Multimodal AI Framework for Social Media Based Mental Disorder Detection and Personalized Wellbeing Insights

Submitted by

SOUMYADEEP NANDY (13000121033)

PRITHWISH SARKAR (13000121037)

SAGNIK MUKHOPADHYAY (13000121040)

ARKAPRATIM GHOSH (13000121058)

Group 29

Final Year 8th Semester

June, 2025

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 **"Multimodal AI Framework for Social Media Based Mental Disorder Detection and Personalized Wellbeing Insights"** 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: 20th June, 2025

SOUMYADEEP NANDY (13000121033)

PRITHWISH SARKAR (13000121037)

SAGNIK MUKHOPADHYAY (13000121040)

ARKAPRATIM GHOSH (13000121058)

Table of Content

Abstract

This project employs AI to detect early signs of mental health issues by analyzing text, images, videos, audio, and PDFs shared on platforms like social media. It integrates advanced techniques such as facial expression recognition, image captioning, multilingual support, and psychological assessments like the Rorschach Inkblot Test and Ryff's Wellbeing Scale. Using an ensemble of models (Logistic Regression, SVM, LSTM, etc.), the system achieves 97.63% accuracy, with a hierarchical model reaching 96.25% on larger datasets. The web application supports real-time analysis, model retraining, and wellbeing surveys, providing actionable insights through visualizations and mapping mental health concerns to wellbeing parameters for timely interventions.

1 Introduction

1.1 Project Overview

Mental health disorders, including depression and anxiety, affect millions globally. Social media has become a platform where individuals often express emotions and struggles, inadvertently revealing signs of mental health challenges. This project leverages machine learning to analyze social media posts, detecting mental health issues by examining language patterns and contextual cues. By classifying posts indicative of mental health concerns, the project aims to enable early intervention and guide individuals toward appropriate support services.

1.2 Project Purpose

This project aims to develop a web application using machine learning and deep learning models (Logistic Regression, Naive Bayes, SVM, XGBoost, LSTM, Transformer) to detect mental health disorders from text, images, videos, PDFs, User responses to image and social media profiles. By addressing technical challenges in processing large datasets and optimizing algorithms, the project seeks to enable early detection of mental health issues, contributing to public health improvement through technology.

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 macOS, Windows 10/11, Linux distributions (Ubuntu, Linux Mint).
- **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** : This project leverages a comprehensive suite of libraries and frameworks to enable its functionality, including data processing (Pandas, NumPy), machine learning (Scikit-learn, XGBoost, TensorFlow, Transformers), image and text analysis (OpenCV, Tesseract, Pytesseract, DeepFace), audio processing (Librosa, PyDub, SpeechRecognition), social media integration (PRAW, Tweepy), and visualization (Plotly, Matplotlib). Additional tools like Streamlit, NLTK, and Google Generative AI ensure seamless application development, multilingual support, and advanced model implementation.
  - **Development Environment** : Google Colab is used 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 offers significant value in mental health monitoring, public health awareness, and social media governance. As mental health issues rise globally, organizations seek innovative solutions to address this crisis. By using machine learning to detect mental health disorders from social media data, this project can transform mental health management at both individual and societal levels, impacting various industries effectively.

- **Mental Health Services** : Mental health providers, including hospitals and therapy centers, can leverage machine learning to detect early signs of mental disorders from social media data. This proactive approach complements traditional self-reporting and clinical assessments, enabling earlier intervention and support for patients.

- **Social Media Platforms** : Social media platforms like Twitter and Reddit are key spaces for expressing thoughts and emotions, including mental health struggles. This project's machine learning models can help these platforms safeguard user well-being by identifying concerns early, while maintaining ethical standards.
- **Public Health Organizations** : Public health organizations can use real-time social media data to monitor mental well-being, identify trends, and design data-driven interventions. By analyzing language patterns, they can create targeted awareness campaigns that better engage individuals facing mental health challenges.

1.5 Glossary / Keywords

Term	Definition
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 for accessing Reddit data, such as posts, comments, and user information.

Term	Definition
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.
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.
DeepFace	DeepFace is a Python library for deep learning-based facial recognition and attribute analysis. It supports several pre-trained models and simplifies face recognition tasks, making it suitable for various applications in image analysis.
Transformers Module	The Transformers module in Python, developed by Hugging Face, is a library for natural language processing (NLP) tasks like text classification, translation, and summarization, using state-of-the-art models like BERT and GPT.
Gemini 1.5 Flash	Gemini 1.5 Flash is a cutting-edge AI model developed by Google, capable of performing advanced generative and analytical tasks across text, image, and other modalities.
FFmpeg	FFmpeg is a multimedia framework used for encoding, decoding, transcoding, streaming, and manipulating audio and video files, supporting a wide range of formats and codecs.
Hyperparameter Tuning	Hyperparameter tuning involves selecting the best parameters for a machine learning model to optimize its performance on a given task, using methods like grid search or random search.

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 [4].

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 [5].

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 [6].

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 [9].

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 [7].

The paper titled ”Single classifier vs. ensemble machine learning approaches for mental health prediction” (PMC9810771) explores the use of various machine learning techniques to predict mental health issues. Specifically, the study compares the performance of single classifiers (such as Logistic Regression, Gradient Boosting, Neural Networks, K-Nearest Neighbors, and Support Vector Machine) with ensemble machine learning approaches, which combine multiple classifiers to improve prediction accuracy. The study is based on a dataset of survey responses collected by Open Sourcing Mental Illness (OSMI), and the goal is to classify mental health issues based on these responses. The authors also compare newer machine learning techniques like Extreme Gradient Boosting (XGBoost) and Deep Neural Networks (DNN). The results indicate that Gradient Boosting achieved the highest accuracy (88.80%), followed by Neural Networks (88.00%). The ensemble classifier, which combined multiple models, achieved an accuracy of 85.60%. Overall, the paper demonstrates that machine learning techniques, particularly ensemble models, show promise for automated prediction of mental health problems, providing valuable insights for early diagnosis and intervention in mental health care [3].

The paper titled ”Ensemble of hybrid model based technique for early detecting of depression based on SVM and neural networks” (DOI: 10.1038/s41598-024-77193-0) presents a novel approach for early detection of depression using machine learning techniques. The study proposes an ensemble hybrid model combining Support Vector Machines (SVM) and Multilayer Perceptrons (MLP) to improve depression prediction accuracy. The DeprMVM hybrid model serves as a meta-learner, where the SVM and MLP networks act as level-0 learners. The model addresses class imbalance by applying the Synthetic Minority Over-sampling Technique (SMOTE) and

cluster sampling, which improves detection accuracy and reduces the risk of overfitting. The study finds that the ensemble approach achieved an accuracy of 99.39% and an F1-score of 99.51%, outperforming previous models in depression detection. This paper highlights the potential of ensemble hybrid models for early detection of depression and their applicability in mental health care [8].

The paper titled ”Survey of transformers and towards ensemble learning using transformers for natural language processing” (DOI: 10.1186/s40537-023-00842-0) provides a comprehensive review of transformer models in natural language processing (NLP). The study compares several prominent transformer-based models, including BERT, XLNet, RoBERTa, GPT-2, and ALBERT, evaluating their performance across multiple NLP tasks such as sentiment analysis, question answering, and text generation. The authors also introduce ensemble learning approaches using these models to enhance task performance. The results demonstrate that ensemble models outperform single classifier approaches, offering significant improvements for specific NLP tasks. This paper highlights the potential of ensemble learning with transformer models and their versatility in solving complex NLP challenges [10].

These studies collectively underscore the growing body of evidence supporting the integration of social media analytics and machine learning for mental health detection.

Sl no	Paper Title and Author	Year	Aim and Objective	Uniqueness claimed	Outcome achcived	Algorithm/Tools/Platform used	Feature Extraction	Modality of data	Data Resource
1	Combining Sentiment Analysis Models Using Stacking Ensemble Learning Techniques on BIST30 Stocks by Mahmut Sami SIVRI	2024	Develop a stacking ensemble model to enhance sentiment analysis accuracy and robustness for BIST30 financial news.	Integrate diverse models (LSTM, BERT, Naive Bayes, SVM) in a stacking ensemble with a focus on BIST30 stocks, robust data strategies, and comparative analysis.	Enhance financial sentiment analysis with superior performance, generalization over baseline models with accuracy of 85%	Stacking ensemble with LSTM, BERT, Naive Bayes, SVM, and Logistic Regression, trained on 5,287 financial news articles (2020-2023) with an 80-10-10 split.	SVM uses TF-IDF with text preprocessing: cleaning, tokenization, stopwords removal, and lemmatization.	Financial news articles classified into Positive, Negative, and Neutral sentiment categories.	Thomson Reuters news articles
2	Predicting mental health using social media: A roadmap for future development by Ramin Safa, S. A, Edalatpanah, Ali Sorourkhah	2022	A roadmap for analyzing and predicting mental disorders using social media data, covering collection, feature extraction, and prediction methods.	Comprehensive analysis of mental state assessment methods on social data, categorizing approaches and discussing challenges and future directions.	Conceptual framework for e-mental health research via social media, covering assessment strategies, data methods, algorithms, metrics, and future challenges.	Machine learning, NLP, and sentiment analysis using platforms like Twitter and Facebook, with tools like LIWC and frameworks such as CNN and LSTM.	Tools like LIWC, OpinionFinder, SentiStrength, VADER, Word2Vec, LDA, tf-idf, VGG-Net, and Imagga	Textual Data and Visual Content	myPersonality project CLPsych workshop data, eRisk workshops data, AutoDep dataset, SDCNL dataset, CAMS dataset
3	Detecting Depression and Mental Illness on Social Media: An Integrative Review by Sharath Chandra Guntuku, David B. Yaden, Margaret L. Kern, Lyle H. Ungar, Johannes C. Eichstaedt	2016	Reviewing recent studies and comparing approaches for predicting mental illness through social media analysis.	Comprehensive review comparing mental illness detection methods and prediction performances with clinical baselines.	Social media-based screening shows AUCs between 0.70 and 0.91, bridging clinician assessment and screening surveys with accuracy of 87%	Using Linear Regression, SVM, Neural Networks, and Random Forest with analysis tools like LIWC and LabMT.	LIWC and N-Gram	Textual Data	Survey and Social media
4	Single classifier vs. ensemble machine learning approaches for mental health prediction by Jetli Chung and Jason Teo	2023	Empirically evaluate and compare single classifiers and ensemble approaches for predicting mental health problems.	Comprehensive comparison of traditional algorithms, Deep Neural Networks, XGBoost, and ensemble methods for mental health prediction using survey data.	Gradient Boosting led with 88.80% accuracy, followed by Neural Networks (88.00%) and XGBoost (87.20%).	Logistic Regression, Gradient Boosting, Neural Networks, K-NN, SVM, Deep Neural Networks, XGBoost, and Ensemble Voting Classifier.	Extra Trees Classifier was used for feature selection to reduce overfitting	Survey	OSMI's 2014 Mental Health in Tech Survey measured workplace mental health attitudes in the tech industry, available under Creative Commons.
5	Survey of transformers and towards ensemble learning using transformers for natural language processing by Hongzhi Zhang and M. Omair Shafiq	2024	Compare and analyze transformer models across six NLP tasks, developing ensemble models to leverage their strengths.	Compare 5 transformer models across 6 NLP tasks, analyzing nuances and developing novel ensemble approaches.	Outperformed single classifiers with two ensemble models, identifying optimal combinations for specific NLP tasks with accuracy of 79%	BERT, XLNet, GPT2, RoBERTa, and ALBERT on Google Colab using Transformers, Deep Neural Networks, BERTopic, EncoderDecoderModel, and Seq2SeqTrainer.	Using transformer-based embeddings, word vectors, BERTopic for topic modeling, class-based TF-IDF, and contextual semantic representations.	Using transformer embeddings, BERTopic, TF-IDF, and contextual semantic representations for NLP.	Using datasets like Kaggle's coronavirus tweets, SQuAD1.1, Groningen Meaning Bank, CNN daily mail, disaster tweets, and Trump 2020 election speeches for various NLP tasks.
6	Ensemble of hybrid model based technique for early detecting of depression based on SVM and neural networks by Dip Kumar Saha, Tuhin Hossain, Mejdl Safran, Sultan Alfarhood, M. F. Mridha & Dunren Che	2024	Develop an automated system to detect depression using psychological and sociodemographic characteristics.	Develop an automated system to detect depression using psychological and sociodemographic characteristics.	Developed an automated depression detection system with improved accuracy and reduced class imbalance,with accuracy of 99.39%	Algorithms like SVM, MLP, Hybrid DeprMVM, RFC, KNN, XGB with SMOTE, cluster sampling, and Scikit-learn for depression detection.	Using Label Encoder, Standard Scaler, and feature selection techniques for data preprocessing.	Survey data with 30 psychosocial/demographic predictors and 1 depression status target, including 25 BDC questions.	Dataset of 604 Bangladeshi participants (65.7% depressed, 34.3% not) collected from April to August 2020.
7	Machine Learning-based Approach for Depression Detection in Twitter Using Content and Activity Features by Hatoon AlSagri and Mourad Ykhlef (2023)	2023	Detect depression in Twitter users by analyzing tweet content and network behavior using machine learning techniques.	Incorporating tweet text, user behavior, and new features like activity categories, Dept_Sent lexicon, and synonyms for enhanced depression detection.	Developed a binary classification model with improved accuracy by combining user activities and tweets for better mental health detection with accuracy 82.5%	Using SVM, Naive Bayes, and Decision Tree algorithms with R 3.3, RStudio, Twitter API, WordNet, and R packages for data analysis.	Feature extraction using TF-IDF, WordNet, Information Gain, sentiment analysis tools, and R packages.	Text data (tweets, pronouns, sentiment words, depression terms), and user activity data (following, posts, mentions, retweets, emojis, hashtags).	Collected over 300,000 tweets from 111 users via Twitter API, with manual verification of self-disclosed depression and data selection criteria.
8	Predicting Depression via Social Media' Authors & Year: Mumnun De Choudhury, Michael Gamon, Scott Counts, Eric Horvitz	2013	Leveraging social media behavioral patterns to detect and predict Major Depressive Disorder (MDD) early.	Pioneering crowdsourced clinical data and multifaceted behavioral analysis for early depression prediction.	Achieved 70% accurate depression prediction, highlighting key behavioral and linguistic markers.	Utilized SVM with RBF kernel, PCA, and 10-fold cross-validation on Twitter data for depression analysis.	Employed LIWC, ANEW, and custom lexicons for linguistic, emotional, and medication-related feature extraction.	Analyzed Twitter posts, surveys (CES-D, BDI), demographics, and social network data for depression detection.	Collected data from 1,583 U.S.-based crowdworkers, analyzed 2M+ tweets and surveys (CES-D, BDI) with strict quality controls.
9	Predicting Mental Health Illness using Machine Learning Algorithms by Konda Vaishnavi	2022	Evaluate and compare machine learning techniques for accurate mental health issue prediction.	Unique comparison of five ML techniques for mental health prediction using diverse accuracy metrics and ROC curves.	Achieved 81.75% accuracy with stacking, with all classifiers showing strong ROC values (0.8–0.9).	Utilized Logistic Regression, K-NN, Decision Tree, Random Forest, and Stacking for mental health prediction.	Performed feature selection to reduce 27 attributes to 8 for mental health prediction.	Used a dataset with 27 attributes and 1259 entries, including text documents for mental health prediction.	The paper lacks details on dataset sources, collection method, attributes, time period, and demographics.
10	Generalizability of Machine Learning to Categorize Various Mental Illness Using Social Media Activity Patterns by Ang, C.S. & Venkatachala, R.	2023	Explore linguistic patterns and cross-platform machine learning models for classifying mental health groups based on social media activity.	Improved accuracy by 2.11% with simpler cross-platform models for mental health classification on Twitter and Reddit.	Achieved 91.5% accuracy, with better Reddit model generalization and distinct linguistic patterns between platforms.	Used CNN, Word2Vec, NLTK, and Google Colab for Twitter and Reddit-based mental health classification.	Employed LIWC, Porter Stemmer, and NLTK for feature extraction in mental health classification.	Analyzed 606K Reddit posts and 23M tweets covering 6 mental health conditions from relevant subreddits and hashtags.	Used 23M Twitter tweets and 606K Reddit posts from mental health-related subreddits for cross-platform model training and testing.

Figure 1: Pre frame study review

3 Problem Definition and Preliminaries

3.1 Context and Background

Mental health disorders affect approximately 1 in 8 people globally. Social media platforms like Reddit and Twitter offer vast amounts of real-time data reflecting mental health struggles, but the unstructured nature of this data poses challenges for effective identification and categorization of specific disorders.

3.2 Objective

The primary objective is to develop a system that uses NLP and machine learning to analyze Reddit and Twitter posts for detecting mental health disorders like depression, anxiety, bipolar disorder, and PTSD. It seeks to **classify posts** accurately and provide **data-driven insights** into mental health trends for researchers, professionals, and policymakers.

3.3 Challenges

- **Data Variability**: Social media posts vary in structure, style, and language.
- **Imbalanced Data**: Uneven distribution of mental issues can impact model training.
- **Cultural Nuances**: Mental health discussions differ across cultures.
- **Privacy and Ethics**: Analyzing social media data raises concerns about user privacy.

3.4 Scope

The scope of the project is to develop a multimodal AI framework to detect mental disorders from social media inputs using ML and NLP. It analyzes text, images, videos, PDFs, dynamic responses to image shown and Reddit/Twitter data via APIs, classifying inputs into Normal, Anxiety, Depression, Bipolar, or PTSD using a Reddit dataset. Finally an association is created between mental health disorder and mental wellbeing paramters from Ryff's Psychological Well-being Scale.

3.5 Exclusions

This project excludes real-time sentiment analysis, platform-specific features like hashtags or subreddits, and ethical implications of data ownership. It also does not analyze comments, metadata, or Reddit/Twitter-specific elements, focusing solely on detecting mental health disorders from user profiles and mapping them to wellbeing insights, though the dataset has been made solely from Reddit posts from various subreddits.

3.6 Assumptions

The project assumes that the Reddit dataset obtained via PRAW represents diverse mental health discussions and that user posts accurately reflect emotions. NLP techniques are presumed effective for sentiment classification, and selected ML models (Logistic Regression, SVM, Naïve Bayes, LSTM, Transformer, XGBoost) are expected to perform optimally. Social media sentiments are considered valid proxies for public mental health perceptions. Data preprocessing is assumed sufficient to reduce noise, and ethical standards are maintained to protect user privacy. The user responses well-being surveys are considered valid for updating the association matrix between mental health disorders and well-being parameters.

4 Proposed Solution

This project ”Multimodal AI Framework for Social Media-Based Mental Disorder Detection and Personalized Wellbeing Insights,” includes the below contributions.

4.1 Special Contributions

- **Dataset Acquisition**: Reddit data was sourced using PRAW from relevant subreddits (Normal, Depression, Anxiety, Bipolar, PTSD). Extensive preprocessing ensured data integrity.
- **Text Vectorization**: TF-IDF, Bag-Of-Words model weree used to convert text into numerical format via Scikit-learn, enabling efficient feature extraction and model training. Other vectorization techniques like Word2Vec, LIWC, N-Grams were also explored.
- **Machine Learning Models**: Logistic Regression, SVM, Naïve Bayes, LSTM, Transformer, and XGBoost were implemented for multi-class classification of mental health conditions.
- **Model Evaluation**: Accuracy, precision, recall, and F1-score were used to assess performance. Confusion matrices and ROC curves were also generated for model evaluation.
- **Insights & Recommendations**: Findings inform mental health professionals and policymakers on probable issues and wellbeing insights.
- **Documentation & Reproducibility**: Detailed documentation ensures usability, including methodology, code, and instructions for result reproduction.

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.
- **Machine and Deep Learning Model Functions** : Functions for implementing Logistic Regression, Naive Bayes, Support Vector Machine, Random Forest, XGboost, Long Short Term Memory and Transformer 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

The project utilized an Iterative Waterfall model, combining structured progression with flexibility for revisions. This approach allowed revisiting earlier phases, such as refining requirements or enhancing data preprocessing, based on insights from analysis or testing. Feedback loops between phases ensured continuous improvement, fostering a robust system for detecting mental health disorders in social media posts while adapting to dynamic project needs.

The different phases are as follows:

- **Requirement Gathering and Analysis** : This initial phase involved understanding the project's goals, objectives, and stakeholder expectations.
- **Data Collection and Preparation** : Utilizing Reddit API, the data collection phase was executed. This included downloading the dataset, examining its structure, and performing data cleaning and prepossessing to ensure its suitability for analysis.
- **Model Development** : This phase included the creation of a Bag of Words model, splitting the dataset into training and test sets, and implementing various machine learning algorithms.

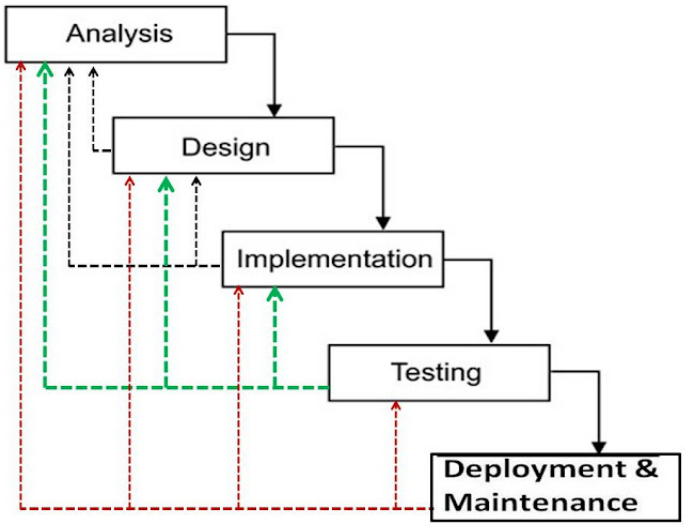


Figure 2: Iterative Waterfall Model

- **Model Evaluation** : Following model development, testing and validation of the models were done, 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 focused on deploying the best-performing model and creating comprehensive documentation. This included user manuals and technical documentation to facilitate future maintenance and enhancements.

