analysis of mental health indicators across various user bases. Additionally, expanding the dataset to include multilingual posts or integrating language translation capabilities could make the system applicable to a global audience, helping to identify mental health issues in non-English speaking populations.

Moreover, future improvements could focus on integrating real-time data analysis capabilities. Currently, our project is based on batch processing of historical data. However, in future iterations, the system could be developed to perform real-time monitoring, offering immediate feedback and potentially alerting health professionals or loved ones when someone shows signs of mental distress. This real-time capability would be invaluable in emergency situations, allowing for immediate intervention. Developing a mobile application or a web-based interface where users can voluntarily connect their social media accounts to monitor their mental health status could also increase user engagement and provide individuals with direct feedback on their well-being.

Another significant future enhancement could involve incorporating ethical considerations and improving user privacy. As mental health is a sensitive subject, ensuring that the system is designed with robust privacy protections is critical. Future work could focus on using differential privacy or other anonymization techniques to ensure that user data remains confidential while still allowing for effective analysis. Moreover, collaborating with psychologists, ethicists, and legal experts could help refine the system to ensure it adheres to ethical guidelines and avoids potential harm, such as misdiagnosis or privacy violations.

Lastly, the future scope of this project could include expanding its use in clinical settings. While the current system is primarily designed as a research tool, future iterations could be developed in collaboration with mental health professionals to ensure that it meets clinical standards.

11 References

[1] Hatoon S AlSagri and Mourad Ykhlef. Machine learning-based approach for depression detection in twitter using content and activity features. *IEICE Transactions on Information and Systems*, 103(8):1825–1832, 2020.

[2] Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. Predicting depression via social media. *Proceedings of the International AAAI Conference on Web and Social Media*, 2013.

[3] Sharath Chandra Guntuku, David Bryce Yaden, Margaret L. Kern, Lyle H. Ungar, and Johannes C. Eichstaedt. Detecting depression and mental illness on social media: an integrative review. *Current Opinion in Behavioral Sciences*, 18:43–49, 2017.

[4] Priya Mathur, Amit Kumar Gupta, and Abhishek Dadhich. Mental health classification on social-media: Systematic review. *Proceedings of the 4th International Conference on Information Management & Machine Intelligence*, 2022.

[5] Moin Nadeem. Identifying depression on twitter. *arXiv preprint arXiv:1607.07384*, 2016.

[6] Ramin Safa, S. A. Edalatpanah, and Ali Sorourkhah. Predicting mental health using social media: A roadmap for future development, 2023.

[7] Konda Vaishnavi, U Nikhitha Kamath, B Ashwath Rao, and N V Subba Reddy. Predicting mental health illness using machine learning algorithms. *Journal of Physics: Conference Series*, 2161(1):012021, jan 2022.

APPENDIX A - Prototype

Import Libraries and Setup

```
import streamlit as st
import joblib
import pandas as pd
import praw
from PIL import Image
from deep_translator import GoogleTranslator
import requests
from io import BytesIO
from collections import Counter
import google.generativeai as genai
import pytesseract

pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'
```

This section imports necessary libraries such as streamlit for the web interface, joblib for loading pre-trained models, pandas for data manipulation, praw for interacting with Reddit, and pytesseract for Optical Character Recognition (OCR). It also sets the Tesseract executable path for OCR.

Reddit API Setup and Functions

```
# Initialize Reddit API
reddit = praw.Reddit(client_id='<REDDIT_CLIENT_ID>',
                     client_secret='JREDDIT_CLIENT_SECRET',
                     user_agent='Mental␣Health')

# Function to fetch text-based posts from Reddit
def fetch_user_text_posts(username):
    try:
        user = reddit.redditor(username)
        posts = [post.title + "␣" + post.selftext for post
           in user.submissions.new(limit=20)]
        return posts
    except Exception as e:
        st.write(f"Error␣fetching␣text␣posts:␣{e}")
        return []
```

This part sets up the Reddit API using praw to interact with Reddit. It also defines a function fetch_user_text_posts that fetches the latest 20 posts from a given Reddit username, returning their title and self-text.

Image OCR Functionality

```
# Function to fetch image-based posts from Reddit and
   perform OCR
def fetch_user_images_and_extract_text(username):
    try:
        user = reddit.redditor(username)
        images = [post.url for post in user.submissions.new(
           limit=20) if post.url.endswith((’.jpg’, ’.jpeg’,
           ’.png’, ’.webp’, ’.bmp’, ’.tiff’))]

        extracted_texts = []
        for image_url in images:
            try:
                response = requests.get(image_url)
                image = Image.open(BytesIO(response.content)
                   )
                st.image(image, caption="Fetched␣Image",
                   use_column_width=True)

                # Extract text from image
                extracted_text = extract_text_from_image(
                   image)
                extracted_text = "\n".join(extracted_text)

                # Translate to English if needed
                if extracted_text.strip():
                    translated_text = GoogleTranslator(
                       source=’auto’, target=’en’).translate
                       (extracted_text)
                    extracted_texts.append(translated_text)
                    st.write("Extracted␣and␣Translated␣Text␣
                       from␣Image:")
                    st.text(translated_text)
            except Exception as e:
                st.write(f"Error␣processing␣image␣{image_url
                   }:␣{e}")

        return extracted_texts
    except Exception as e:
        st.write(f"Error␣fetching␣images:␣{e}")
        return []

        # Function to extract text from image using
           Tesseract
        def extract_text_from_image(image):
            extracted_text = pytesseract.image_to_string(
               image)
            return extracted_text.splitlines()
```