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 vital for project success. A detailed timeline with tasks like requirement gathering, data preprocessing, model implementation, and testing was created, with flexibility for adjustments based on feedback. Key milestones, including data analysis, model validation, and user acceptance testing, ensure progress tracking. Using tools like Microsoft Project, we monitor tasks, manage resources, and maintain communication to deliver a high-quality solution on time.

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	% Complete
1		**GR29 MAFSMBMDDPWI**	**243 days**	**Mon 01-07-24**	**Mon 02-06-25**		**100%**
2		**Phase 1: 7th Semester Activities**	**150 days**	**Mon 01-07-24**	**Wed 22-01-25**		**100%**
3		**Project Startup**	**20 days**	**Mon 01-07-24**	**Fri 26-07-24**		**100%**
4		Team Building	2 days	Mon 01-07-24	Tue 02-07-24		100%
5		Brainstorm on Project Topic	2 days	Wed 03-07-24	Thu 04-07-24	4	100%
6		Project agreed with Guide	2 days	Fri 05-07-24	Mon 08-07-24	5	100%
7		Related Study& Documentation	4 days	Mon 08-07-24	Thu 11-07-24	6	100%
8		Deliver Project Synopsis for Guide's review	2 days	Fri 12-07-24	Mon 15-07-24	7	100%
9		Close review feedbacks	9 days	Mon 15-07-24	Thu 25-07-24	8	100%
10		**Project Synopsis Finalized**	**1 day**	**Fri 26-07-24**	**Fri 26-07-24**	**9**	**100%**
11		**Requirement Analysis**	**17 days**	**Thu 01-08-24**	**Fri 23-08-24**		**100%**
12		Gather Requirements	7 days	Thu 01-08-24	Fri 09-08-24	10	100%
13		Prepare Draft Requirement Matrix	9 days	Mon 12-08-24	Thu 22-08-24	10	100%
14		**Requirement Matrix Finalized**	**1 day**	**Fri 23-08-24**	**Fri 23-08-24**	**13**	**100%**
15		**Design**	**46 days**	**Mon 26-08-24**	**Thu 24-10-24**	**14**	**100%**
16		**Detailed Design**	**25 days**	**Mon 26-08-24**	**Thu 26-09-24**	**14**	**100%**
17		Data Collection	3 days	Mon 26-08-24	Wed 28-08-24	14	100%
18		Data Preprocessing	4 days	Thu 29-08-24	Mon 02-09-24	17	100%
19		Model Training and Evaluation	18 days	Tue 03-09-24	Thu 26-09-24	18	100%
20		**Test Plan Preparation**	**21 days**	**Fri 27-09-24**	**Thu 24-10-24**	**19**	**100%**
21		Text Classification	4 days	Fri 27-09-24	Wed 02-10-24	19	100%
22		Image Classification	9 days	Thu 03-10-24	Tue 15-10-24	21	100%
23		Video Classification	4 days	Wed 16-10-24	Sat 19-10-24	22	100%
24		Reddit and Twitter User Analysis	4 days	Mon 21-10-24	Thu 24-10-24	23	100%
25		**Phase 1 Closure**	**59 days**	**Fri 01-11-24**	**Wed 22-01-25**	**3**	**100%**
26		Prepare 7th Semester Project Rep	14 days	Fri 01-11-24	Wed 20-11-24	20	100%
27		**Updated Requirement Matrix**	**2 days**	**Thu 21-11-24**	**Fri 22-11-24**	**26**	**100%**
28		**Updated Project Plan**	**1 day**	**Fri 22-11-24**	**Fri 22-11-24**	**27**	**100%**
29		Project Viva	2 days	Mon 20-01-25	Tue 21-01-25	28	100%
30		**Approved Project Report - 7th Semester**	**1 day**	**Wed 22-01-25**	**Wed 22-01-25**	**29**	**100%**
31		**Semester Gap**	**9 days**	**Thu 23-01-25**	**Tue 04-02-25**	**30**	**100%**
32		**Phase 2: 8th Semester Activities**	**84 days**	**Wed 05-02-25**	**Mon 02-06-25**	**31**	**100%**
33		**Coding & Únit Testing**	**27 days**	**Wed 05-02-25**	**Thu 13-03-25**	**31**	**100%**
34		Data Collection and Preprocessing	7 days	Wed 05-02-25	Thu 13-02-25	31	100%
35		Model Training and Evaluation	6 days	Fri 14-02-25	Fri 21-02-25	34	100%
36		Web Application Components	14 days	Mon 24-02-25	Thu 13-03-25	35	100%
37		**System Integration Testing**	**45 days**	**Fri 14-03-25**	**Thu 15-05-25**	**36**	**100%**
38		API Calls	23 days	Fri 14-03-25	Tue 15-04-25	36	100%
39		Deployment	22 days	Wed 16-04-25	Thu 15-05-25	38	100%
40		**Project Closure**	**14 days**	**Wed 14-05-25**	**Mon 02-06-25**	**39**	**100%**
41		Prepare 8th Semester Project Report	5 days	Wed 14-05-25	Tue 20-05-25	39	100%
42		**Updated Requirement Matrix**	2 days	Wed 21-05-25	Thu 22-05-25	41	100%
43		**Updated Project Plan**	5 days	Fri 23-05-25	Thu 29-05-25	42	100%
44		Review by Faculties	1 day	Fri 30-05-25	Fri 30-05-25	43	100%
45		**Approved Project Report - 8th Semester**	1 day	Mon 02-06-25	Mon 02-06-25	44	100%

Figure 3: Project Plan

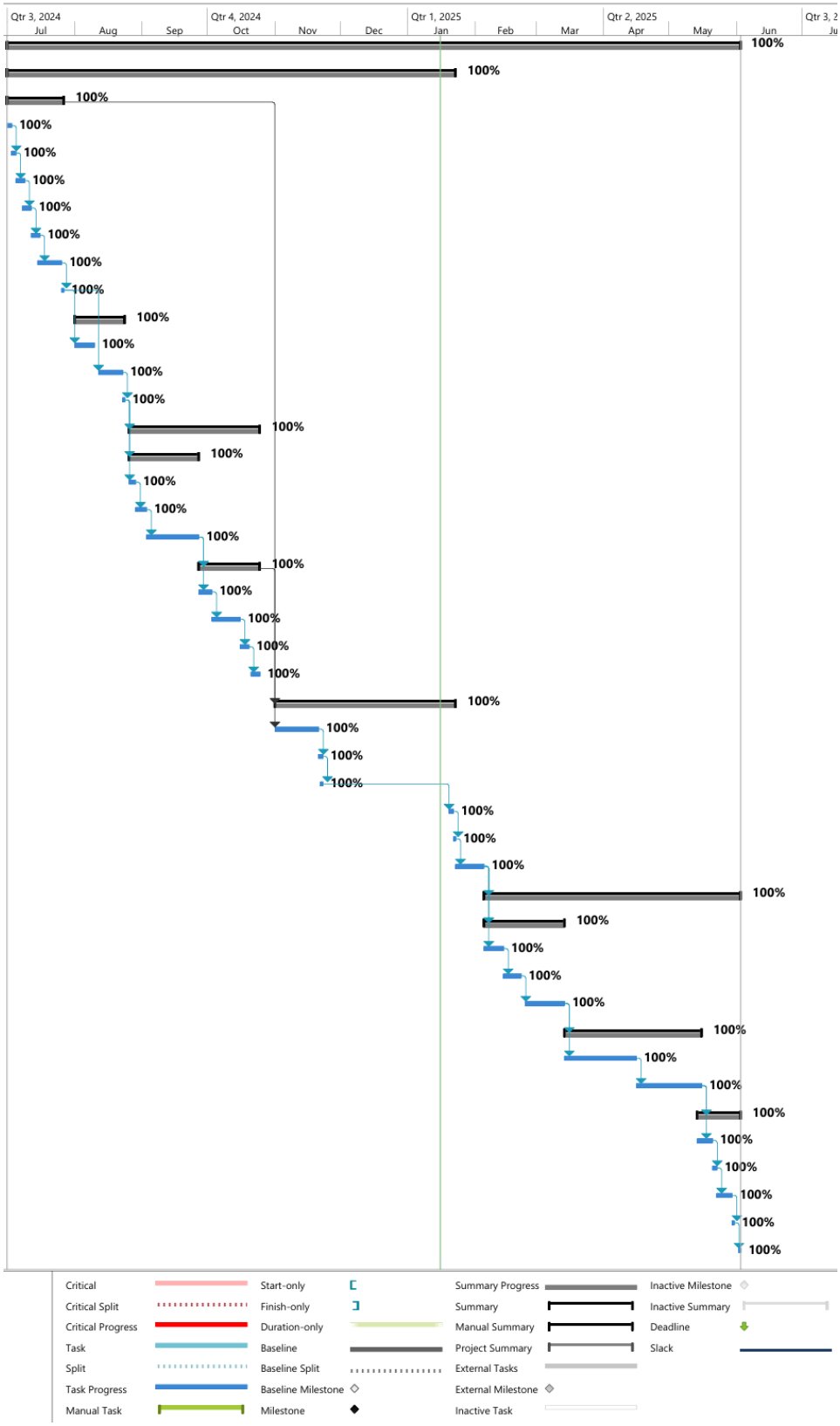


Figure 4: Gantt Chart

6 Requirement Analysis

6.1 Requirement Matrix

Rqmt ID	*Requirement Item*	*Requirement Analysis Status*	*Design Module (As per Prototype folder structure)*	*Design Reference (section# under project Report)*
FR-001	Collect social media data from Reddit.	Completed	D01	8.2.1
FR-002	Implement data cleaning and preprocessing.	Completed	D02	8.2.2
FR-003	Train machine learning and deep learning models.	Completed	D03	8.2.3 - 8.2.10
FR-004	Evaluate models using performance metrics (accuracy, recall, F1 Score, Support).	Completed	D03	9.2 - 9.9
FR-005	Text Analysis	Completed	D04	APPENDIX A - PROTOTYPE
FR-006	Image Upload Analysis	Completed	D04	APPENDIX A - PROTOTYPE
FR-007	Video Upload Analysis	Completed	D04	APPENDIX A - PROTOTYPE
FR-008	PDF Upload Analysis	Completed	D04	APPENDIX A - PROTOTYPE
FR-009	User response to image	Completed	D04	APPENDIX A - PROTOTYPE
FR-010	Reddit and Twitter Username Analysis	Completed	D04	APPENDIX A - PROTOTYPE
FR-011	Wellbeing survey and mapping using association matrix	Completed	D04	APPENDIX A - PROTOTYPE
FR-012	Application Deployment and Model Retraining	Completed	D05	APPENDIX A - PROTOTYPE
NFR-001	Scalability and Performance	Completed	D03	10 - 11

Figure 5: Requirement Matrix

6.2 Requirement Elaboration

No	Requirement	Input	Output
FR-001	Collect social media data from Reddit	API queries	Raw text data
FR-002	Implement data cleaning and preprocessing	Raw text data	Cleaned, structured text
FR-003	Train machine learning and deep learning models	Preprocessed data	Trained models
FR-004	Evaluate models using performance metrics	Trained models	Accuracy, Recall, F1 Score
FR-005	Text Analysis	User text input	Sentiment classification
FR-006	Image Upload Analysis	Uploaded image	Extracted text and sentiment
FR-007	Video Upload Analysis	Uploaded video	Extracted frames and sentiment
FR-008	PDF Upload Analysis	Uploaded PDF	Extracted text and analysis
FR-009	User response to image	User input	Sentiment classification
FR-010	Reddit and Twitter Username Analysis	Username input	User sentiment trends
FR-011	Wellbeing survey and mapping using association matrix	Survey responses	Mental health insights
FR-012	Application Deployment and Model Retraining	Updated dataset	Improved model accuracy within the web application
NFR-001	Scalability and Performance	High user load	Efficient response time

Functional and Non-Functional Requirements

7 Design

7.1 Technical Environment

The technical environment for the project ”Multimodal AI Framework for Social Media Based Mental Disorder Detection and Personalized Wellbeing Insights” comprises a combination of hardware, software, and tools that enable smooth data analysis, machine learning model training, and deployment. Below is aN overview of the minimum hardware configuration, software tools, and package details necessary to carry out this project effectively.

Component	Specification
Processor	Intel Core i5 (or equivalent) with a base clock speed of at least 2.5 GHz. A multi-core processor is preferred for parallel processing, essential for model training and data preprocessing.
RAM	8 GB recommended for handling data loading, cleaning, and transformation. For large datasets, 16 GB is ideal to prevent memory overflow and processing delays.
Storage	Minimum 256 GB SSD recommended. SSD offers faster read/write speeds, significantly improving dataset loading times, especially for large datasets like Reddit-based social media posts.
GPU	Not necessary for basic ML tasks like Logistic Regression or SVM. For deep learning, an NVIDIA GTX 1060 with 4 GB VRAM or higher is 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 OS should support ML libraries and project tools.

Minimum Hardware Configuration

Library/Package	Description
Python	Primary programming language for data processing and model training.
pandas	Data manipulation and preprocessing.
scikit-learn	Machine learning models and evaluation tools.
Streamlit	Web framework for deploying interactive ML applications.
pyngrok	Creates secure tunnels for sharing local applications.
Google Colab	Cloud-based Python environment with GPU/TPU support.
PRAW	Access and retrieve Reddit data via API.
pytesseract	OCR library for extracting text from images.
Pillow	Image processing and handling library.

Libraries and Packages Used

Library/Package	Description
joblib	Serialization and model saving utility.
protobuf	Efficient data serialization format by Google.
deep-translator	Multilingual text translation.
Requests	Library for making HTTP requests.
google-generativeai	Integrate Google's generative AI models.
ffmpeg	Multimedia framework for processing audio/video.
tesseract-ocr	OCR tool for text extraction.
portaudio19-dev	Required for handling audio input/output.
poppler-utils	Converts PDFs to images for text extraction.
pydub	Audio processing library.
sounddevice	Real-time audio recording/playback.
wavio	WAV file handling.
numpy	Numerical computing library.
PyAudio	Audio input/output handling.
SpeechRecognition	Converts speech to text.
pdf2image	Converts PDFs to images.
tweepy	Access and analyze Twitter data.
openai-whisper	Speech-to-text model from OpenAI.
tiktoken	Tokenizer for language models.
librosa	Audio analysis and feature extraction.
opencv-python	Computer vision and image processing.
xgboost	Gradient boosting for ML models.
deepface	Facial recognition and emotion analysis.
tf_keras	Keras API for TensorFlow.
transformers	Pre-trained NLP models from Hugging Face.
tensorflow	Deep learning framework.
nltk	Natural language processing toolkit.
plotly	Interactive data visualization.
matplotlib	Static and animated plots.
scipy	Scientific computing and signal processing.
networkx	Graph analysis and visualization.
yt-dlp	YouTube video/audio downloader.

Libraries and Packages Used

7.2 Hierarchy of Modules

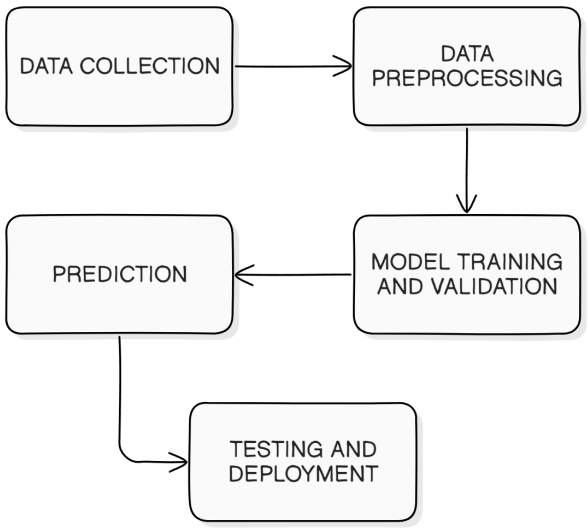


Figure 6: Project Modules

Module	Description
Data Collection	Gathers relevant text data from CSV files and platforms like Reddit via PRAW.
Data Preprocessing	Cleans and prepares text data using: • Tokenization (splitting text into words/tokens) • Lowercasing for uniformity • Stop-word removal • Lemmatization or stemming
Feature Extraction	Converts text into numerical features using: • Term Frequency-Inverse Document Frequency (TF-IDF) • Bag of Words (BoW) model • Word2Vec, LIWC (Linguistic Inquiry and Word Count) and N-Gram were also explored
Model Training and Validation	Splits dataset into training/testing sets and trains models such as: • Logistic Regression, Naive Bayes, SVM, XGBoost, KNN, LSTM, Transformer. These formed the base models for the ensemble model used in application where Random Forest was used as the meta-learner All were validated to assess performance.
Prediction	Processes received and combined text input for classification using trained models.
Testing and Deployment	Deploys models on Streamlit Cloud for real-time predictions and wellbeing insights using association matrix. Provides a user interface for easy access.

Hierarchy of Modules in the system

7.3 Detailed Design

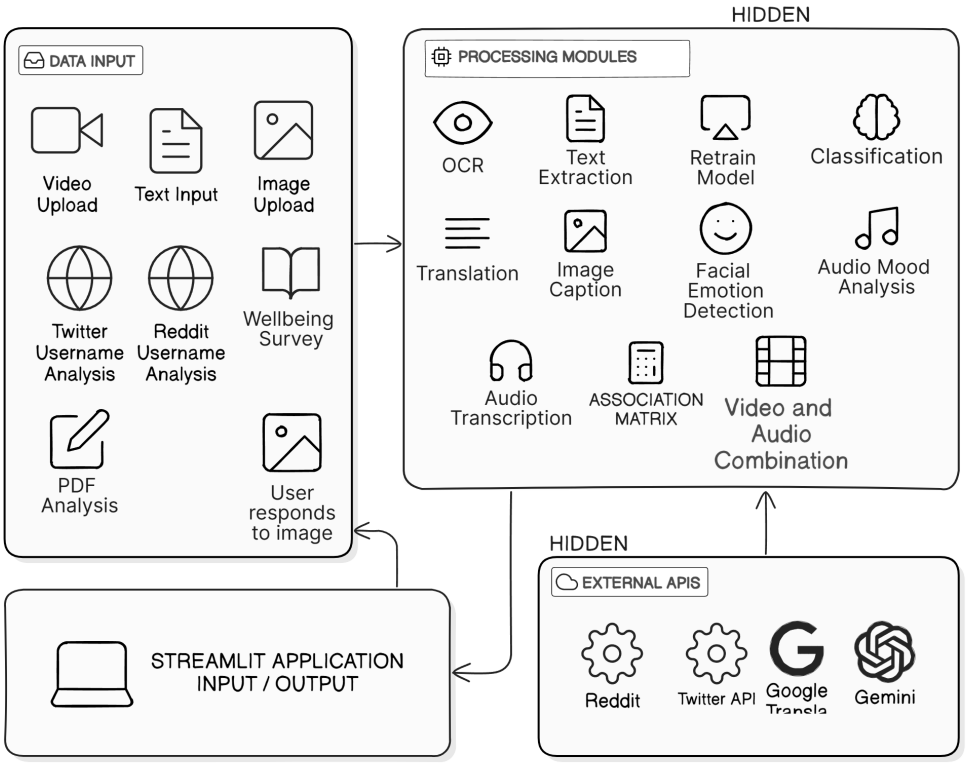


Figure 7: System Overview

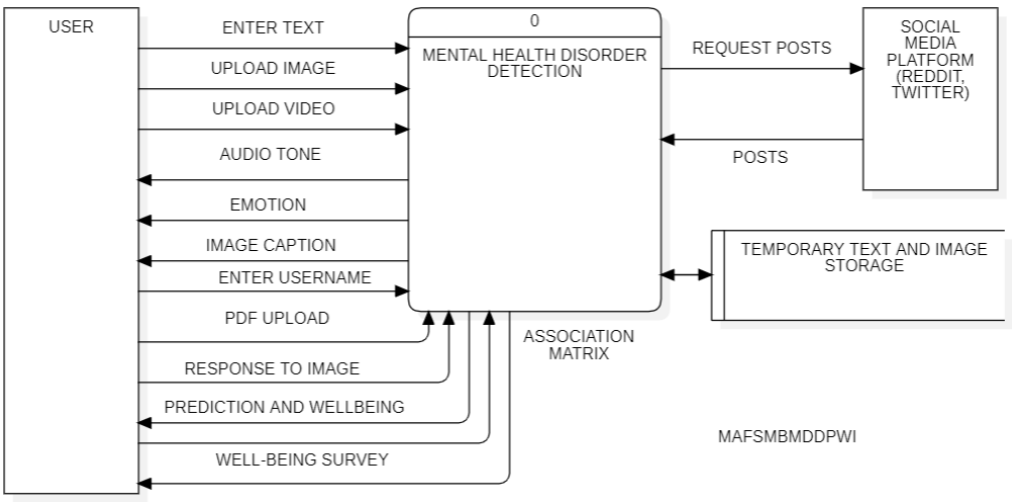


Figure 8: DFD Level 0 of the System

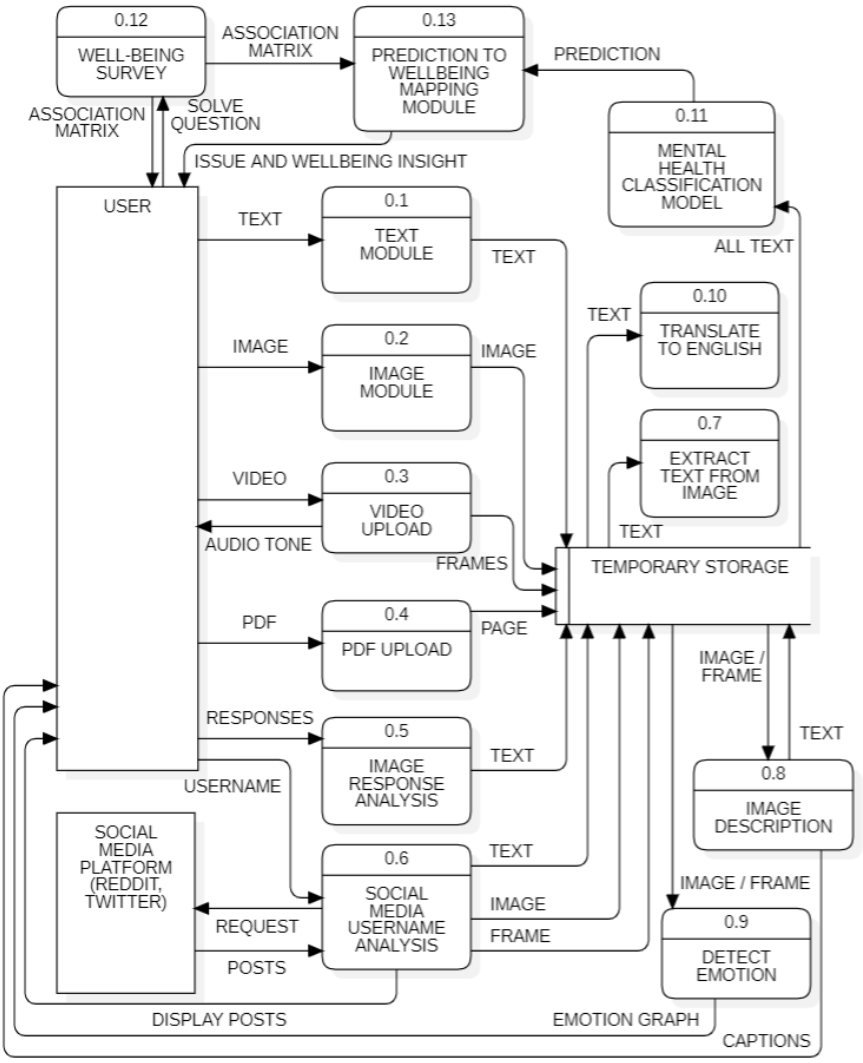


Figure 9: DFD Level 1 of the System

7.4 Image Description

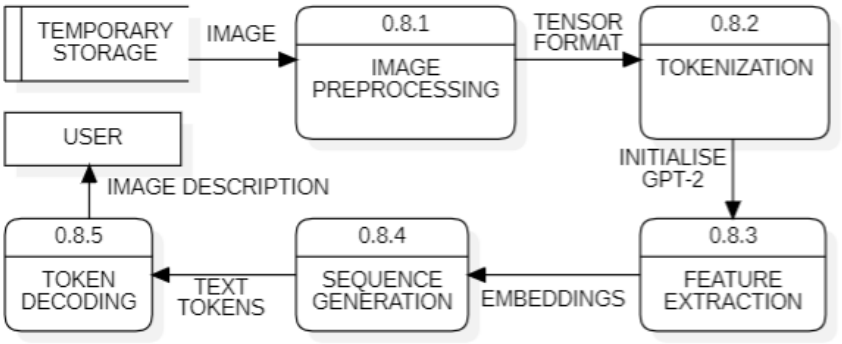


Figure 10: DFD Level 2 of Image Description

The image captioning process using Python's Transformers module and the ViT-GPT2 model involves several key steps. The user uploads an image, which is preprocessed by resizing to 224×224 pixels, normalizing pixel values, and converting it into a tensor format. The Vision Transformer (ViT) divides the image into 16×16 patches, embeds them, and processes them through transformer layers to extract visual features. The resulting embedding is passed to GPT-2, which generates a sequence of text tokens iteratively using its language model. Tokens are decoded back into human-readable text using the GPT-2 tokenizer, cleaned for readability, and output as a concise description. This pipeline leverages ViT for feature extraction and GPT-2 for language generation, enabling efficient and almost accurate image captioning.

7.5 Emotion Detection Functionality

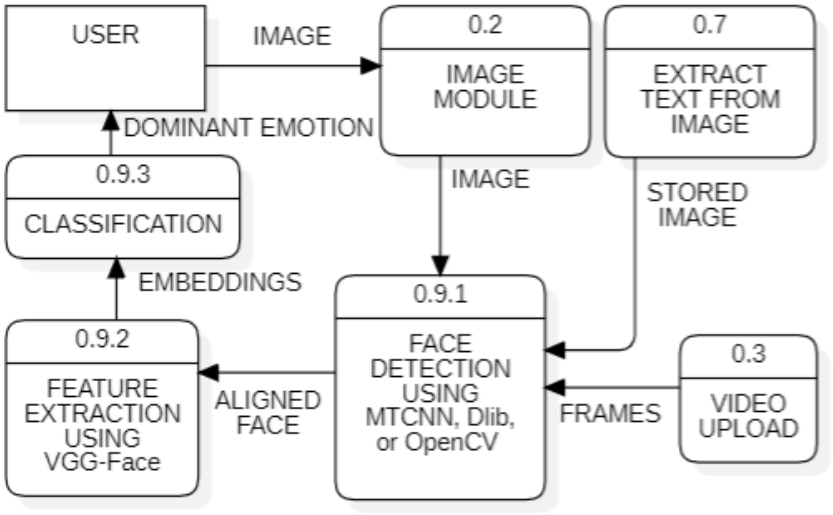


Figure 11: DFD Level 2 of Emotion Detection Functionality

DeepFace analysis processes facial features from uploaded images or video frames through a structured pipeline. It starts with face detection using models like MTCNN, Dlib, or OpenCV to locate and align faces. The detected faces are cropped and passed to feature extraction, where pre-trained models like VGG-Face or Facenet generate numerical embeddings. These embeddings are compared using cosine similarity or Euclidean distance for tasks like emotion detection (e.g., happiness, sadness), demographic analysis (e.g., age, gender), or face verification. The results, such as detected emotions or attributes, are then displayed to the user. This process enables real-time analysis of facial expressions and emotions, providing valuable insights for mental health monitoring and wellbeing assessment.

7.6 Extract Text From Image

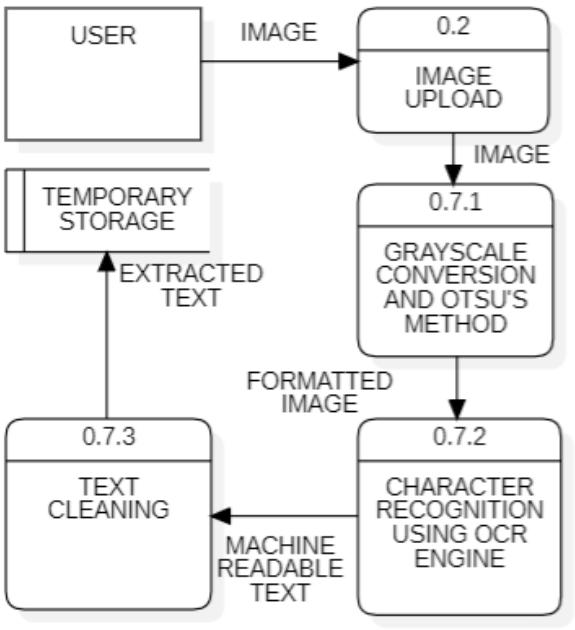


Figure 12: DFD Level 2 of Text Extraction from Image

The process of extracting text from an image using Tesseract-OCR begins with preprocessing the uploaded image. The image is converted to grayscale, noise is reduced using Gaussian or Median Blur, and binarization (e.g., Otsu's thresholding) isolates text from the background. Text regions are detected using contour analysis or connected components. The processed image is then passed to Tesseract, which uses an LSTM-based neural network to recognize characters and generate machine-readable text, enhanced by language models for accuracy. Postprocessing corrects errors via spell-checking and rule-based replacements (e.g., `0` $\rightarrow$ `O`), and formats the text into paragraphs or lines. The final output, displayed or saved as `.txt` or `.docx`, is ready for applications like document analysis. This pipeline combines image enhancement, deep learning, and text correction for high-quality results.

7.7 Translation to English

The process of translating text to English using DeepTranslator begins with preprocessing the input to clean unwanted characters and detect the source language using machine learning models like Google's Compact Language Detector. The prepared text is passed to DeepTranslator, which interfaces with APIs like Google Translate or Microsoft Translator. These APIs use neural machine translation (NMT) with encoder-decoder architectures and attention mechanisms to translate the text into English. Transformer-based models enhance accuracy by capturing context and long-range dependencies. Postprocessing ensures quality by validating completeness,

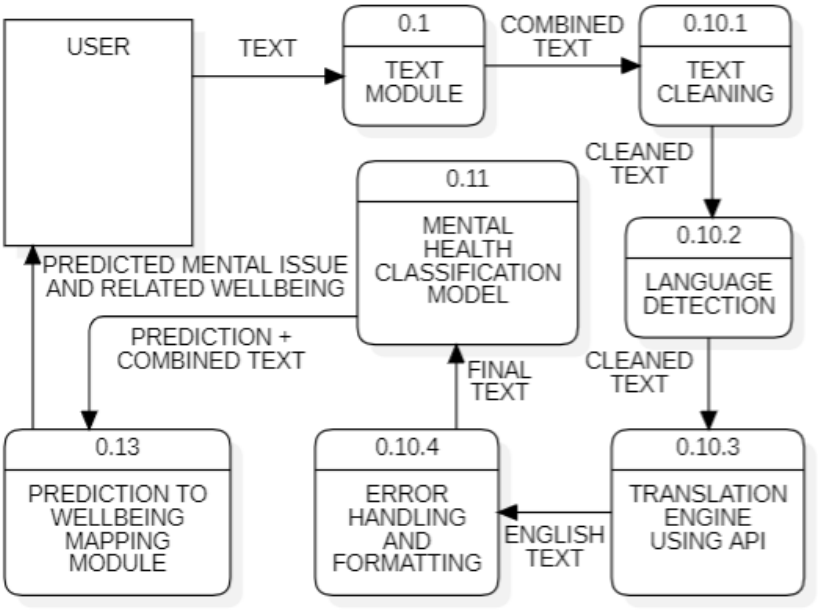


Figure 13: DFD Level 2 of Translation to English

correcting errors, and preserving formatting. The final translated text is displayed or saved, ensuring accuracy and readability. This pipeline integrates NLP, deep learning, and error handling for reliable translations.

7.8 Audio Mood Analysis

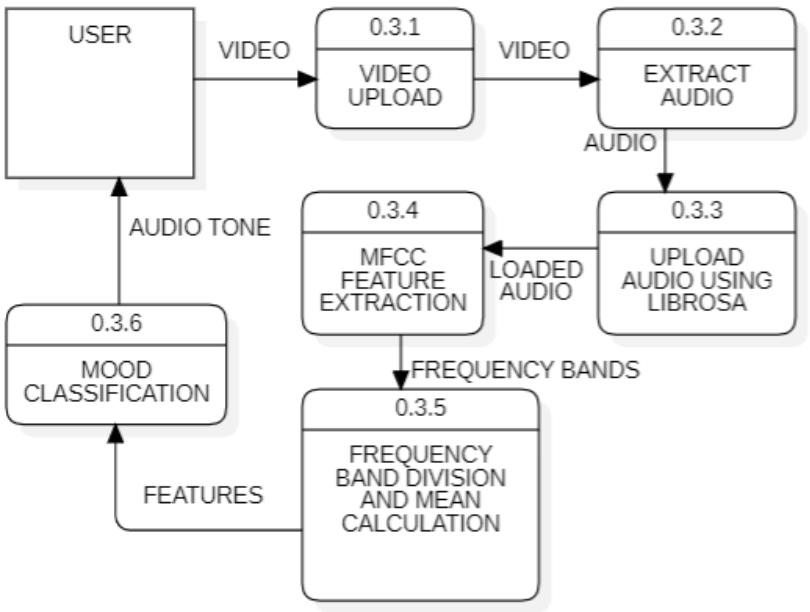


Figure 14: DFD Level 2 of Audio Mood Analysis

The `analyze_audio_mood` function begins with the user providing a video file path. The audio is extracted using the `extract_audio_from_video` function and loaded into memory with the Librosa library. Mel-frequency cepstral coefficients (MFCCs) are computed using `librosa.feature.mfcc` to capture frequency patterns for mood analysis. The MFCC array is segmented into four frequency bands—low, mid-low, mid-high, and high—and the scalar mean of each band is calculated to simplify data for classification. The mood is classified (e.g., normal, calm, anxious) based on the dominant frequency characteristics. Results can be further enhanced using the Gemini API to provide tone, mood, and summary details for the audio. This process combines audio feature extraction and classification for comprehensive mood analysis.

7.9 Prediction to Wellbeing Mapping

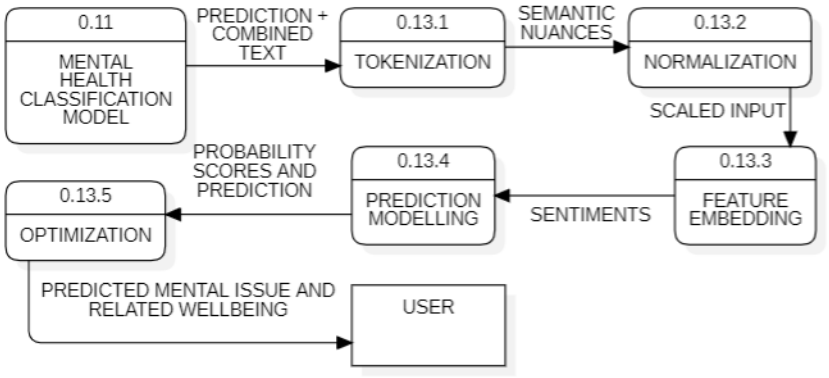


Figure 15: DFD Level 2 of Prediction to Wellbeing Mapping

The system first processes user input (text, behavioral data, or health metrics) through preprocessing and feature extraction. Machine learning models then predict the most likely mental health condition along with associated probabilities. The top predicted issue and its probability are then fed into GEMINI 1.5 FLASH with a structured prompt to generate wellbeing insights based on Ryff's six parameters. To refine these insights, an association matrix maps the probabilities of all predicted issues to specific wellbeing parameters, selecting 1 to 3 key dimensions (e.g., autonomy, personal growth, or self-acceptance). This ensures that the user receives targeted recommendations for improving their psychological wellbeing.

The system enhances mental health classification using ML and DL models, analyzing text, behavioral data, and multimedia inputs. DeepFace aids facial emotion recognition, while text classifiers (Logistic Regression, Naive Bayes, SVM, Transformer, XGBoost, LSTM) process written data. Audio mood analysis (MFCC) and facial expression detection refine emotional assessment. Additionally, real-time social media monitoring provides deeper insights into a user's mental state.

8 Implementation

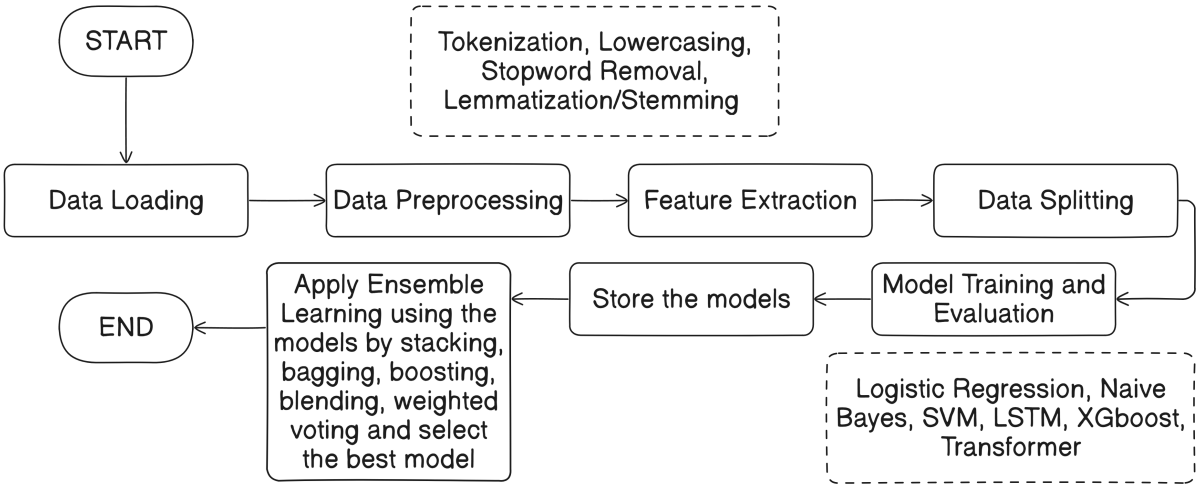


Figure 16: Workflow for getting the model for the web application

8.1 Data Collection and Dataset Preparation

Stepwise Algorithm for Data Collection and Dataset Combination

Step	Description
1	**Import Libraries:** Import `praw`, `pandas`, `time` for data collection and `sklearn.utils.shuffle` for combining datasets.
2	**Initialize Reddit API:** Use the provided credentials (`client_id`, `client_secret`, `user_agent`) to create a Reddit instance.
3	**Define Subreddits and Labels:** Create a dictionary mapping labels to subreddit lists: • **normal**: `news, AskReddit` • **depression**: `depression` • **ptsd**: `PTSD` • **anxiety**: `Anxiety` • **bipolar**: `BipolarReddit`
4	**Set Post Types and Limit:** Define post types (`hot, new, top`) and set `posts_per_type` to 100.
5	**Collect Posts:** • For each label and its associated subreddits, iterate over each post type. • Retrieve posts from the subreddit (using the corresponding post type and a limit of 100). • For each post, combine the title and selftext, and append the result along with its label to a data list. • Pause for 1 second between requests.

Stepwise Algorithm for Data Collection and Dataset Combination

Step	Description
6	**Save Collected Data:** Convert the data list into a DataFrame with columns `text` and `label` and save it as `{label}_dataset.csv`.
7	**Load Datasets:** Read the individual CSV files for `bipolar`, `depression`, `normal`, `anxiety`, and `ptsd` into separate DataFrames.
8	**Determine Minimum Length:** Compute `min_length` as the minimum number of records among datasets (using `len(normal_df)//6` for the normal dataset to balance its count).
9	**Create a Balanced Pattern:** • Loop from 0 to `min_length - 1`. • For each iteration, append to a new list: – The i^{th} record from `bipolar_df`. – The i^{th} record from `depression_df`. – Six consecutive records from `normal_df` (indices `i*6` to `(i+1)*6`). – The i^{th} record from `anxiety_df`. – The i^{th} record from `ptsd_df`.
10	**Convert to DataFrame:** Transform the balanced list into a DataFrame (`pattern_df`).
11	**Prepare Remaining Data:** Concatenate the leftover records from each dataset (beyond `min_length` for all datasets and beyond `min_length*6` for the normal dataset) and shuffle them.
12	**Merge and Save Final Dataset:** Combine `pattern_df` with the shuffled remaining data, reset the index, and save the final DataFrame as `mental_health_combined.csv`.

	text	mental_health_issue
0	2024 Election Due to the 2024 US Presidential ...	bipolar
1	Our most-broken and least-understood rules is ...	depression
2	Rules **UPDATED** 1. If you're here you must p...	normal
3	Happy Cakeday, r/Normal! Today you're 11 Let's...	normal
4	Happy Cakeday, r/Normal! Today you're 10 Let's...	normal
5	I am also a person I am very normal. Today I t...	normal
6	Happy Cakeday, r/Normal! Today you're 9 Let's ...	normal
7	I am a person I am a person	normal
8	Elections and Politics Hello friends!\n\nIt's ...	anxiety
9	You are more than just one emotion	ptsd

Figure 17: Obtained Dataset

The dataset for mental health classification was compiled from subreddit communities, initially containing 385,80 records across five categories: anxiety (54 subreddits), PTSD (38), depression (80), bipolar disorder (60), and normal mental states (53). After data cleaning to ensure quality and relevance, the dataset was reduced to 167,279 records. A subset of 18,596 cleaned records was used for analysis to address computational constraints, such as memory and pro-

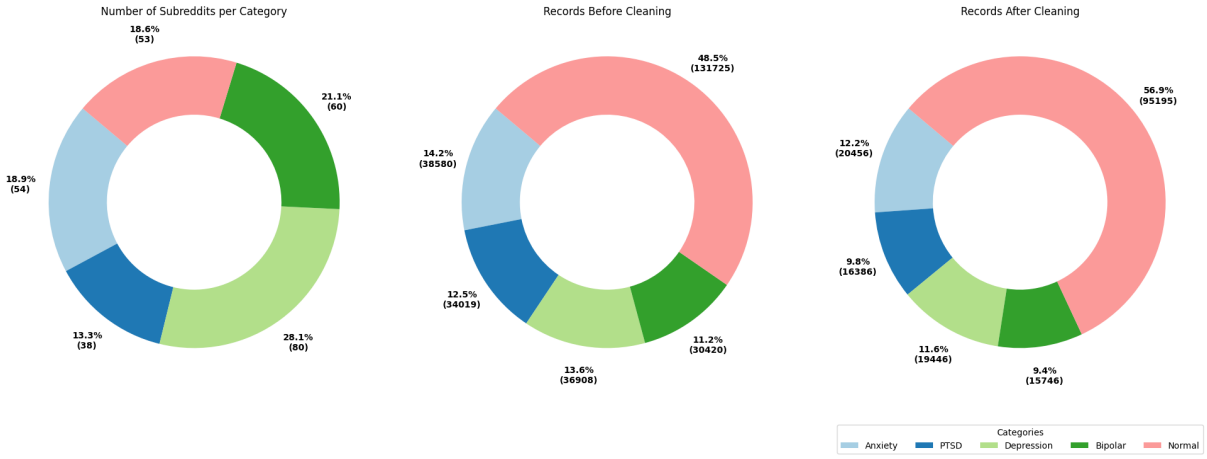


Figure 18: Collected Data Statistics

cessing power, which would make training on the full dataset infeasible on standard hardware. This approach balances efficiency and performance, enabling faster experimentation and iterative model improvement while maintaining a representative sample.