The above code fetches image posts from a Reddit user’s submissions and extracts text from those images using OCR (Tesseract). It also translates the extracted text into English if necessary.

Text Classification and Wellbeing Insight

```
# Configure the Gemini API for wellbeing mapping
genai.configure(api_key="<GEMINI_API_KEY>")
generation_config = {
    "temperature": 1,
    "top_p": 0.95,
    "top_k": 40,
    "max_output_tokens": 8192,
    "response_mime_type": "text/plain",
}
gemini_model = genai.GenerativeModel(
    model_name="gemini-1.5-flash",
    generation_config=generation_config,
)

# Function to classify text and display result
def classify_text(text):
    input_vectorized = vectorizer.transform([text])
    prediction_proba = model.predict_proba(input_vectorized)

    issue_labels = model.classes_
    proba_df = pd.DataFrame(prediction_proba, columns=
       issue_labels).T
    proba_df.columns = ['Probability']

    top_issue = proba_df['Probability'].idxmax()
    top_probability = proba_df['Probability'].max()

    st.write(f"The␣most␣likely␣mental␣health␣concern␣is:␣{
       top_issue}␣with␣a␣probability␣of␣{top_probability
       :.2%}")

    # Call the Gemini model to get well-being insights
    get_wellbeing_insight(text, top_issue)
```

Text Classification and Wellbeing Insight

```
# Function to get well-being insights from Gemini model
def get_wellbeing_insight(text, top_issue):
    try:
        chat_session = gemini_model.start_chat(history=[])
        prompt = f"Analyze␣the␣following␣text␣for␣mental␣
           wellbeing␣insights␣related␣to␣{top_issue}:␣{text
           }.␣Based␣on␣this,␣provide␣practical␣advice␣or␣
           actions␣the␣user␣can␣take␣to␣reduce␣or␣improve␣{
           top_issue}.␣Be␣supportive␣and␣provide␣actionable␣
           suggestions."
        response = chat_session.send_message(prompt)

        st.write("###␣Wellbeing␣Insight:")
        st.write(response.text)
    except Exception as e:
        st.write(f"Error␣retrieving␣wellbeing␣insights:␣{e}"
           )
```

The classify_text function takes text input, vectorizes it using a pre-trained vectorizer, and predicts the most likely mental health issue. It also calls the Gemini model to provide wellbeing insights and advice.

Main Streamlit Application Logic

```
    # Define the Streamlit app
    def run_app():
        st.title("Mental␣Health␣Classifier␣App")

        # Option to choose functionality
        option = st.sidebar.selectbox(
            "Choose␣an␣option",
            ["Text␣Input", "Image␣Upload", "Reddit␣Username␣
               Analysis"]
        )
```

There are options to predict only text, predict only image or input a reddit username. The username would be used along with PRAW to extract the top 20 texts and 20 images from the recent posts. This extracted data would then be used further for mental health classification. Also Gemini AI would help in mapping this mental health issue with probable mental wellbeing.

Main Streamlit Application Logic

```
    # 1. Text Input
    if option == "Text Input":
        st.subheader("Enter Text to Classify Mental Health Issue
            ")
        input_text = st.text_area("Enter your text here:")

        if st.button("Classify Text"):
            if input_text.strip() == "":
                st.write("Please enter some text to classify.")
            else:
                # Translate if not in English
                translated_text = GoogleTranslator(source='auto'
                    , target='en').translate(input_text)
                st.write("Translated Text (to English):")
                st.write(translated_text)

                # Classify and display result
                classify_text(translated_text)

    # 2. Image Upload
    elif option == "Image Upload":
        st.subheader("Upload an Image to Extract and Classify
            Text")
        uploaded_image = st.file_uploader("Upload an Image",
            type=["jpg", "jpeg", "png", "webp", "bmp", "tiff"])

        if uploaded_image is not None:
            image = Image.open(uploaded_image)
            st.image(image, caption="Uploaded Image",
                use_column_width=True)

            # Extract text from image
            extracted_text = extract_text_from_image(image)
            extracted_text = "\n".join(extracted_text)

            st.subheader("Extracted Text")
            st.text(extracted_text)

            # Translate text to English if needed
            translated_text = GoogleTranslator(source='auto',
                target='en').translate(extracted_text)
            st.subheader("Translated Text (to English)")
            st.text(translated_text)
```

Main Streamlit Application Logic

```
    if st.button("Classify Extracted Text"):
        classify_text(translated_text)

    # 3. Reddit Username Analysis
    elif option == "Reddit Username Analysis":
        st.subheader("Enter Reddit Username for Analysis")
        username = st.text_input("Enter Reddit username:")

        if st.button("Analyze"):
            if username.strip() == "":
                st.write("Please enter a Reddit username.")
            else:
                # Fetch and display text posts
                text_posts = fetch_user_text_posts(username)
                if text_posts:
                    st.write("Recent Text Posts:")
                    st.write(text_posts[:3])  # Display a few posts for review

                # Fetch and display image-based posts with extracted text
                image_texts = fetch_user_images_and_extract_text(username)

        # Combine text from both text posts and image text
        all_text = text_posts + image_texts
        if all_text:
            predictions = []
            for text in all_text:
                # Vectorize and classify each post
                input_vectorized = vectorizer.transform([text])
                prediction = model.predict(input_vectorized)
                predictions.append(prediction[0])

        # Count the most common mental health issue
        issue_counts = Counter(predictions)
        top_issue, top_count = issue_counts.most_common(1)[0]
        top_percentage = (top_count / len(predictions)) * 100
```

Main Streamlit Application Logic

```
            st.write(f"The␣most␣frequently␣detected␣mental␣
               health␣concern␣is:␣{top_issue}␣appearing␣in␣{
               top_percentage:.2f}%␣of␣analyzed␣text.")
            issue_distribution = pd.DataFrame(issue_counts.items
               (), columns=['Mental␣Health␣Issue', 'Count'])
            st.write("Mental␣health␣issue␣distribution␣across␣
               posts:")
            st.write(issue_distribution)

            # Call the Gemini model to get well-being insights
            get_wellbeing_insight("␣".join(all_text), top_issue)
        else:
            st.write("No␣valid␣text␣found␣for␣analysis.")

# Run the app
if __name__ == '__main__':
    run_app()
```

This section defines the main structure of the Streamlit app, providing options for users to input text, upload images, or analyze Reddit usernames. Each option triggers different functionalities like text classification, image-based text extraction, and Reddit post analysis.

The main logic of the Streamlit application focuses on text and image classification for mental health issues, alongside an analysis of Reddit posts. The application offers three primary features: text input, image upload, and Reddit username analysis. For text input, users can enter text which is automatically translated into English if needed. The translated text is then passed to a classifier, which determines the most probable mental health concern based on the content. If the user uploads an image, the application extracts text from the image using OCR (Optical Character Recognition), translates it to English if necessary, and then classifies the extracted text. The Reddit username analysis feature allows users to input a Reddit username. The application fetches the most recent posts by the user, classifies both text-based and image-based posts, and then aggregates the results to identify the most frequent mental health issue discussed. If both text and image data are available, the application analyzes all content, classifies the posts, and provides insights into the most commonly detected issue. It also visualizes the distribution of detected issues and uses the Gemini AI model to provide well-being advice and actionable steps for users. The code integrates multiple components including the 'GoogleTranslator' for text translation, Tesseract for image text extraction, and a pre-trained logistic regression model to classify text data. Additionally, it uses the Gemini API to generate well-being insights based on the classified mental health concerns.