8.2 Data Cleaning and Feature Extraction

Stepwise Algorithm for Data Cleaning and Feature Extraction

Step	Description
1	**Import Libraries & Resources**: Import `pandas`, `re`, `TfidfVectorizer` from scikit-learn, and NLTK modules. Download the necessary NLTK resources (e.g., `stopwords`, `punkt`, and `punkt_tab`).
2	**Load Dataset**: Read the CSV file `mental_health.csv` into a pandas DataFrame.
3	**Handle Missing Values**: Drop any rows where the `text` field is missing.
4	**Remove Duplicates**: Eliminate duplicate rows based on the `text` column.
5	**Define Negative Words**: Create a set of negative words (e.g., `not`, `no`, `nor`, etc.) that will be retained during stopword removal.
6	**Define Cleaning Function**: Write the function `clean_text(text)` to preprocess text by: (a) Removing URLs using regex. (b) Removing mentions (e.g., `@username`). (c) Removing special characters, numbers, and punctuation. (d) Converting text to lowercase. (e) Tokenizing the text using NLTK's `word_tokenize`. (f) Removing stopwords (while keeping negative words). (g) Rejoining tokens into a cleaned string.

Stepwise Algorithm for Data Cleaning and Feature Extraction

Step	Description
7	**Apply Cleaning Function**: Execute `clean_text` on the `text` column and store the result in a new column called `cleaned_text`.
8	**Initialize TF-IDF Vectorizer**: Create a TF-IDF vectorizer instance with a maximum of 10,000 features. (Other methods include Bag-Of-Words, LIWC, N-Gram, Word2Vec)
9	**Fit & Transform Data**: Apply the vectorizer to the `cleaned_text` to generate the TF-IDF feature matrix `X`.
10	**Optional - Convert to DataFrame**: Convert the TF-IDF sparse matrix `X` into a pandas DataFrame for easier inspection (e.g., print the first few rows).
11	**Optional - Save Preprocessed Data**: Save the final preprocessed DataFrame to `preprocessed_mental_health.csv`.

```
[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 19: After TF-IDF Vectorization

Datasets:

Before Cleaning: `mental_health.csv`

After Cleaning: `preprocessed_mental_helth.csv`

Stage	Schema	Description
Before Cleaning	`text`: Original text data `mental_issue`: Mental health issues	Contains raw, unprocessed text data with possible missing values, duplicates, URLs, mentions, and special characters.
After Cleaning	`text`: Original text data `mental_issue`: Mental health issues `cleaned_text`: Processed text data	Data is cleaned by removing URLs, mentions, special characters, and extra noise. The text is converted to lowercase, tokenized, stopwords (except key negative words) are removed, and the cleaned text is stored in the new column `cleaned_text`.

Dataset Schema Before and After Data Cleaning

The matrix dimensions for Bag of Words are determined by the number of records, which is 18,597 in this case, and the size of the vocabulary, which represents the number of unique words in the `cleaned_text` column. Similarly, for TF-IDF, the dimensions of the matrix are the same as BOW, calculated as the number of records multiplied by the vocabulary size. For N-gram, the matrix size depends on the range of n-grams used. For example, a unigram produces dimensions equivalent to BOW, while bigram or trigram models increase the vocabulary size due to the inclusion of word combinations. Word2Vec, on the other hand, creates a dense vector representation for each word, with dimensions based on the predefined vector size, such as 100 or 200. Aggregating these vectors at the sentence level, typically by averaging, results in a matrix of dimensions equal to the number of records multiplied by the vector dimensions. For LIWC, the dimensions are determined by the number of predefined LIWC categories, which is typically around 70. The resulting matrix dimensions are the number of records multiplied by the number of LIWC categories.

8.3 Machine Learning Models

Algorithm like Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Random Forest, XGBoost and Naive Bayes are implemented for the classification of mental health issues. The code algorithm below demonstrates the implementation. The models are evaluated on the test set using metrics like accuracy and classification reports. Hyperparameter Tuning is performed using RandomizedSearchCV to optimize the model performance. Naive Bayes model after hyperparamter tuning is selected along with the basic Logistic Regression, Support Vector Machine, XGboost for the final ensemble model for the web application.

Stepwise Algorithm for Machine Learning Models

Step	Description
1	**Data Loading and Verification:** Load the dataset (`preprocessed_mental_health.csv`) using pandas, verify required columns (`cleaned_text` and `mental_health_issue`), and drop rows with missing values.
2	**Tokenization and Feature Extraction:** Choose a method (Bag-of-Words, TF-IDF, LIWC, Word2Vec, or N-gram) and transform `cleaned_text` into a numerical feature matrix X.
3	**Target Variable Preparation:** Extract the target variable y from the `mental_health_issue` column.
4	**Dataset Splitting:** Split X and y into training and test sets (e.g., 80/20 split) using `train_test_split` with a fixed random state.
5	**Model Initialization:** For each algorithm (Logistic Regression, Naive Bayes, SVM, KNN, Random Forest, XGBoost), initialize the model with appropriate hyperparameters.
6	**Model Training:** Train the selected model on the training data.
7	**Prediction:** Use the trained model to predict mental health issues on the test set.
8	**Model Evaluation:** Evaluate performance by computing accuracy, classification reports, and confusion matrices.
9	**Cross-Validation:** Apply Stratified K-Fold (e.g., 5 folds) cross-validation to record accuracies and calculate the mean and standard deviation.
10	**Performance Comparison:** Compare evaluation metrics across all models and feature extraction methods to select the best combination.
11	**Hyperparameter Tuning:** For Logistic Regression, Naive Bayes, SVM, and KNN, optimize hyperparameters using `RandomizedSearchCV` to search the parameter space and select the optimal configuration.

8.4 Deep Learning Models

Stepwise Algorithm for LSTM-based Model

Step	Description
1	**Data Loading:** Load `preprocessed_mental_health.csv` using pandas.
2	**Feature & Target Preparation:** • Extract text (X) and target (y). • Encode y using `LabelEncoder` and convert to one-hot with `to_categorical`.
3	**Tokenization & Padding:** • Initialize Keras `Tokenizer` with `num_words=25000`. • Fit on X, convert texts to sequences, and pad to `max_length=128` using `pad_sequences`.
4	**Cross-Validation Setup:** Define a 5-fold StratifiedKFold (shuffle=True, random_state=42).

Stepwise Algorithm for LSTM-based Model

Step	Description
5	**LSTM Model Building:** For each fold, build a Keras Sequential model with: • `Embedding(vocab_size, 128, input_length=128)` • `LSTM(128, return_sequences=True)` • `Dropout(0.2)` • `LSTM(64)` • `Dropout(0.2)` • `Dense(64, activation='relu')` • `Dense(num_classes, activation='softmax')`
6	**Model Training:** Compile the model with `optimizer='adamw'` and `loss='categorical_crossentropy'`. Train for 20 epochs with a batch size of 16 using the training fold and validate on the validation fold.
7	**Evaluation per Fold:** • Evaluate the model on the validation set to obtain loss and accuracy. • Predict probabilities on the validation set and compute the confusion matrix.
8	**ROC Curve Calculation:** • Concatenate true labels and predictions from all folds. • Binarize true labels and compute ROC curves and AUC for each class plus a micro-average.
9	**Visualization:** Plot ROC curves, validation loss/accuracy across folds, and the average confusion matrix using Matplotlib and Seaborn.
10	**Final Evaluation:** Compute the average cross-validated accuracy and display a classification report.

```
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 20: Output for LSTM Epochs

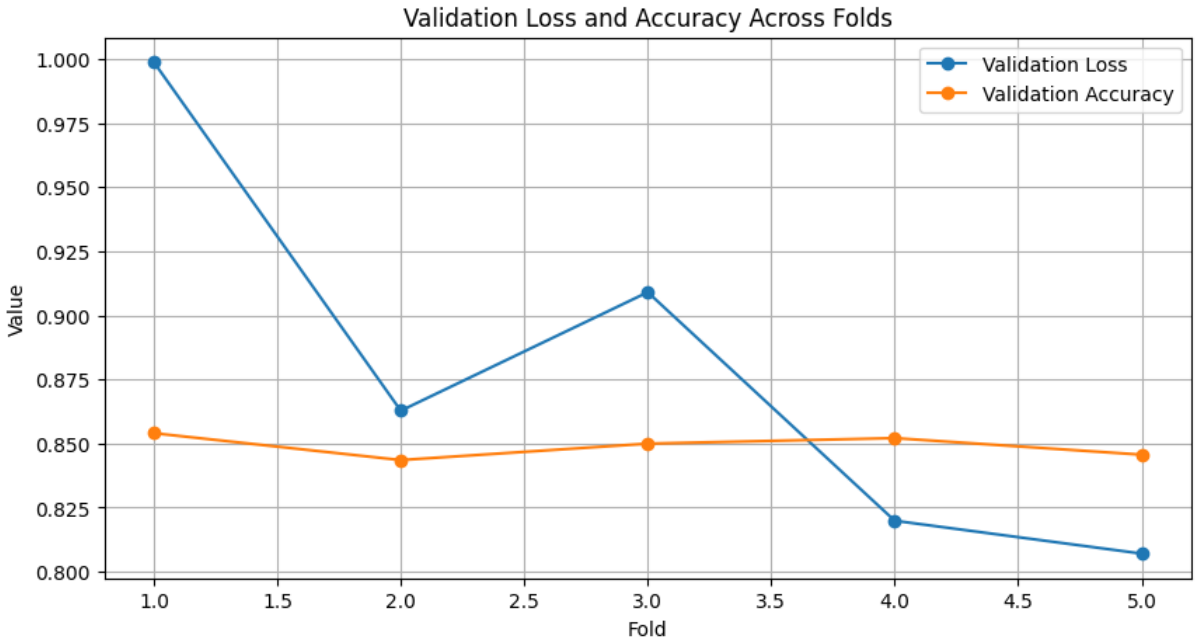


Figure 21: LSTM Validation loss and accuracy

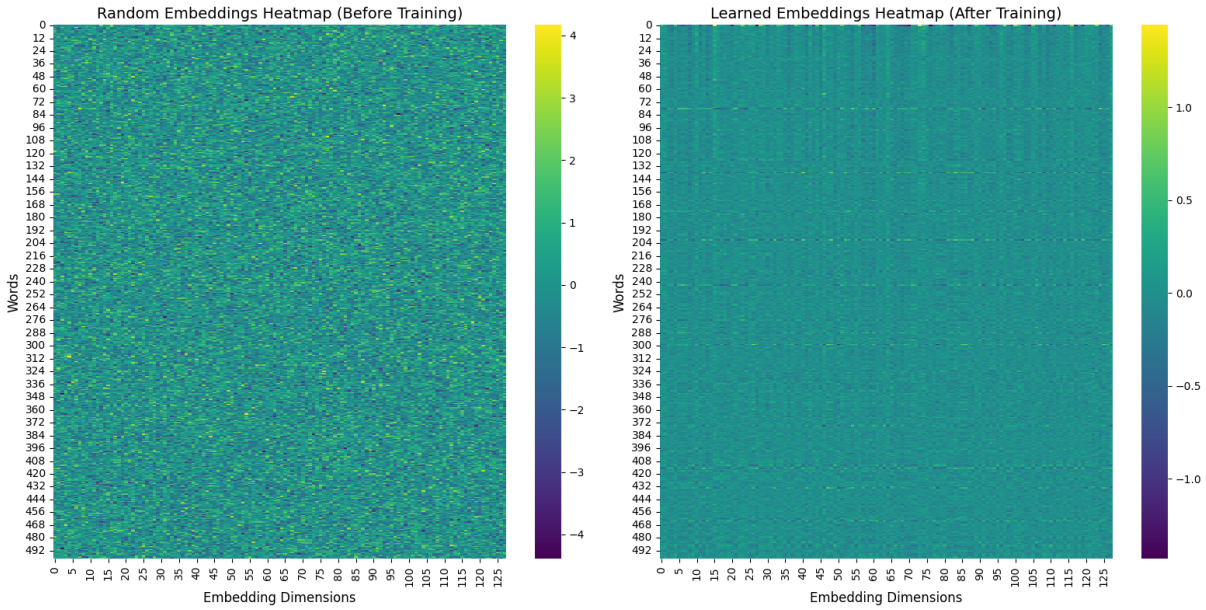


Figure 22: LSTM Random and Learned Embeddings

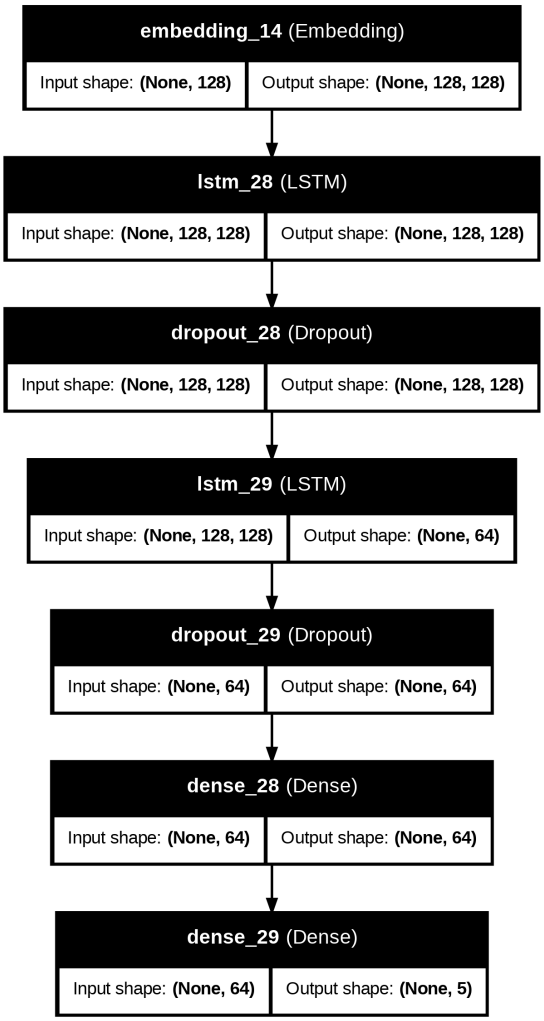


Figure 23: LSTM Model Architecture

Stepwise Algorithm for Custom Transformer-based Model

Step	Description
1	**Import Libraries:** Import required packages including NumPy, pandas, Matplotlib, Seaborn, scikit-learn modules (e.g., `LabelEncoder`, `confusion_matrix`), and TensorFlow Keras modules (e.g., `TextVectorization`, `Embedding`, `LSTM`, `MultiHeadAttention`, etc.).
2	**Load Dataset:** Use Google Colab's file upload to import `preprocessed_mental_health.csv`. Drop rows with missing `cleaned_text` and extract features (`texts`) and labels (`labels`).
3	**Preprocess Labels:** Encode labels with `LabelEncoder`, convert them to one-hot format via `to_categorical`, and retrieve class names.
4	**Text Vectorization:** Set `vocab_size = 25000` and `sequence_length = 300`. Create a `TextVectorization` layer, adapt it on the input texts (using a TensorFlow `Dataset`), and vectorize the texts.

Stepwise Algorithm for Custom Transformer-based Model

Step	Description
5	**Define Custom Transformer Model:** • **EmbeddingLayer:** Custom layer that sums word embeddings with positional embeddings. • **EncoderLayer:** Custom layer using `MultiHeadAttention`, followed by a two-layer Dense network (with ReLU activation) and `LayerNormalization`. • **Model Architecture:** Build the model with an Input layer → `EmbeddingLayer` → `EncoderLayer` → `GlobalAveragePooling1D` → Dense (ReLU) → Output Dense (softmax). • Compile with optimizer `adamw` and loss `categorical_crossentropy`.
6	**Train the Model:** Convert vectorized texts to a NumPy array and split into training and validation sets (80/20 split, `random_state=42`). Train the model for 5 epochs with a batch size of 32.
7	**Evaluate the Model:** • Evaluate validation loss and accuracy. • Predict on the validation set and convert predictions to class labels. • Compute the confusion matrix using scikit-learn's `confusion_matrix`.
8	**Visualization:** Plot the confusion matrix with `ConfusionMatrixDisplay` (using Matplotlib) and optionally visualize random and learned embeddings.

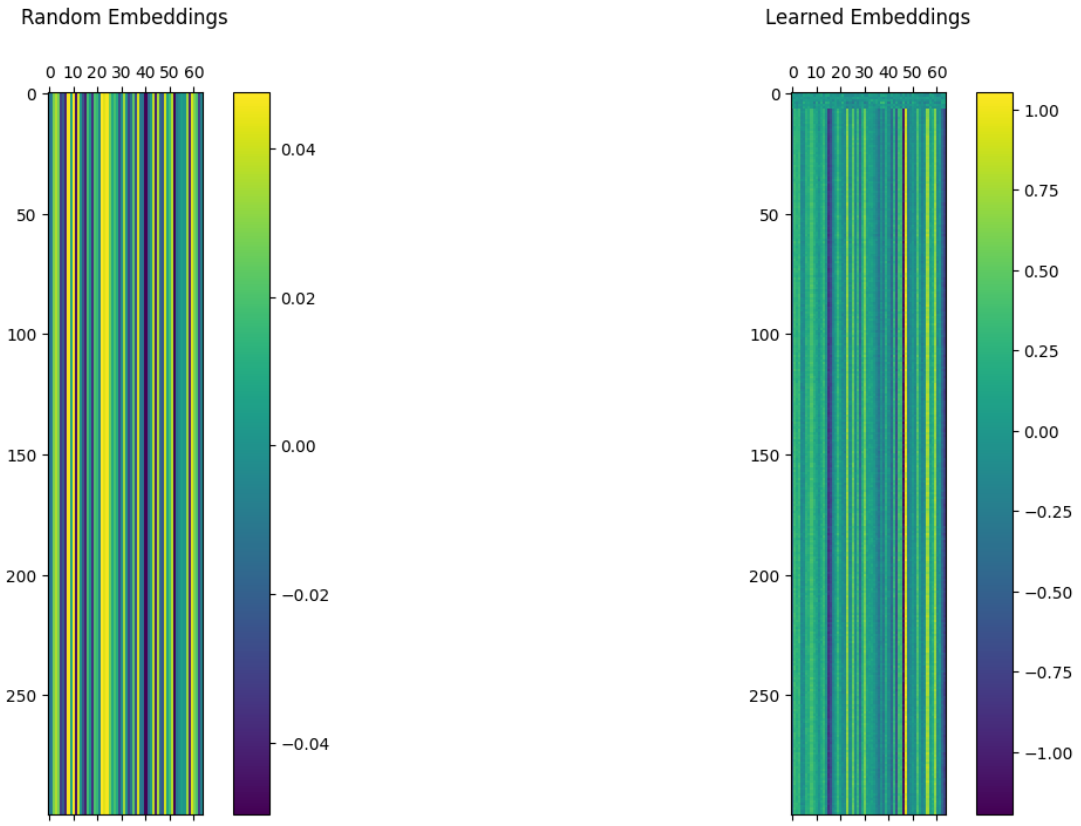


Figure 24: Transformer Model Random and Learned Embeddings

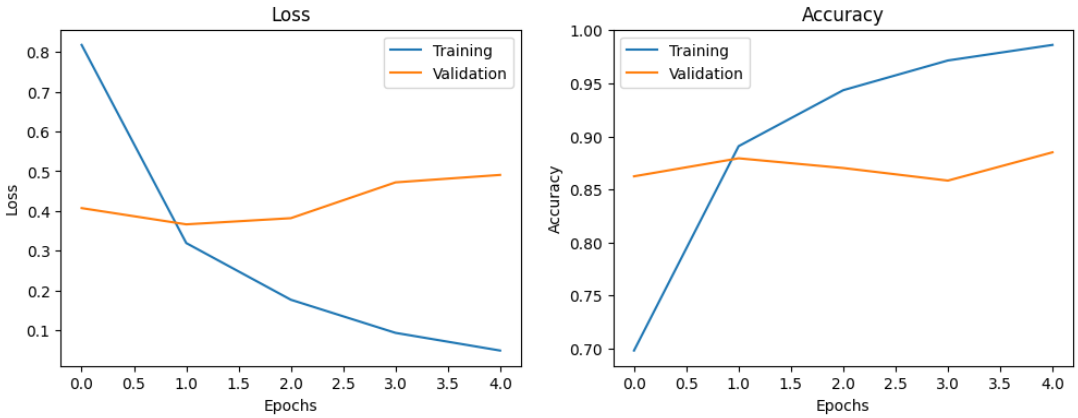


Figure 25: Transformer Epoch, Loss, Accuracy

8.5 Ensemble Model

General Stepwise Algorithm for Ensemble Models

Step	Description
1	**Import Libraries:** Import required packages including `numpy`, `pandas`, `tensorflow` (Keras modules such as `MultiHeadAttention`, `Embedding`, etc.), `scikit-learn` (e.g., `RandomForestClassifier`, `train_test_split`, `confusion_matrix`), and `pickle`.
2	**Load Pre-trained Base Models & Vectorizers:** Load saved models and vectorizers using `pickle` and `load_model`. Base models include: Logistic Regression (LR), Support Vector Machine (SVM), Naive Bayes (NB), XGBoost, LSTM, and Transformer. Also load corresponding vectorizers/tokenizers (e.g., `LRvectorizer`, `SVMvectorizer`, `NBvectorizer`, `tfidf_vectorizer`, `LSTM_tokenizer`, `Tvectorize_layer`) and label encoders.
3	**Load & Preprocess Test Data:** Read `preprocessed_mental_health.csv` with pandas, drop rows with missing `cleaned_text`, extract texts and labels, and encode labels using `LabelEncoder`.
4	**Text Preprocessing for Each Model:** Transform the raw texts using respective vectorizers/-tokenizers: • LR, SVM, NB: Use `LRvectorizer`, `SVMvectorizer`, `NBvectorizer`. • XGBoost: Use `tfidf_vectorizer`. • LSTM: Convert texts using `LSTM_tokenizer` and apply `pad_sequences`. • Transformer: Process texts using `Tvectorize_layer`.
5	**Generate Base Model Predictions:** For each base model, compute prediction probabilities: • LR: `lr_model.predict_proba` • SVM: `svm_model.predict_proba` • NB: `nb_model.predict_proba` • XGBoost: `xgb_model.predict_proba` • LSTM: `lstm_model.predict` • Transformer: `transformer_model.predict`

General Stepwise Algorithm for Ensemble Models

Step	Description
6	**Stack Predictions:** Horizontally stack all base model prediction probabilities to form a new feature matrix (`stacked_features`) for the meta-learner.
7	**Ensemble Configuration & Data Splitting:** Split the stacked features and true labels (using `train_test_split`) into training and testing sets. Specify ensemble configurations: • **Ensemble Model 1:** Base models: LR, XGBoost; Meta-learner: XGBoost. • **Ensemble Model 2:** Base models: LR, NB, SVM, XGBoost, LSTM; Meta-learner: XGBoost. • **Ensemble Model 3:** Base models: LR, NB, SVM, XGBoost, LSTM; Meta-learner: Random Forest. • **Ensemble Model 4:** Same base models using Bagging. • **Ensemble Model 5:** Same base models using Blending. • **Ensemble Model 6:** Same base models using Weighted Voting. • **Ensemble Model 7:** Base models: LR, SVM, NB, LSTM, XGBoost, Transformer; Meta-learner: Random Forest.
8	**Train Meta-Learner:** Train the meta-learner (e.g., Random Forest, XGBoost, or ensemble strategies like Bagging, Blending, Voting) on the training portion of the stacked feature matrix.
9	**Evaluate Ensemble Model:** Predict on the test set using the meta-learner and compute evaluation metrics: accuracy, classification report, and confusion matrix. Also, perform cross-validation (e.g., using `cross_val_score`) to assess stability.
10	**Output Results:** Print final evaluation metrics (accuracy, report, confusion matrix) and cross-validation results.