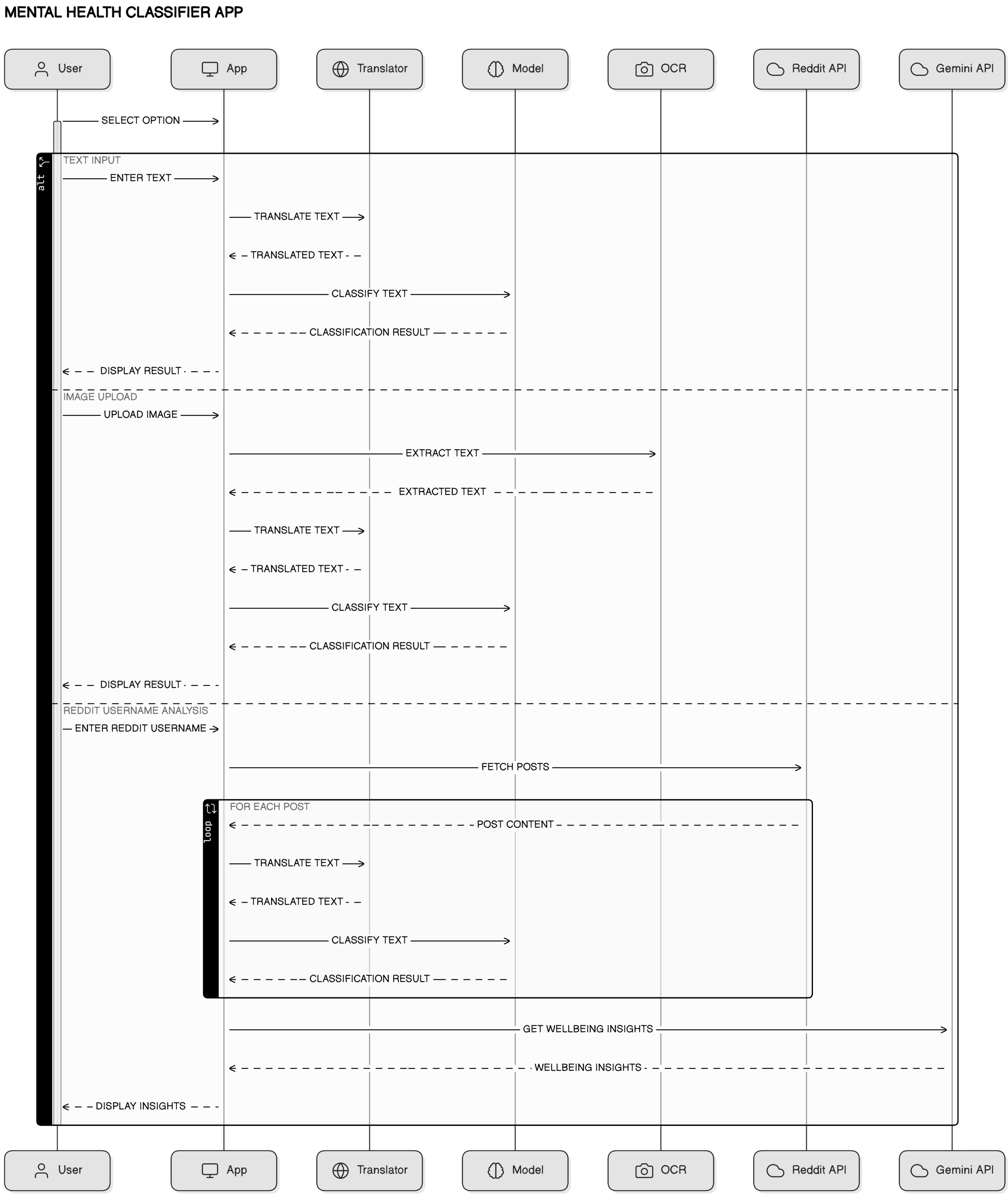

Figure 25: Application Sequence Diagram

Below are some screenshots from the web application.

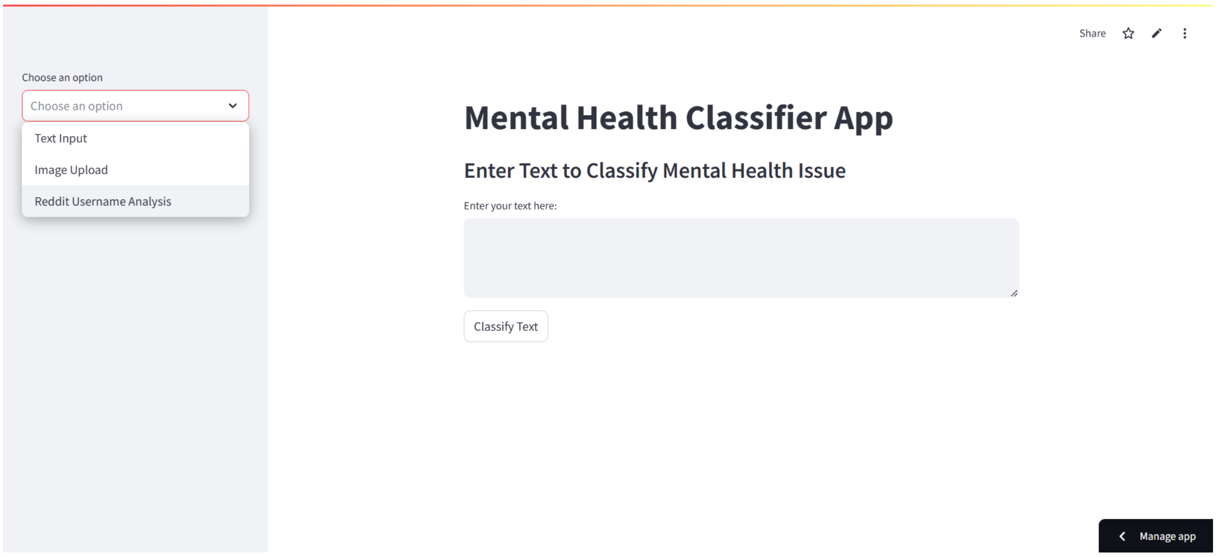

Figure 26: 01 Website with all options

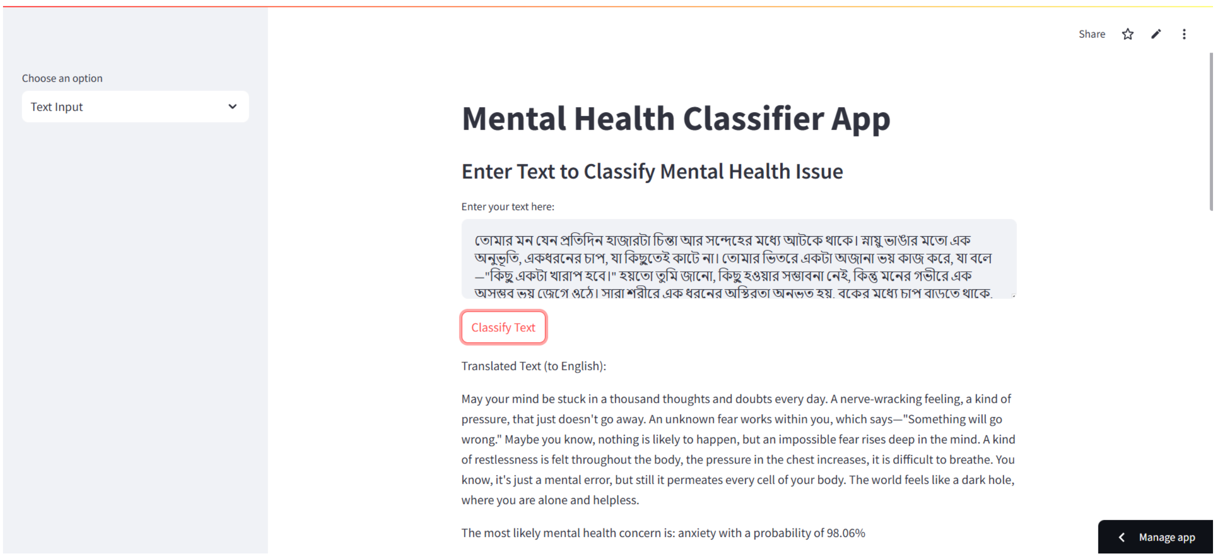

Figure 27: 02 Entering Text for classification

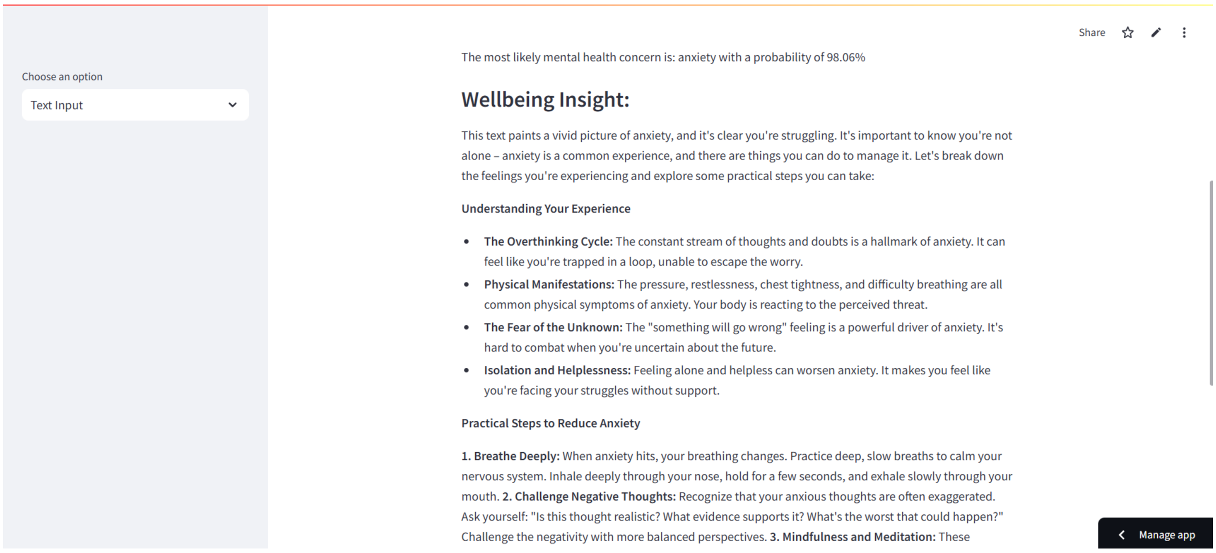

Figure 28: 03 Text Classification Result

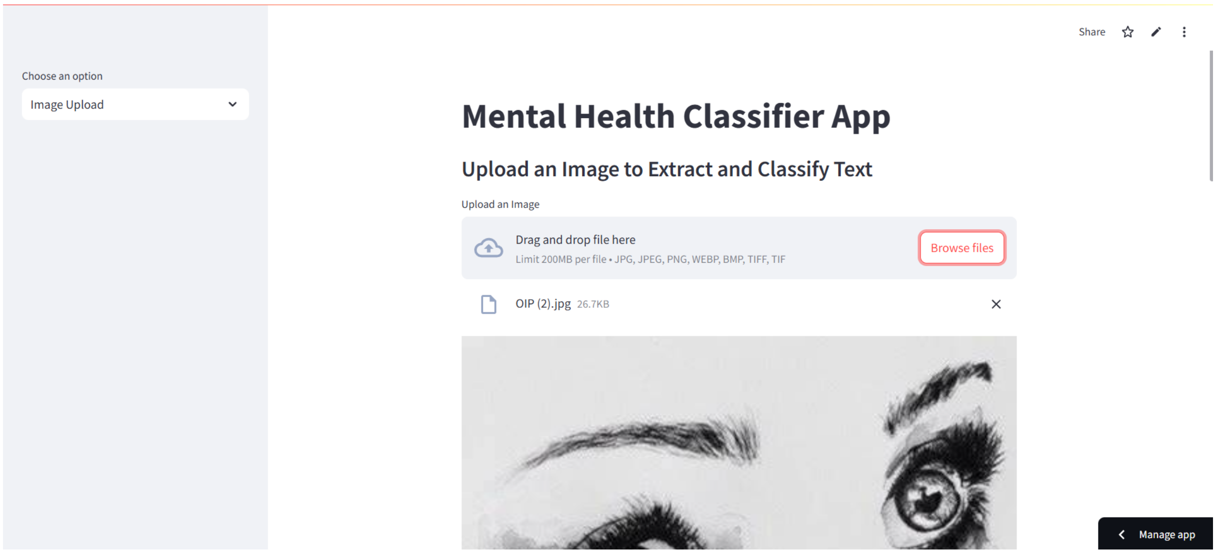

Figure 29: 04 Image Classification

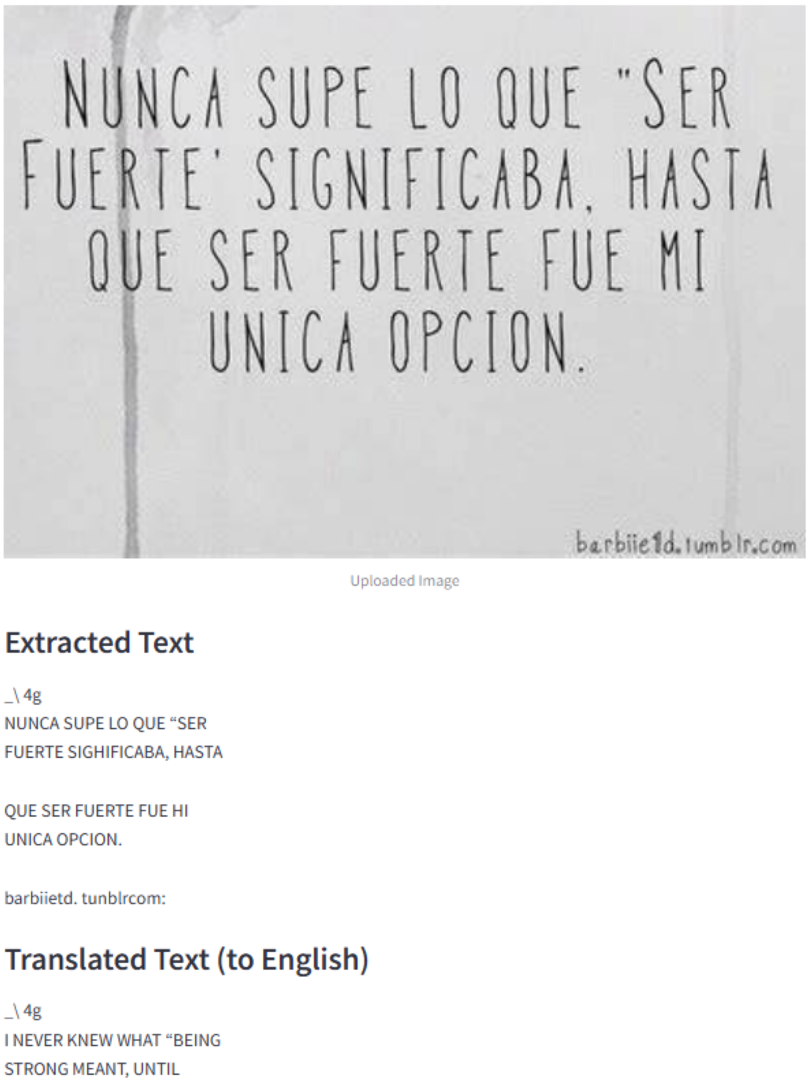

Figure 30: 05 Extracted Text from Image

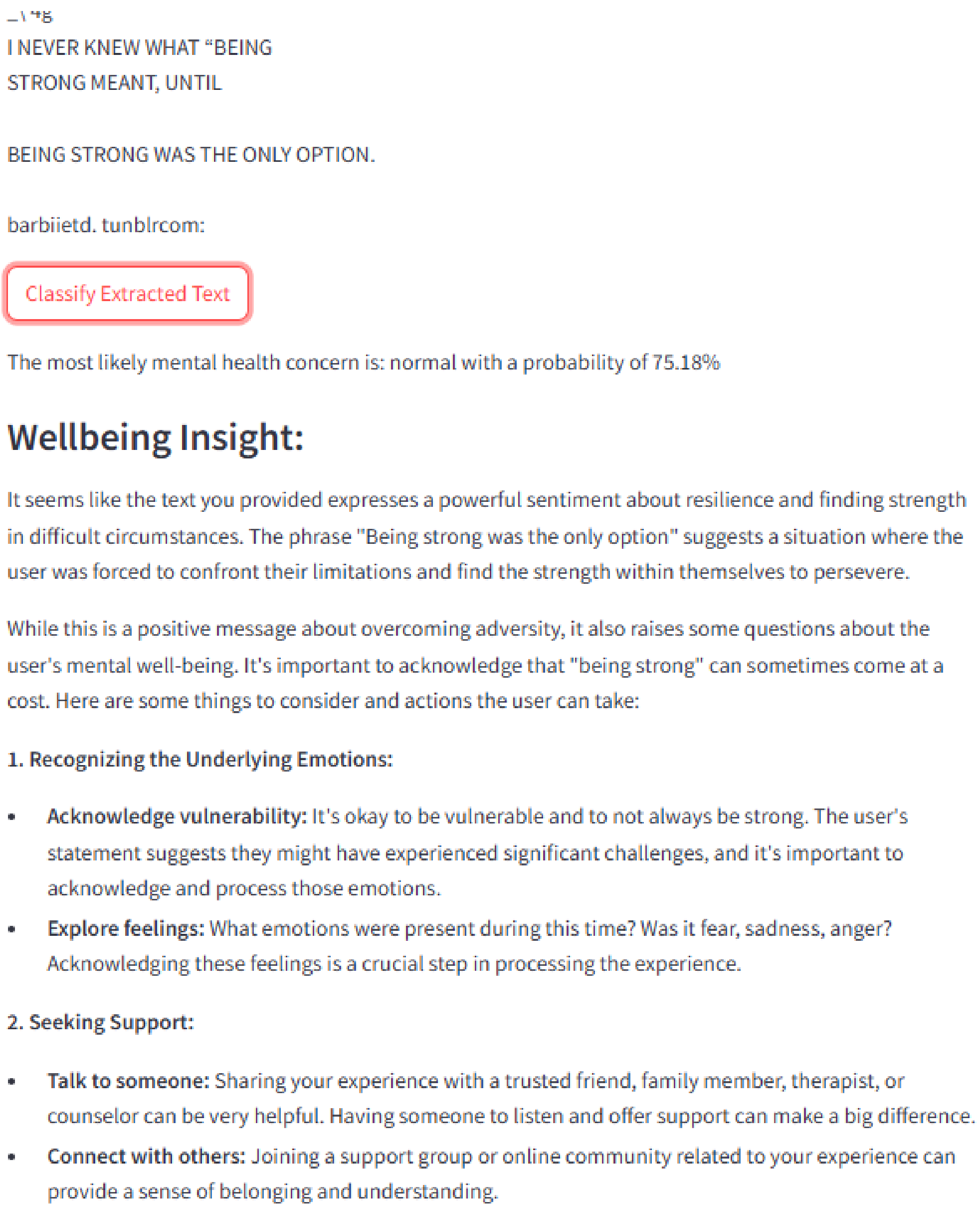

Figure 31: 06 Image Classification Result