9 Test Plans, Results and Analysis

Test Case Plan Table

Test Case ID	Description	Expected Result	Actual Result	Status
T01	User provides input in the text area.	Text is accepted for further processing.	Text is accepted for further processing.	✓
T02	Display mental health issues with probabilities for classes.	Probabilities for each mental health class are shown.	Probabilities for each mental health class are shown.	✓
T03	Highlight the mental health issue with the highest probability.	Correct issue with the highest probability is displayed.	Correct issue with the highest probability is displayed.	✓
T04	Accept username input and provide prediction.	Prediction is displayed based on the provided username.	Prediction is displayed based on the provided username.	✓

Test Case Plan Table

Test Case ID	Description	Expected Result	Actual Result	Status
T05	Translate multiple language inputs to English.	Non-English text is translated correctly to English.	Non-English text is translated correctly to English.	✓
T06	Extract text and detect emotion from image.	Extracted text and detected emotions are displayed accurately.	Extracted text and detected emotions are displayed accurately.	✓
T07	Pass a prompt to the system and retrieve a valid response.	Correct response is generated based on the prompt.	Correct response is generated based on the prompt.	✓
T08	Perform a combined test using multiple inputs.	Predictions for all inputs are displayed correctly.	Predictions for all inputs are displayed correctly.	✓
T09	Analyze uploaded audio files and transcribe them into text.	Audio transcription and analysis results are displayed.	Audio transcription and analysis results are displayed.	✓
T10	Extract frames, analyze emotions, audio from video.	Frame emotions and audio transcription are displayed.	Frame emotions and audio transcription are displayed.	✓
T11	Extract tweets and related media using a Twitter username.	Text and media are extracted and analyzed correctly.	Text and media are extracted and analyzed correctly.	✓
T12	Generate captions for uploaded images or video frames.	Captions are generated for images or frames.	Captions are generated for images or frames.	✓
T13	Upload a PDF file and analyze its text content.	Extracted text from the PDF is processed and analyzed for mental health cues.	Extracted text from the PDF is processed and analyzed for mental health cues.	✓
T14	Capture user response to a displayed image.	User response is captured and correctly classified for emotional analysis.	User response is captured and correctly classified for emotional analysis.	✓
T15	Fill out a survey form to update the association matrix.	Survey responses update the association matrix and generate targeted wellbeing insights.	Survey responses update the association matrix and generate targeted wellbeing insights.	✓

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

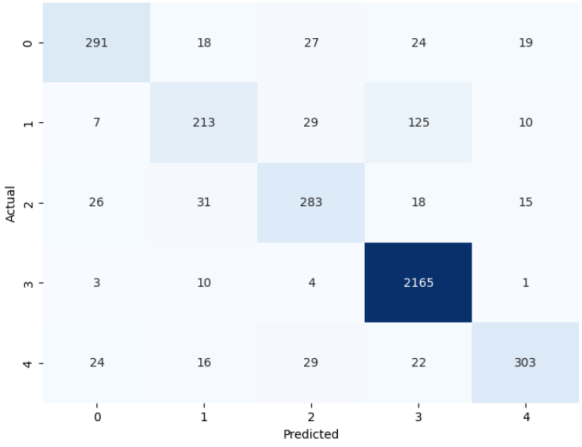
9.2 Results of Logistic Regression

Logistic Regression Classification Report

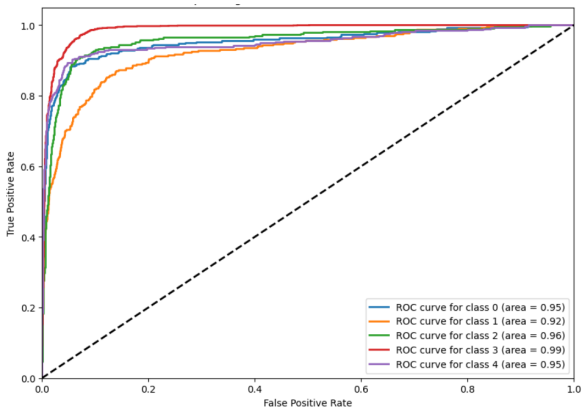
Class	Precision	Recall	F1-Score	Support
Anxiety	0.83	0.77	0.80	379
Bipolar	0.74	0.55	0.63	384
Depression	0.76	0.76	0.76	373
Normal	0.92	0.99	0.95	2183
PTSD	0.87	0.77	0.82	394
Accuracy	87.66%			
Macro Avg	0.82	0.77	0.79	3713
Weighted Avg	0.87	0.88	0.87	3713

ROC Curve Areas for Each Class

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



(a) Confusion Matrix (Logistic Regression)



(b) ROC AUC (Logistic Regression)

Figure 26: Evaluation Results for Logistic Regression

9.3 Results of Naive Bayes

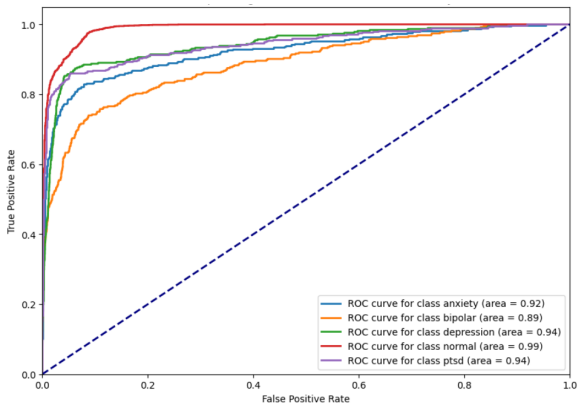
Naive Bayes Classification Report

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

ROC Curve Areas for Each Class

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

(a) Confusion Matrix (Naive Bayes)



(b) ROC AUC (Naive Bayes)

Figure 27: Evaluation Results for Naive Bayes

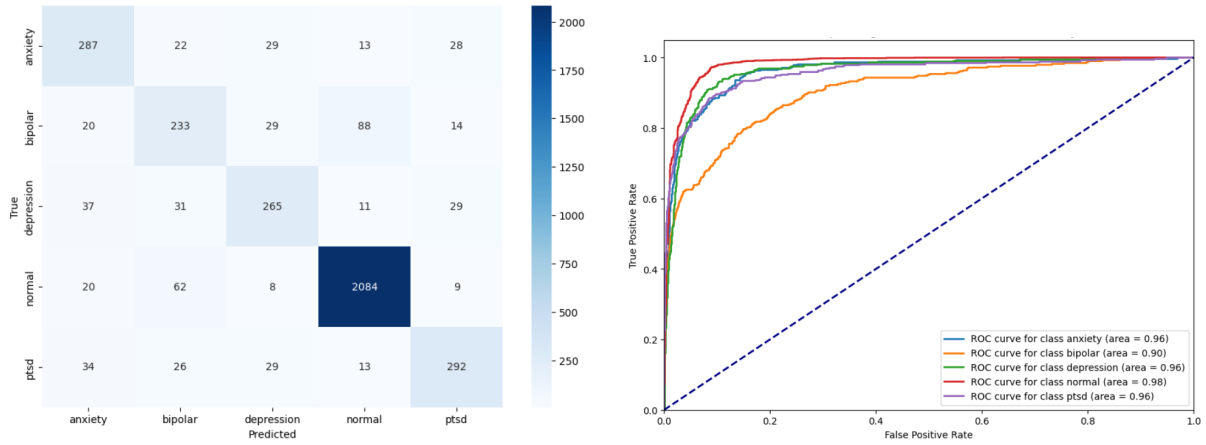
9.4 Results of Support Vector Machine

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

ROC Curve Areas for Each Class

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



(a) Confusion Matrix (SVM)

(b) ROC AUC (SVM)

Figure 28: Evaluation Results for SVM

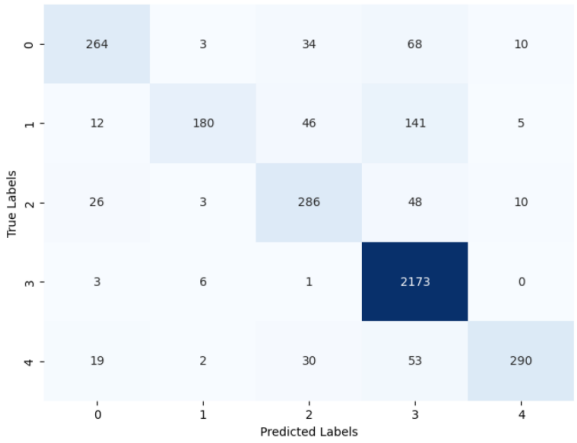
9.5 Results of Random Forest

Random Forest Classification Report

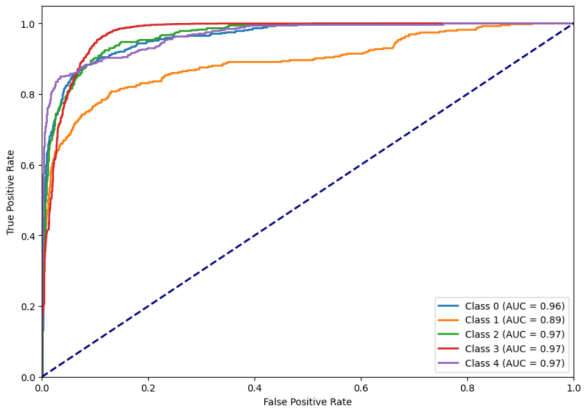
Class	Precision	Recall	F1-Score	Support
Anxiety	0.81	0.70	0.75	379
Bipolar	0.93	0.47	0.62	384
Depression	0.72	0.77	0.74	373
Normal	0.88	1.00	0.93	2183
PTSD	0.92	0.74	0.82	394
Accuracy	86.00%			
Macro Avg	0.85	0.73	0.77	3713
Weighted Avg	0.86	0.86	0.85	3713

ROC Curve Areas for Each Class

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



(a) Confusion Matrix (Random Forest)



(b) ROC AUC (Random Forest)

Figure 29: Evaluation Results for Random Forest

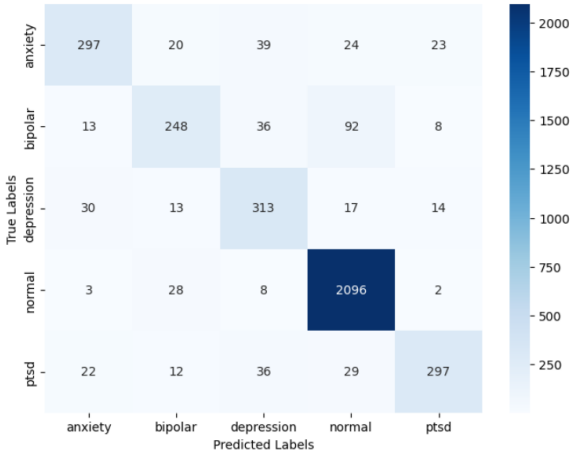
9.6 Results of XGBoost

XGBoost Classification Report

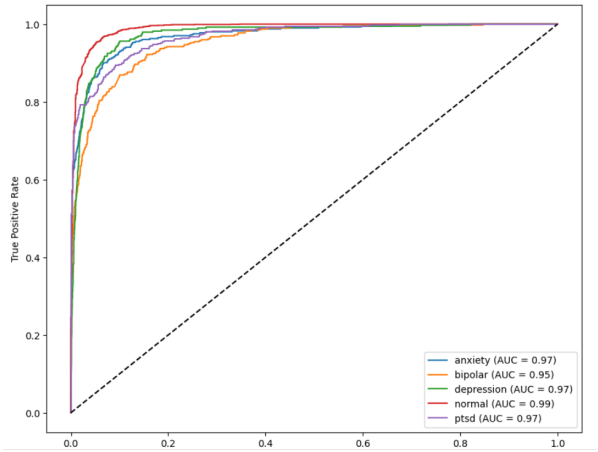
Class	Precision	Recall	F1-Score	Support
Anxiety	0.81	0.74	0.77	403
Bipolar	0.77	0.62	0.69	397
Depression	0.72	0.81	0.76	387
Normal	0.93	0.98	0.95	2137
PTSD	0.86	0.75	0.80	396
Accuracy	87.39%			
Macro Avg	0.82	0.78	0.80	3720
Weighted Avg	0.87	0.87	0.87	3720

ROC Curve Areas for Each Class

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



(a) Confusion Matrix (XGBoost)



(b) ROC AUC (XGBoost)

Figure 30: Evaluation Results for XGBoost

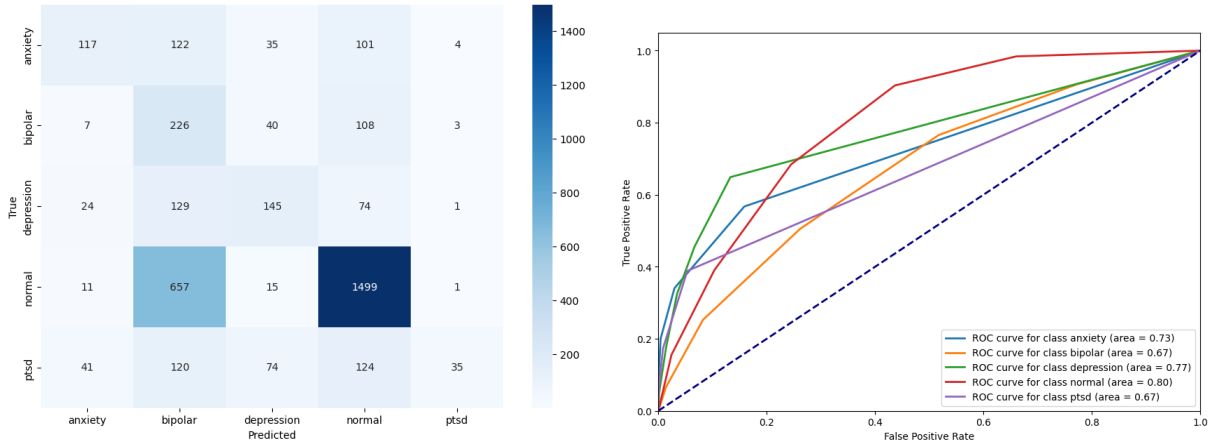
9.7 Results of KNN

KNN Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.58	0.31	0.40	379
Bipolar	0.18	0.59	0.28	384
Depression	0.47	0.39	0.43	373
Normal	0.79	0.69	0.73	2183
PTSD	0.80	0.09	0.16	394
Accuracy	54.46%			
Macro Avg	0.56	0.41	0.40	3713
Weighted Avg	0.67	0.54	0.56	3713

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.73
Bipolar	0.67
Depression	0.77
Normal	0.80
PTSD	0.67



(a) Confusion Matrix (KNN)

(b) ROC AUC (KNN)

Figure 31: Evaluation Results for KNN

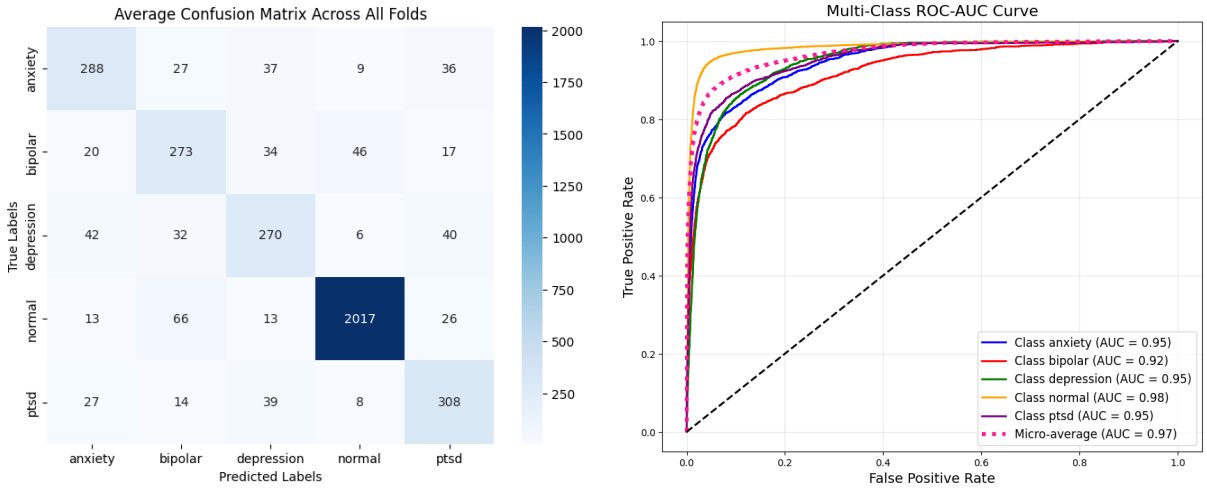
9.8 Results of LSTM

LSTM Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.74	0.72	0.73	1999
Bipolar	0.66	0.70	0.68	1964
Depression	0.68	0.69	0.69	1959
Normal	0.97	0.94	0.95	10688
PTSD	0.72	0.78	0.75	1987
Accuracy	84.91%			
Macro Avg	0.75	0.77	0.76	18597
Weighted Avg	0.85	0.85	0.85	18597

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.95
Bipolar	0.92
Depression	0.95
Normal	0.98
PTSD	0.95



(a) Confusion Matrix (LSTM)

(b) ROC AUC (LSTM)

Figure 32: Evaluation Results for LSTM

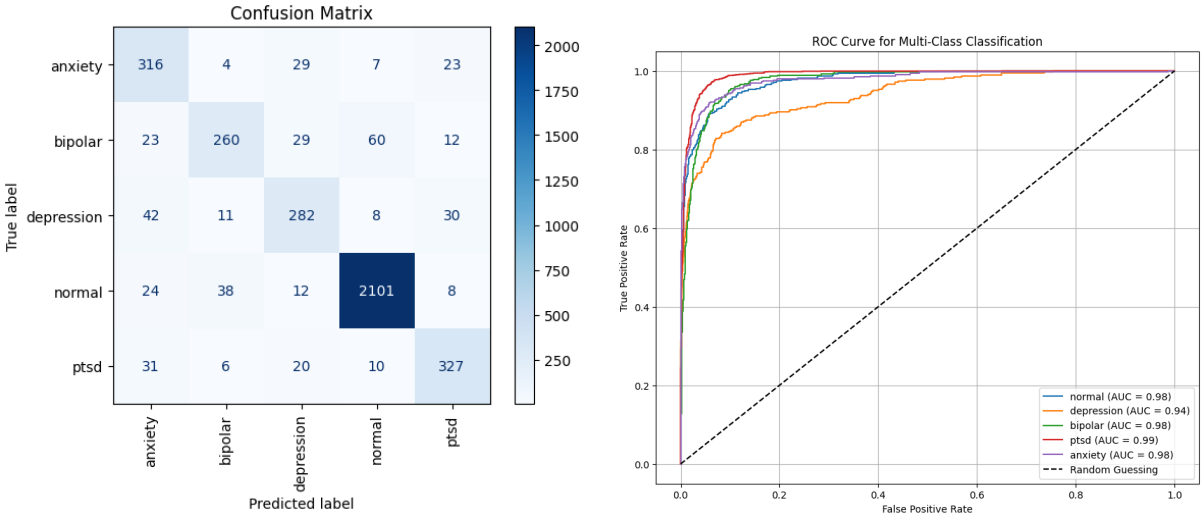
9.9 Results from Transformer based model

Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.75	0.78	0.76	379
Bipolar	0.76	0.69	0.73	384
Depression	0.82	0.72	0.77	373
Normal	0.95	0.97	0.96	2183
PTSD	0.79	0.80	0.79	394
Accuracy	88.5%			
Macro Avg	0.81	0.79	0.80	3713
Weighted Avg	0.88	0.88	0.88	3713

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.97
Bipolar	0.94
Depression	0.98
Normal	0.99
PTSD	0.97



(a) Confusion Matrix for Transformer based Model (b) ROC AUC for Transformer based Model

Figure 33: Evaluation Results for Transformer based Model

Model	**Accuracy (%)**	**Key Observations**
Logistic Regression	87.66	High precision, recall, and F1 for the *Normal* class; struggles with *Bipolar* and *Anxiety*. ROC AUC: Normal (0.99), Anxiety (0.95), Depression (0.96).
Naive Bayes	83.63	Excellent for *Normal* (Precision 0.96, Recall 0.92, F1 0.94) but poor for *Bipolar* (Recall 0.45, F1 0.58). ROC AUC: Normal (0.99), Depression/PTSD (0.94), Bipolar (0.89).
SVM	85.13	Best for *Normal* (Precision 0.94, Recall 0.95, F1 0.95); *Bipolar* underperforms (Precision 0.62, Recall 0.61); *Anxiety* is balanced (Recall 0.76, Precision 0.72). ROC AUC: Normal (0.98), others 0.96, Bipolar (0.90).
Random Forest	86.00	*Normal* class achieves perfect recall (1.00) with high precision (0.88, F1 0.93); *Bipolar* is weak (Recall 0.47, F1 0.62). ROC AUC: Most classes ¿0.96; Bipolar (0.89).
XGBoost	87.39	Strong performance for *Normal*; *Anxiety* shows good metrics (Precision 0.81, Recall 0.74) while *Bipolar* has lower recall (0.62). ROC AUC: 0.97 for Anxiety, Depression, PTSD; 0.99 for Normal.
KNN	54.46	Overall low performance; *Normal* is moderate (Precision 0.79, Recall 0.69) but *PTSD* has very low recall (0.09). ROC AUC: Normal (0.80), indicating weak discrimination.
LSTM	84.91	High performance for *Normal*; *Anxiety* acceptable (Precision 0.74, Recall 0.72); *Bipolar* (Precision 0.66, Recall 0.70) and *PTSD* (Precision 0.72, Recall 0.78) are moderate. Effective ROC AUC for Normal.
Transformer	88.50	Best overall with balanced high precision and recall across all classes. Captures contextual cues effectively and handles class imbalance well. ROC AUC scores range between 0.94 and 0.99.