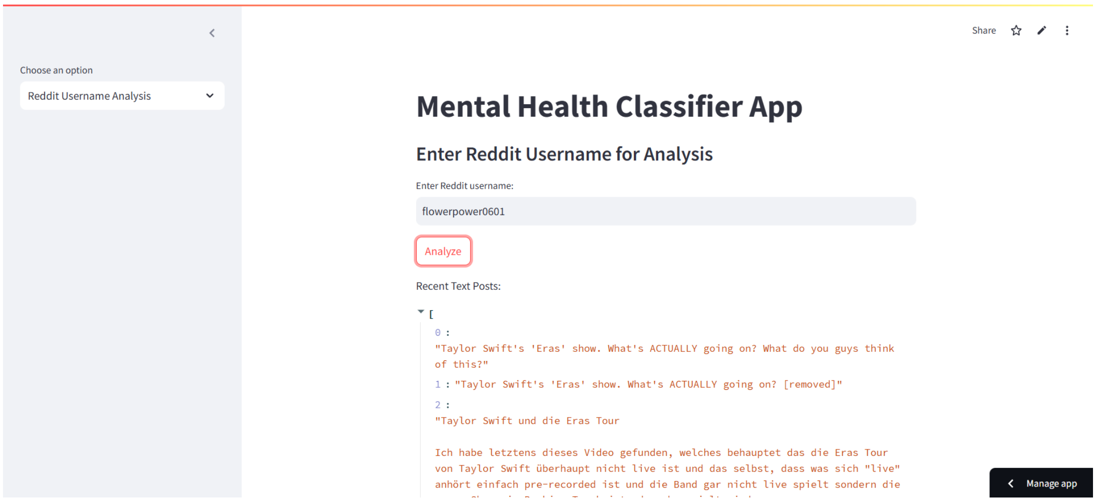

Figure 32: 07 Reddit User Analysis

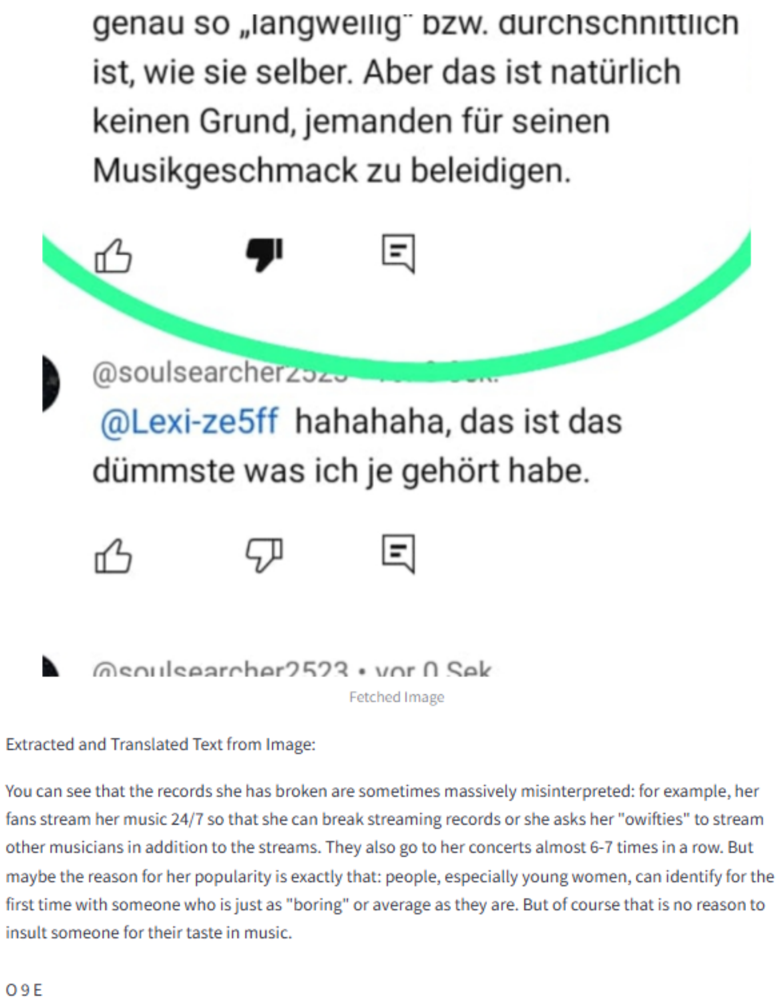

Figure 33: 08 Posts from user profile

The most frequently detected mental health concern is: normal appearing in 76.19% of analyzed text.

Mental health issue distribution across posts:

	Mental Health Issue	Count
0	normal	16
1	depression	5

Wellbeing Insight:

It's clear from the text that the user is experiencing frustration and feeling defensive about their love for Taylor Swift. This is a common experience for fans who are passionate about artists who aren't widely appreciated by mainstream culture. Here's a breakdown of the issues and actionable advice:

The Issues:

- **Social Pressure:** The user feels pressured to justify their love for Taylor Swift, often encountering negative comments and a lack of understanding from friends, family, and online communities. This is causing them to feel embarrassed, defensive, and questioning their own taste.
- **Misconceptions about Taylor Swift's success:** The user is frustrated by people who dismiss Taylor Swift's commercial success and international impact, often claiming she was "irrelevant" until recently. This creates a sense of injustice and makes it harder for the user to feel confident in their fandom.
- **Overgeneralization about Pop Music:** The user is experiencing a generalization about pop music, where it's seen as "generic" and lacking depth, leading to an unfair dismissal of Taylor Swift's artistry and songwriting.

Figure 34: 09 User profile Classification Result