Summary Comparison of Classification Models without Hyperparameter Tuning

9.10 Results of Hyperparameter Tuning

Logistic Regression

Hyperparameter Tuning on Logistic Regression

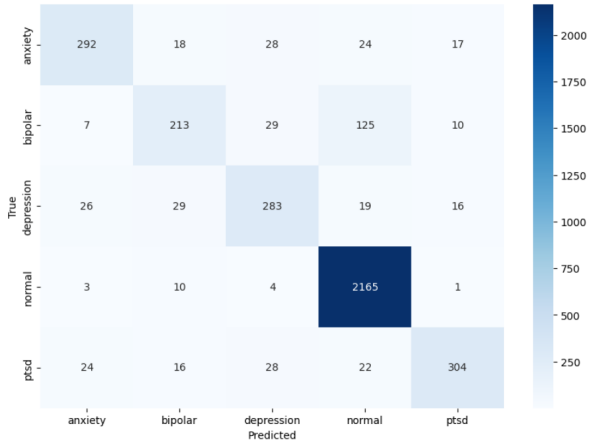
Best Hyperparameters	Value
Solver	liblinear
Penalty	l2
C	1

Classification Report

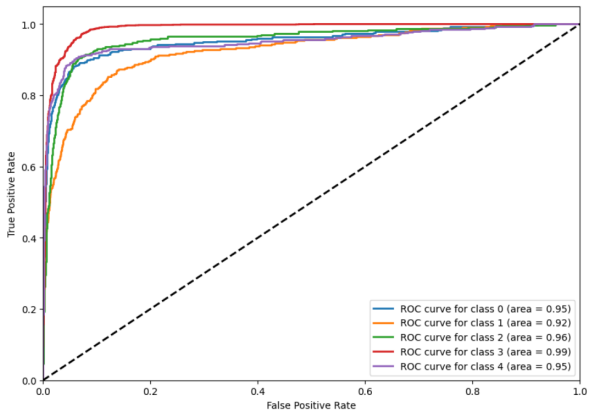
Class	Precision	Recall	F1-Score	Support
Anxiety	0.83	0.77	0.80	379
Bipolar	0.74	0.55	0.64	384
Depression	0.76	0.76	0.76	373
Normal	0.92	0.99	0.95	2183
PTSD	0.87	0.77	0.82	394
Accuracy	87.72%			
Macro Avg	0.83	0.77	0.79	3713
Weighted Avg	0.87	0.88	0.87	3713

ROC Curve Areas for Each Class

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



(a) Classification Matrix after Hyperparameter Tuning (Logistic Regression)



(b) ROC AUC after Hyperparameter Tuning (Logistic Regression)

Figure 34: Evaluation Results after Hyperparameter Tuning (Logistic Regression)

K-Nearest Neighbours

Hyperparameter Tuning on k-NN

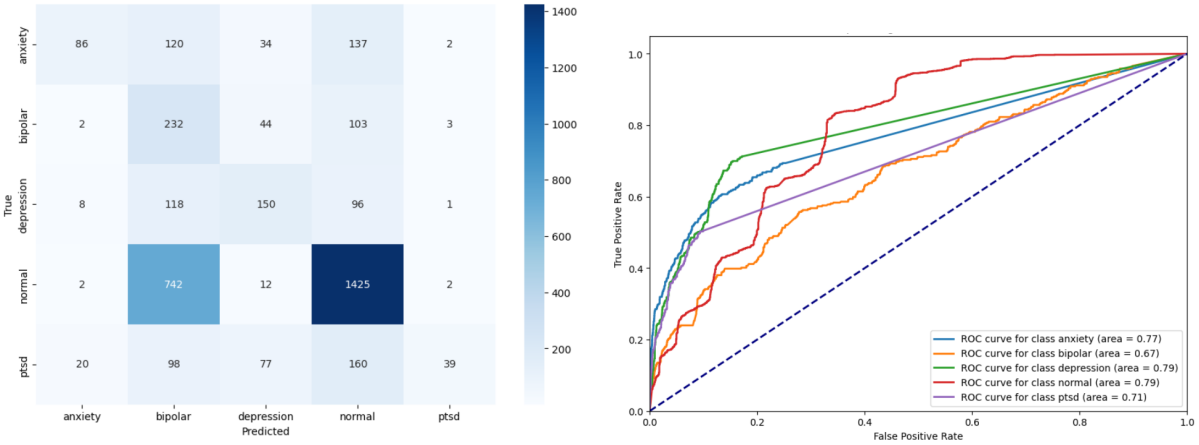
Best Hyperparameters	Value
Weights	distance
n_neighbors	10
Metric	euclidean

Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.73	0.23	0.35	379
Bipolar	0.18	0.60	0.27	384
Depression	0.47	0.40	0.43	373
Normal	0.74	0.65	0.69	2183
PTSD	0.83	0.10	0.18	394
Accuracy	52.03%			
Macro Avg	0.59	0.40	0.39	3713
Weighted Avg	0.66	0.52	0.53	3713

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.77
Bipolar	0.67
Depression	0.79
Normal	0.79
PTSD	0.71



(a) Classification Matrix after Hyperparameter Tuning (KNN)

(b) ROC AUC after Hyperparameter Tuning (KNN)

Figure 35: Evaluation Results after Hyperparameter Tuning (KNN)

Support Vector Machine

Hyperparameter Tuning on SVM

Best Hyperparameters	Value
Kernel	linear
Gamma	scale
C	1

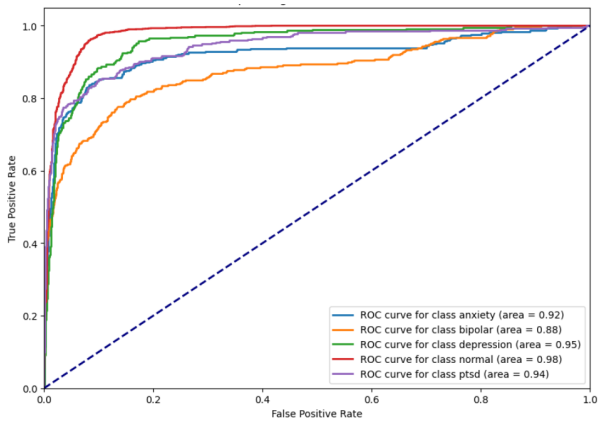
Classification Report

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

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.92
Bipolar	0.88
Depression	0.95
Normal	0.98
PTSD	0.94

(a) Classification Matrix after Hyperparameter Tuning (SVM)



(b) ROC AUC after Hyperparameter Tuning (SVM)

Figure 36: Evaluation Results after Hyperparameter Tuning (SVM)

Naive Bayes

Hyperparameter Tuning on Naive Bayes

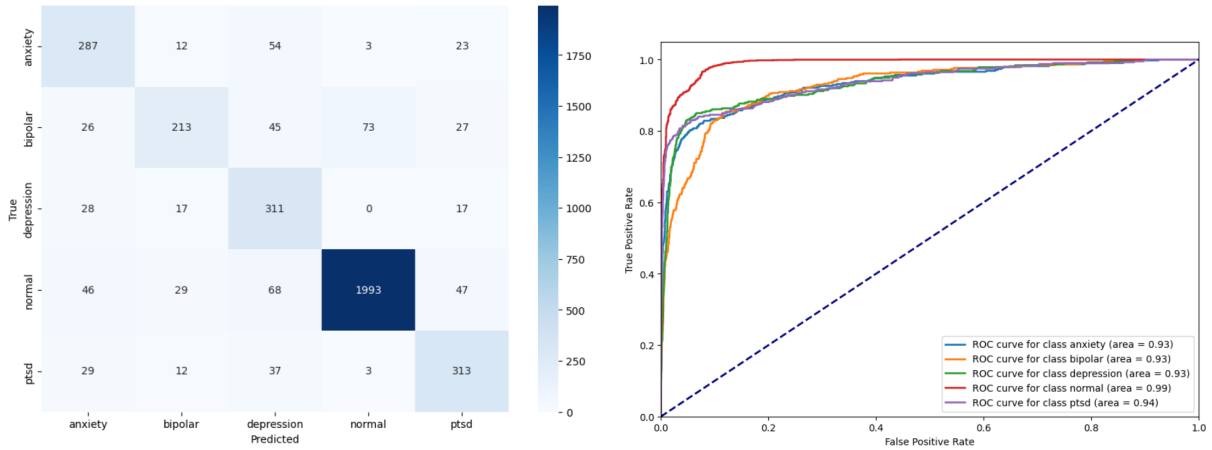
Best Hyperparameters	Value
Alpha	0.2914180608409973

Classification Report

Class	Precision	Recall	F1-Score	Support
Anxiety	0.69	0.76	0.72	379
Bipolar	0.75	0.55	0.64	384
Depression	0.60	0.83	0.70	373
Normal	0.96	0.91	0.94	2183
PTSD	0.73	0.79	0.76	394
Accuracy	83.95%			
Macro Avg	0.75	0.77	0.75	3713
Weighted Avg	0.85	0.84	0.84	3713

ROC Curve Areas for Each Class

Class	ROC AUC
Anxiety	0.93
Bipolar	0.93
Depression	0.93
Normal	0.99
PTSD	0.94



(a) Classification Matrix after Hyperparameter Tuning (Naive Bayes)

(b) ROC AUC after Hyperparameter Tuning (Naive Bayes)

Figure 37: Evaluation Results after Hyperparameter Tuning (Naive Bayes)

Model	Best Hyperparameters	Accuracy & Metrics	Key Observations
Logistic Regression	`solver=liblinear, penalty=l2, C=1`	Accuracy: 87.72%; Normal: 92% accuracy; ROC AUC: Normal 0.99, Depression 0.96	High precision, recall, and F1 for Normal; Bipolar shows lower F1 (64%); overall robust performance, especially for Normal and Anxiety.
k-NN	`weights=distance, n_neighbors=10, metric=euclidean`	Accuracy: 52.03%; Normal F1: 0.69; ROC AUC: Bipolar 0.67, PTSD 0.71, Normal/Depression 0.79	Struggles with minority classes: Anxiety (recall 0.23) and PTSD (recall 0.10); not suitable for this dataset.
SVM	`kernel=linear, gamma=scale, C=1`	Accuracy: 85.13%; F1-scores: Normal 0.95, Depression 0.72, Anxiety 0.74, Bipolar 0.61; ROC AUC: Normal 0.98, Others ¿0.90, Bipolar 0.90	Solid overall performance; excellent for Normal and Anxiety, but slightly lower performance for Bipolar.
Naive Bayes	`alpha=0.2914`	Accuracy: 83.95%; F1-scores: Normal 0.94, PTSD 0.76, Anxiety 0.72; ROC AUC: Normal 0.99, Others ¿0.90	Performs very well for Normal and PTSD; lower scores for Bipolar and Depression.

Summary Comparison of Models after Hyperparameter Tuning

K-Nearest Neighbors (KNN) performs poorly on this mental health dataset compared to other algorithms, even after hyperparameter tuning. KNN relies on distance-based metrics, which struggle with the sparse, high-dimensional nature of text data, making it difficult to capture meaningful relationships between Reddit posts and mental health labels. Despite optimizing parameters like `weights='distance'`, `n_neighbors=10`, and `metric='euclidean'`, KNN's performance remains suboptimal, with lower accuracy, precision, recall, and F1-score than Logistic Regression, Naive Bayes, or SVM. Its inability to handle non-linear, context-dependent patterns in text, especially for nuanced categories like anxiety, PTSD, or bipolar disorder, highlights its limitations in text classification tasks. KNN's reliance on proximity without considering textual context likely explains its poor performance.

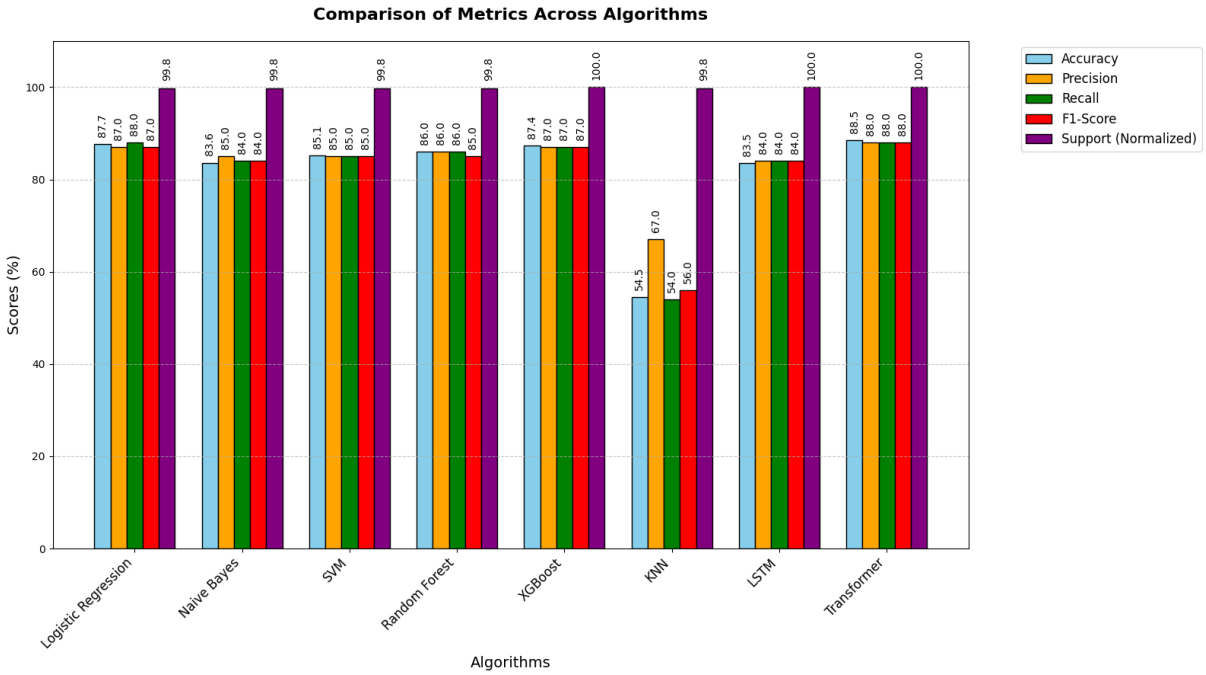


Figure 38: Result Comparison of the Algorithms

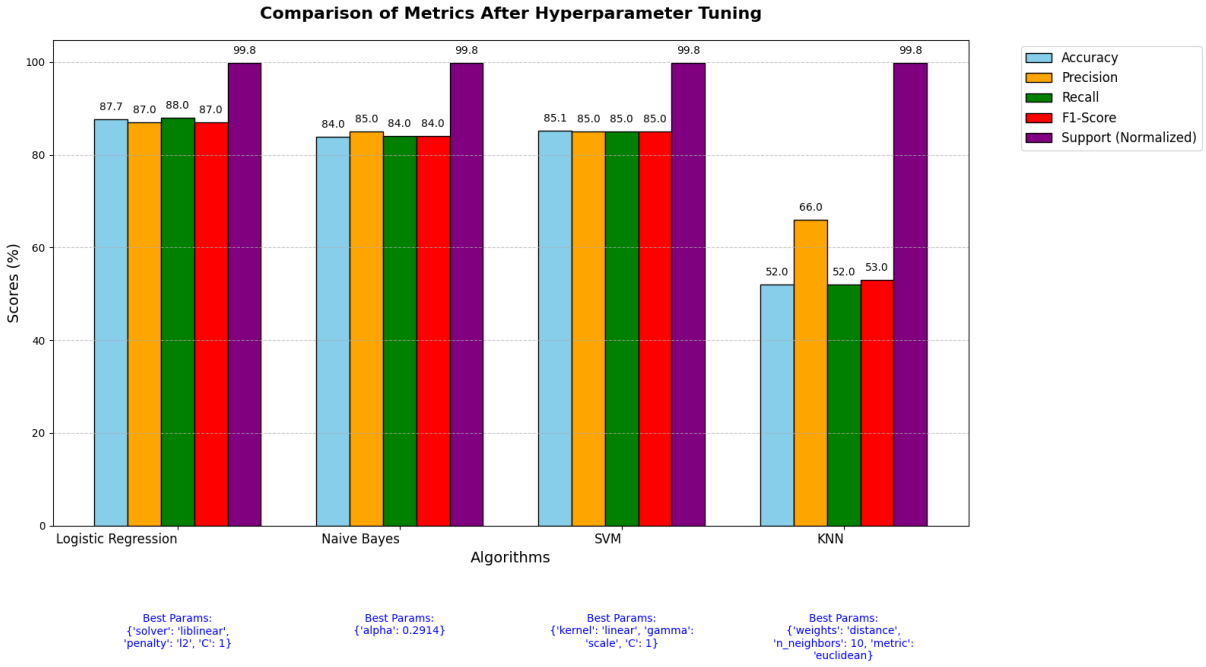


Figure 39: Result Comparison after Hyperparameter Tuning

9.11 Comparison of different tokenizations

Model Performance Comparison

Model	BoW	TFIDF	LIWC	Word2Vec	N-Gram
Logistic Regression	87.66	86.02	66.36	79.40	87.13
KNN	54.56	75.06	70.70	75.52	43.06
SVC	85.13	88.26	68.14	77.89	84.33
Naive Bayes	83.63	79.42	59.63	60.44	81.71
Random Forest	85.32	85.73	76.73	79.88	79.02
XGB	87.42	87.39	78.56	81.01	87.96

In ensemble learning, combining models like Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), XGBoost (XGB), Random Forest (RF), and Long Short-Term Memory (LSTM) poses challenges in computational efficiency, model size, and accuracy. Each model has unique strengths and limitations, making their integration complex. LR, NB, and SVM perform well with simpler vectorization methods like Bag of Words (BoW), which efficiently represents text data as sparse vectors. BoW's simplicity ensures these models remain computationally lightweight while maintaining good classification accuracy. In contrast, XGBoost benefits from more complex feature extraction methods like TFIDF (Term Frequency-Inverse Document Frequency), which captures nuanced relationships by weighting words based on their importance across documents. However, combining TFIDF-based models like XGBoost with BoW-based models can degrade ensemble performance, as the feature representations may not align well. KNN is excluded due to its computational intensity, as it requires storing the entire training dataset and performs poorly with large, high-dimensional datasets. Similarly, Random Forest is omitted because its multiple decision trees increase model size and inference time, outweighing its individual performance benefits. XGBoost, while highly accurate with N-Gram features, is computationally expensive and impractical for real-time systems or large-scale deployment. The choice of feature extraction methods—BoW for LR, NB, and SVM, and TFIDF for XGBoost—balances simplicity and complexity. BoW ensures efficient, interpretable representations for simpler models, while TFIDF provides richer, context-aware features for XGBoost. This combination leverages the strengths of both methods, ensuring each model operates effectively without excessive computational strain. Ultimately, this approach strikes a balance between accuracy and efficiency, enabling the ensemble to perform well while remaining scalable and practical for real-world applications.

9.12 Results from Ensemble Model Training and Testing

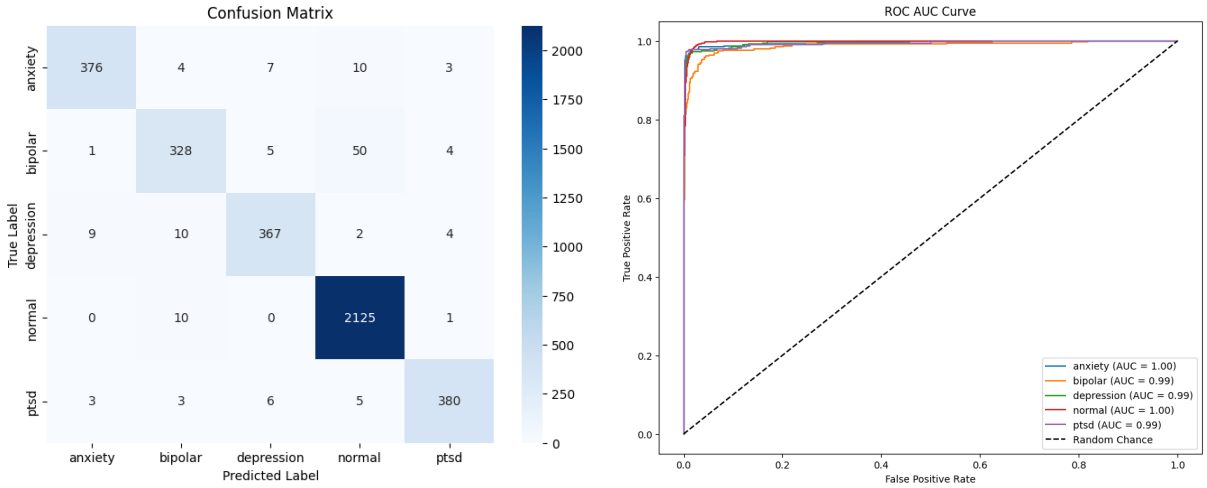
Ensemble Model 1

Cross-Validation Accuracy Scores

Cross-Validation Accuracy Scores	Value
1	0.96364126
2	0.95744681
3	0.96175599
4	0.95933208
5	0.96202532
Mean Validation Accuracy	**96.08%**

Classification Report for Ensemble Model1

	Precision	Recall	F1-Score	Support
Anxiety	0.97	0.94	0.95	400
Bipolar	0.92	0.85	0.88	388
Depression	0.95	0.94	0.94	392
Normal	0.97	0.99	0.98	2136
PTSD	0.97	0.96	0.96	397
Accuracy	**96.08%**			
Macro avg	0.96	0.93	0.95	3713
Weighted avg	0.96	0.96	0.96	3713



(a) Confusion Matrix for Ensemble Model 1 (b) ROC AUC Curve for Ensemble Model 1

Figure 40: Evaluation Results for Ensemble Model 1

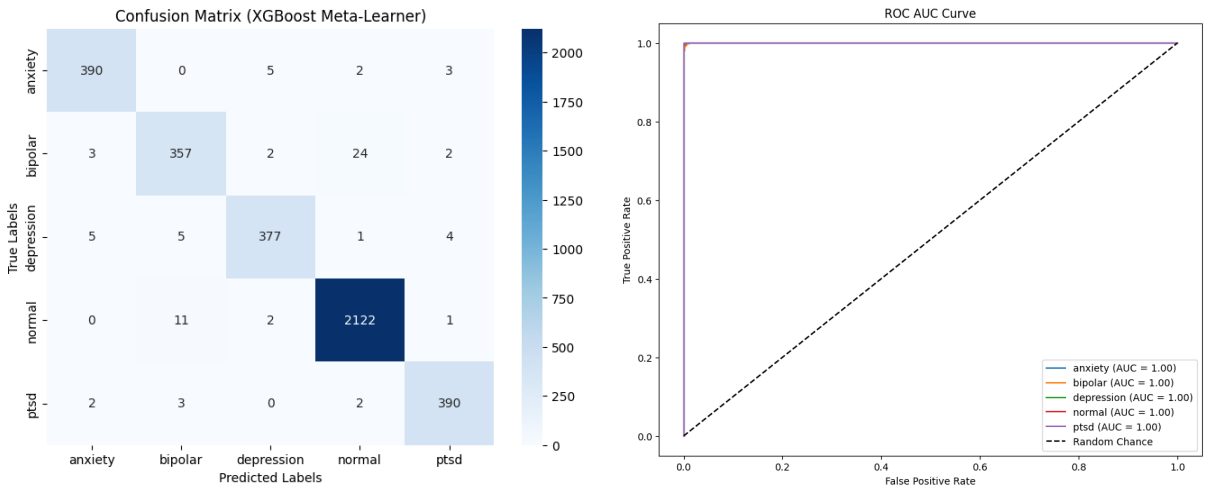
Ensemble Model 2

Cross-Validation Accuracy Scores

Cross-Validation Accuracy Scores	Value
1	0.980878
2	0.97629949
3	0.98249394
4	0.97522219
5	0.97710746
Mean Validation Accuracy	**97.84%**

Classification Report for Ensemble Model2

	Precision	Recall	F1-Score	Support
Anxiety	0.97	0.97	0.97	400
Bipolar	0.95	0.92	0.93	388
Depression	0.98	0.96	0.97	392
Normal	0.99	0.99	0.99	2136
PTSD	0.97	0.98	0.98	397
Accuracy	**97.93%**			
Macro avg	0.97	0.97	0.97	3713
Weighted avg	0.98	0.98	0.98	3713



(a) Confusion Matrix for Ensemble Model 2 (b) ROC AUC for Ensemble Model 2

Figure 41: Evaluation Results for Ensemble Model 2

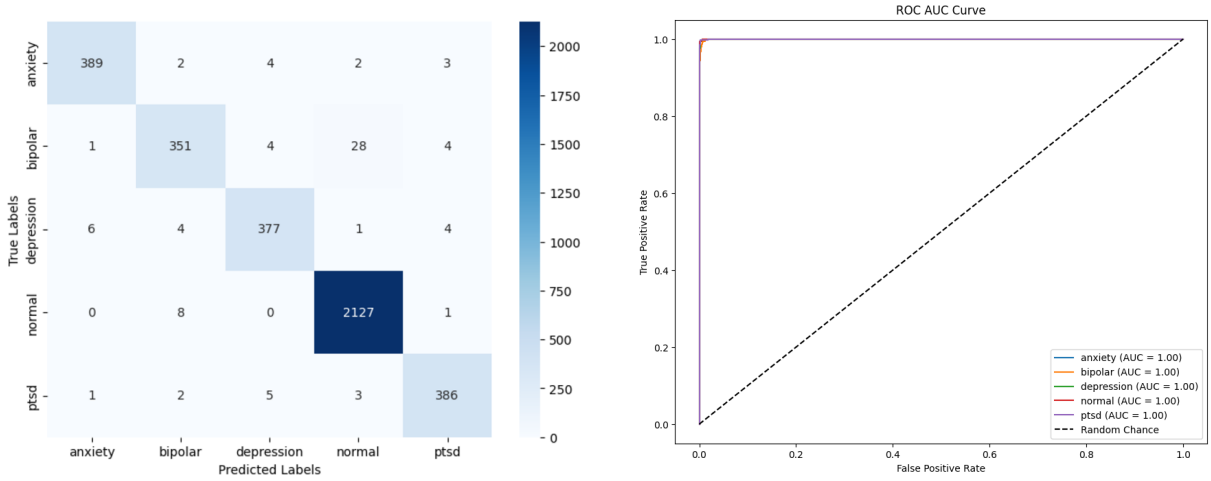
Ensemble Model 3

Cross-Validation Accuracy Scores

Cross-Validation Accuracy Scores	Value
1	0.98195529
2	0.97549152
3	0.98060867
4	0.97603016
5	0.97629949
Mean Validation Accuracy	**97.81%**

Classification Report for Ensemble Model3

	Precision	Recall	F1-Score	Support
Anxiety	0.98	0.97	0.98	400
Bipolar	0.96	0.90	0.93	388
Depression	0.97	0.96	0.96	392
Normal	0.98	1.00	0.99	2136
PTSD	0.97	0.97	0.97	397
Accuracy	**97.76%**			
Macro avg	0.97	0.96	0.97	3713
Weighted avg	0.98	0.98	0.98	3713



(a) Confusion Matrix for Ensemble Model 3 (b) ROC AUC for Ensemble Model 3

Figure 42: Evaluation Results for Ensemble Model 3

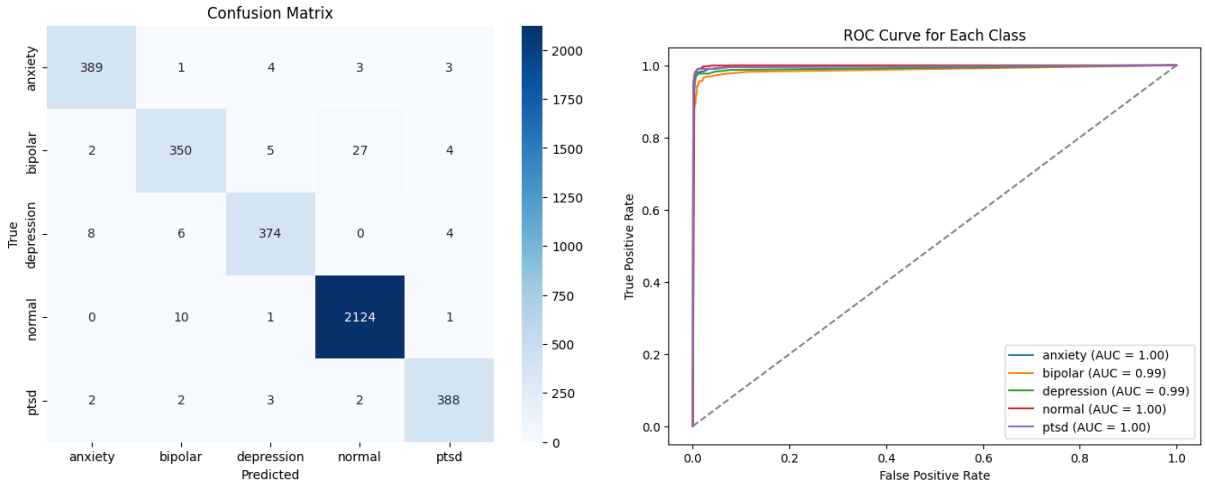
Ensemble Model 4

Cross-Validation Accuracy Scores

Cross-Validation Accuracy Scores	Value
1	0.97683814
2	0.96821977
3	0.97872340
4	0.96821977
5	0.97225963
Mean Validation Accuracy	**97.29%**

Classification Report for Ensemble Model4

Class	Precision	Recall	F1-Score	Support
Anxiety	0.97	0.97	0.97	400
Bipolar	0.95	0.90	0.92	388
Depression	0.97	0.95	0.96	392
Normal	0.99	0.99	0.99	2136
PTSD	0.97	0.98	0.97	397
Accuracy	**97.63%**			
Macro Avg	0.97	0.96	0.96	3713
Weighted Avg	0.98	0.98	0.98	3713



(a) Confusion Matrix for Ensemble Model 4 (b) ROC AUC for Ensemble Model 4

Figure 43: Evaluation Results for Ensemble Model 4

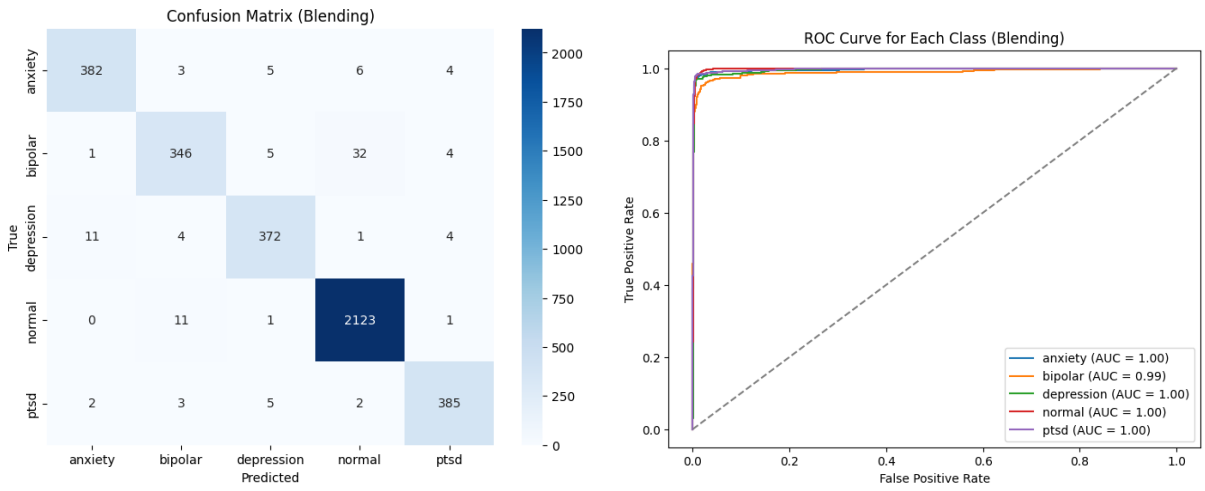
Ensemble Model 5

Cross-Validation Accuracy Scores

Cross-Validation Accuracy Scores	Value
1	0.96768765
2	0.97273645
3	0.97306397
4	0.96801347
5	0.96767677
Mean Validation Accuracy	**96.98%**

Classification Report for Ensemble Model5

	Precision	Recall	F1-Score	Support
Anxiety	0.96	0.95	0.96	400
Bipolar	0.94	0.89	0.92	388
Depression	0.96	0.95	0.95	392
Normal	0.98	0.99	0.99	2136
PTSD	0.97	0.97	0.97	397
Accuracy	**97.17%**			
Macro avg	0.96	0.95	0.96	3713
Weighted avg	0.97	0.97	0.97	3713



(a) Confusion Matrix for Ensemble Model 5 (b) ROC AUC for Ensemble Model 5

Figure 44: Evaluation Results for Ensemble Model 5

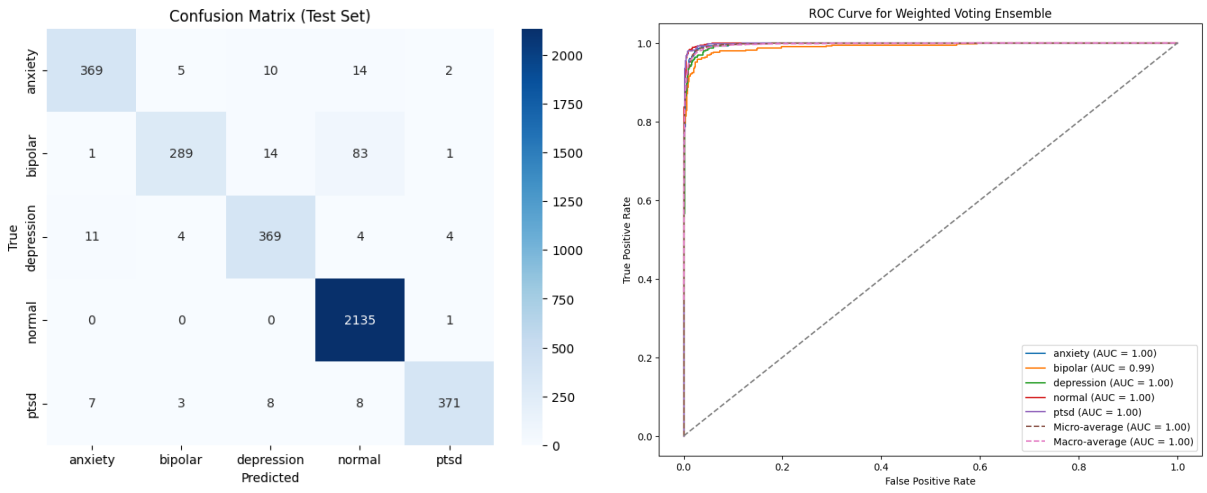
Ensemble Model 6

Cross-Validation Accuracy Scores

Cross-Validation Accuracy Scores	Value
1	0.9582547805009426
2	0.9477511446269863
3	0.9453272286560732
4	0.9464045246431457
5	0.9488284406140587
Mean Validation Accuracy	**94.93%**

Classification Report for Ensemble Model6

	Precision	Recall	F1-Score	Support
Anxiety	0.95	0.92	0.94	400
Bipolar	0.96	0.74	0.84	388
Depression	0.92	0.94	0.93	392
Normal	0.95	1.00	0.97	2136
PTSD	0.98	0.93	0.96	397
Accuracy	**95.15%**			
Macro avg	0.95	0.91	0.93	3713
Weighted avg	0.95	0.95	0.95	3713



(a) Confusion Matrix for Ensemble Model 6 (b) ROC AUC for Ensemble Model 6

Figure 45: Evaluation Results for Ensemble Model 6

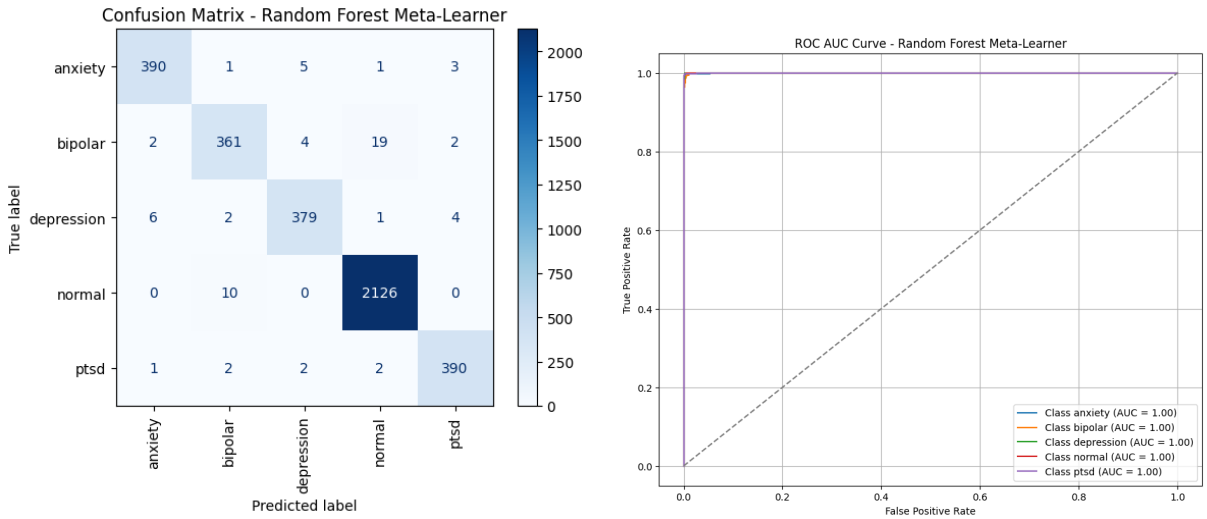
Ensemble Model 7

Cross-Validation Accuracy Scores

Cross-Validation Accuracy Scores	Value
1	0.980878
2	0.97899273
3	0.98410988
4	0.98168597
5	0.97764611
Mean Validation Accuracy	**98.03%**

Classification Report for Ensemble Model7

	Precision	Recall	F1-Score	Support
Anxiety	0.98	0.97	0.98	400
Bipolar	0.96	0.93	0.95	388
Depression	0.97	0.97	0.97	392
Normal	0.99	1.00	0.99	2136
PTSD	0.98	0.98	0.98	397
Accuracy	**98.20%**			
Macro avg	0.98	0.97	0.97	3713
Weighted avg	0.98	0.98	0.98	3713



(a) Confusion Matrix for Ensemble Model 7 (b) ROC AUC for Ensemble Model 7

Figure 46: Evaluation Results for Ensemble Model 7

Comparison of Ensemble Models for Mental Health Classification

Model	**Accuracy**	**Key Components**	**Advantages**	**Limitations**
Ensemble Model 1	96.08%	Logistic Regression + XGBoost	High precision and balanced performance (e.g., Anxiety: Prec=0.97, Rec=0.94, F1=0.95); robust macro/weighted averages	May not fully capture complex, non-linear, context-dependent patterns
Ensemble Model 2	97.93	XGBoost as meta-learner combining base models (LR, SVM, NB, LSTM)	Near-perfect ROC AUC (1.0 for Anxiety, Normal, PTSD; 0.99 for Bipolar, Depression); effective handling of imbalanced data	Potential overfitting if base model predictions are highly correlated
Ensemble Model 3	97.76	Random Forest used as meta-learner	High overall accuracy with very few misclassifications; robust due to bootstrapping and random feature selection	Slightly lower recall in some classes (e.g., Bipolar)
Ensemble Model 4	97.63%	Bagging classifier (similar to Random Forest)	Excellent performance with near-perfect results for the "Normal" class	Tendency to overfit due to using all features without feature-level randomness
Ensemble Model 5	97.17%	Blending Meta-Learner trained directly on base model predictions	Strong precision, recall, and F1-scores across classes; stable cross-validation performance	More prone to overfitting compared to stacking ensembles (e.g., with a Random Forest meta-learner)
Ensemble Model 6	95.15%	Weighted Voting ensemble combining base model outputs	Solid overall performance with high recall for "Normal" and computational efficiency	Lacks optimal inter-model learning; struggles with distinguishing bipolar disorder

Comparison of Ensemble Models for Mental Health Classification

Model	**Accuracy**	**Key Components**	**Advantages**	**Limitations**
Ensemble Model 7	98.03%	Transformer-based model (base) + meta-learner (Random Forest and others)	Near-perfect ROC AUC (1.0 across classes); excellent accuracy and robust classification	Increased computational complexity due to Transformer integration

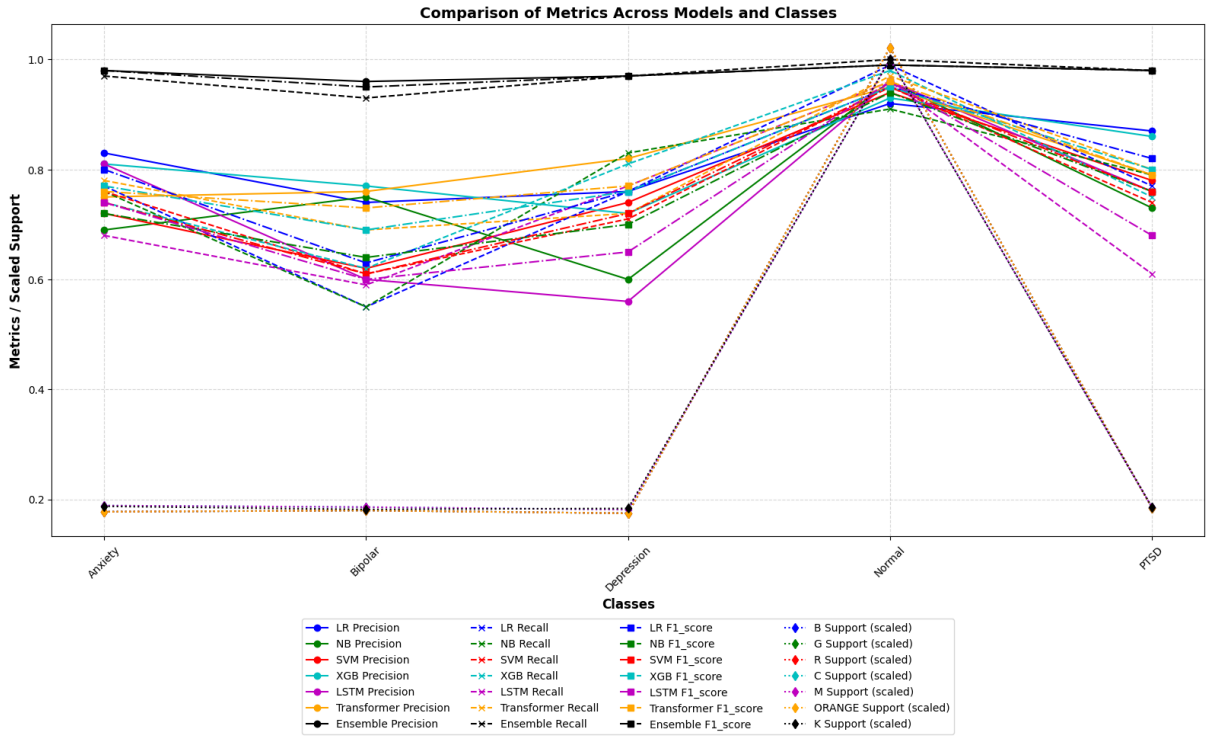


Figure 47: Comparison of Base Models and Ensemble Model7

Below is a comparison of all the ensemble models for reference

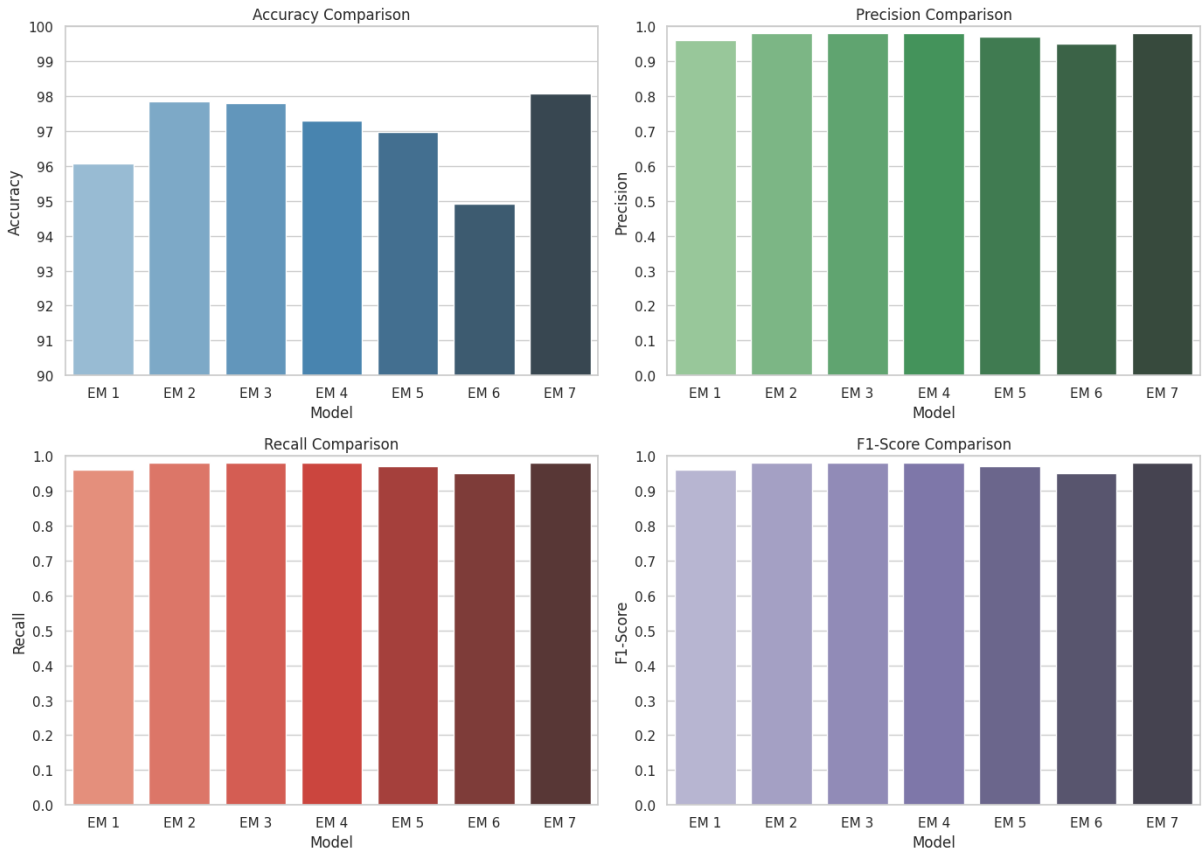


Figure 48: Comparison of all Ensemble Models

9.13 Result from hierarchical Ensemble Models

Ensemble Model 7 is applied on different subsets of a very large dataset to create hierarchical ensemble models. The performance of each model is evaluated, and the results are compared to the global ensemble model.

Performance Comparison of Models

Model	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6	ENSEMBLE
Logistic Regression	87.66	85.07	86.74	90.61	88.17	72.62	**90.90**
Naive Bayes	83.63	81.71	83.50	84.44	85.93	64.33	**84.15**
SVM	85.13	82.30	83.50	88.34	85.45	67.81	**91.79**
XGBoost	87.39	85.79	86.78	91.37	86.32	70.95	**67.67**
LSTM	84.91	81.63	81.83	87.22	85.38	67.74	**87.48**
Transformer	88.50	84.50	86.50	89.35	87.62	72.40	**91.53**
ENSEMBLE	**98.03**	**94.85**	**96.96**	**97.79**	**96.86**	**92.57**	**96.24** / **96.25**

Note: The final ensemble models from two different architectures achieved accuracies of **96.24%** and **96.25%** respectively.

The hierarchical ensemble model addresses the challenge of handling large datasets by adopting a modular and scalable approach. The dataset is divided into manageable subsets, each treated independently. For each subset, base models like Logistic Regression, Naive Bayes, SVM, LSTM, XGBoost, and Custom Transformers are trained and combined into a subset-specific ensemble using Random Forest as the meta learner. These subset ensembles are then aggregated to form a final global ensemble, capturing comprehensive learning from the entire dataset. This approach ensures efficient handling of large-scale data while maintaining high performance. Although the current implementation is sequential for testing purposes, the design is inherently scalable to a distributed architecture. In such a setup, subsets can be processed in parallel across multiple computational nodes. Each node trains base models, creates a subset ensemble, and sends results to a central controller, which aggregates them into a global ensemble. This parallelized framework aligns with distributed computing principles, making it ideal for real-world, large-scale applications.

The hierarchical structure of this approach aligns with distributed systems by effectively implementing data parallelism and task parallelism. By dividing the dataset into subsets, the computational load is distributed, preventing memory bottlenecks and ensuring resource optimization. Each subset is processed independently, enabling a modular design where tasks are decoupled. This design also enhances fault tolerance, as the failure of a single subset's processing does not affect the others. Moreover, the scalability of the approach allows it to adapt to varying resource constraints. Adding more computational nodes makes it feasible to handle additional subsets or process the data faster, showcasing its adaptability and efficiency. The distributed nature of the architecture ensures high throughput and reduces training time, making it suitable for scenarios involving large-scale datasets. The hierarchical ensemble model design is a future-ready solution that can transition seamlessly to distributed platforms, leveraging cloud services or on-premises high-performance computing clusters. The methodology combines the strengths of ensemble learning with the efficiencies of distributed systems to create a robust and scalable solution for modern data science challenges.

The second architecture optimizes the hierarchical ensemble approach by restructuring the model aggregation process to enhance scalability and computational efficiency. Instead of creating ensemble models for each of the n subsets independently, models of the same type from all subsets are combined into a single ensemble for that type using Random Forest (or Logistic Regression) as the meta learner. This results in six intermediate ensemble models, one for each model type: Logistic Regression, Naive Bayes, SVM, XGBoost, LSTM, and Transformer. These intermediate ensemble models are then used to form the final ensemble model, where Random Forest acts as the meta learner. This architectural adjustment significantly reduces the

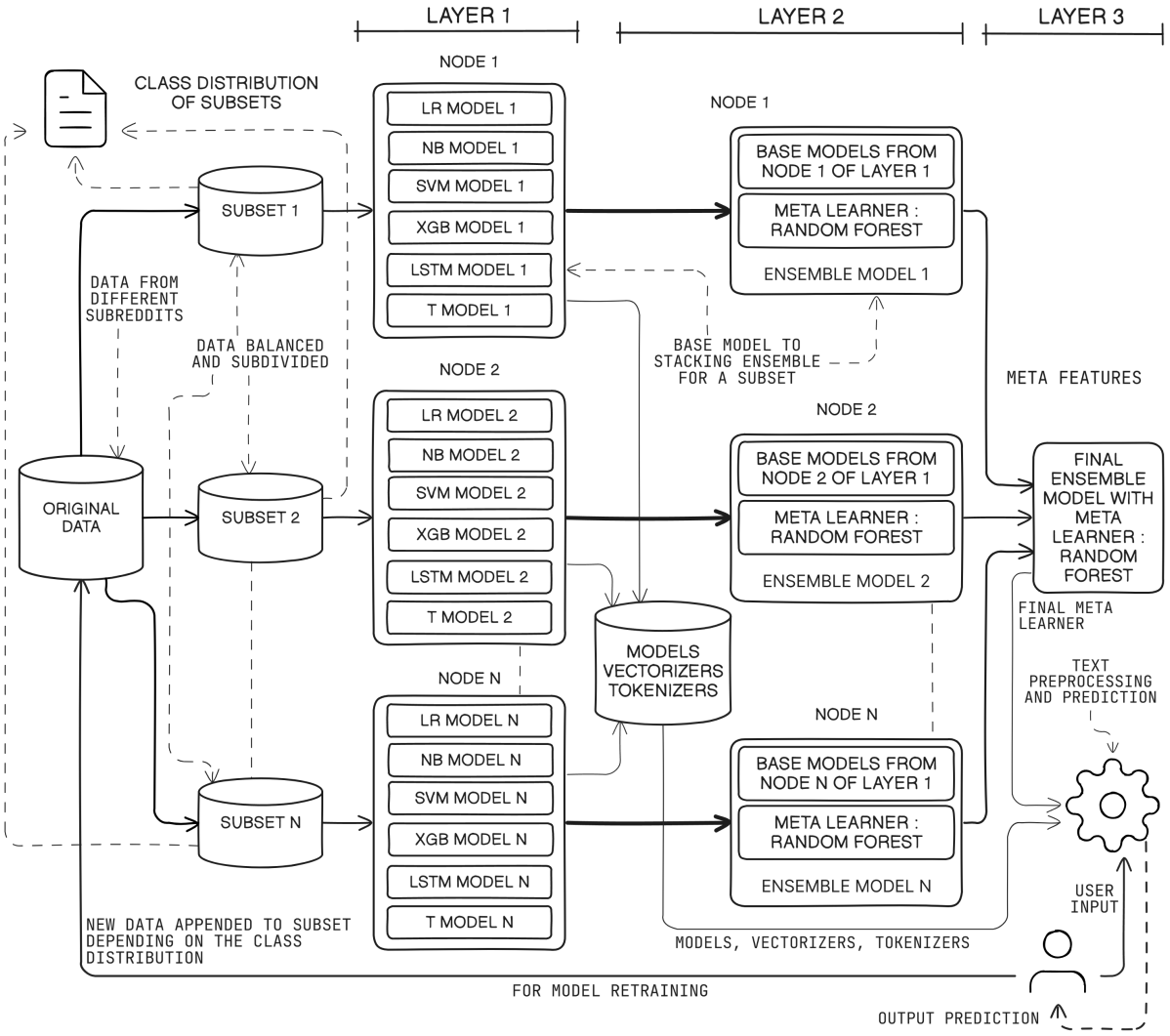


Figure 49: Scalable Distributed Architecture 1

number of intermediate ensemble models from n to six while preserving the hierarchical structure, thereby maintaining a consistent three-layer design. The second architecture achieved an accuracy of 96.25%. However, its design provides distinct advantages when dealing with much larger datasets or a higher number of subsets. As the dataset size increases, the number of subsets and computational nodes may also grow. In the previous architecture, this scaling would lead to a proportional increase in the number of intermediate ensemble models, which could strain computational resources and introduce inefficiencies in model aggregation. In contrast, the second architecture fixes the number of intermediate ensemble models at six, irrespective of the number of subsets, ensuring that the computational load remains manageable while scalability is maintained. Another significant improvement lies in the efficient use of hardware resources. In the second architecture, GPUs are utilized exclusively for training the LSTM and Transformer-based ensemble models, rather than being partially engaged across all inter-

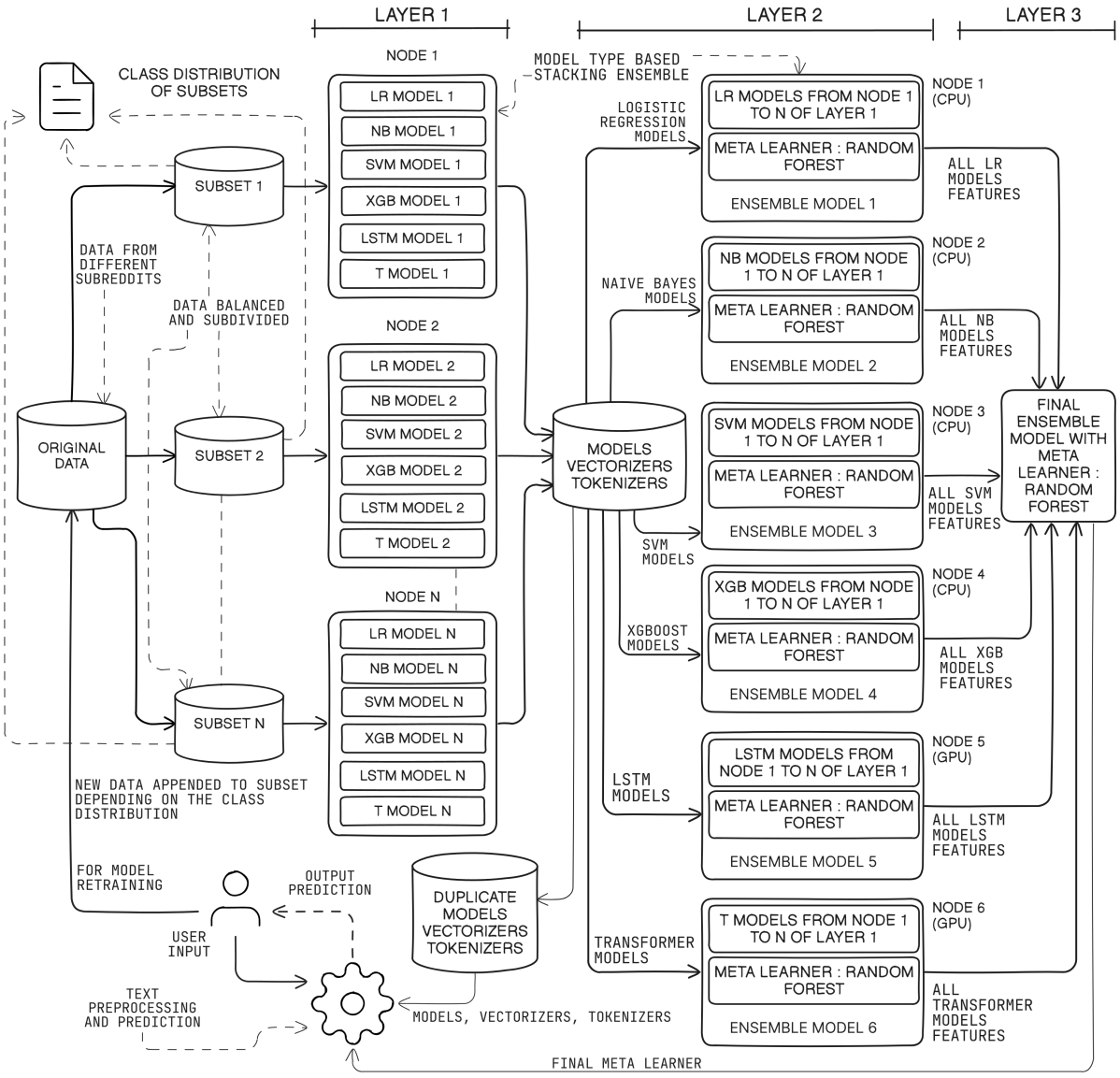


Figure 50: Scalable Distributed Architecture 2

mediate models as in the first architecture. This selective usage of GPUs streamlines resource allocation, ensuring that the most computationally intensive models benefit from accelerated hardware while reducing unnecessary overhead for other model types. The use of Logistic Regression as the meta learner in the intermediate ensemble models further contributes to this efficiency. Logistic Regression is computationally lightweight and linear, making it an excellent choice for aggregating predictions across subsets without introducing excessive complexity. Its linearity ensures faster training while maintaining strong predictive performance, particularly for well-separated data classes. Random Forest as a meta-learner is a better option for considering overfitting issues. A key addition to the second architecture is the implementation of an efficient data bus for transferring models of the same type from different subsets to the respec-

tive intermediate nodes responsible for creating the type-specific ensemble. This structured flow of data and models ensures seamless communication between subsets and intermediate nodes, reducing latency and synchronization overhead. By fixing the number of intermediate ensemble models at six, the architecture limits the number of aggregation layers while still leveraging the diversity and complementary strengths of all base models. This design strikes a balance between computational efficiency and predictive performance, making it particularly advantageous for large-scale datasets. The second architecture's streamlined design, efficient resource utilization, and fixed number of intermediate ensemble models make it a robust and scalable solution for modern data science challenges. By addressing the potential bottlenecks of the first architecture while maintaining comparable performance, it provides a practical alternative for real-world applications where scalability and computational efficiency are critical.

Architecture 1 takes more time in cross-validation compared to Architecture 2 due to its decentralized design. In Architecture 1, each subset independently creates its intermediate ensemble model, resulting in a larger number of models being trained and validated. This increases the computational workload and requires more synchronization across subsets. On the other hand, Architecture 2 aggregates models of the same type into a single intermediate ensemble model, reducing the number of intermediate ensembles. This streamlining minimizes redundancy and allows for faster cross-validation while maintaining predictive performance. In Architecture 2, models of the same type are used as base models in the intermediate ensemble to ensure consistency and simplify the aggregation process. Mixing different types of models, such as Logistic Regression from one subset and SVM from another, would introduce heterogeneity in predictions, making the meta-learning process more complex. By grouping models of the same type, the meta-learner can better capture specific patterns and variations unique to each model type. This design also ensures that the strengths of each algorithm are utilized optimally, allowing the final ensemble to benefit from their complementary capabilities. The observation that SMOTE is not required and yields the same results as stratified K-fold cross-validation indicates that the dataset is either balanced or sufficiently robust to handle minor class imbalances. Stratified K-fold cross-validation ensures that each fold maintains the same class distribution as the original dataset, effectively mitigating the impact of class imbalance during training and testing. This suggests that the dataset's inherent balance, combined with the model's robustness, eliminates the need for synthetic oversampling techniques like SMOTE. Logistic Regression with L1 or L2 regularization is less suited for Architecture 2 because it takes more time to train on the larger, aggregated datasets at the intermediate level. Regularization adds computational overhead by optimizing penalty terms alongside the model's weights, which can slow down training. Furthermore, Logistic Regression may struggle with the diverse, high-dimensional feature space created by aggregated predictions from multiple subsets. Random Forest, as a meta-learner, is

a better choice in Architecture 2 because it is less prone to overfitting, can capture nonlinear relationships, and handles high-dimensional data effectively without extensive parameter tuning. Its ensemble-based nature ensures better generalization, making it more efficient and reliable for intermediate and final aggregation.

To determine n, the number of subsets, based on the size of the original dataset, it is crucial to consider computational efficiency, memory constraints, and sequential execution. Given that the dataset has $D = 167,229$ records and is processed sequentially in Google Colab with 12GB of RAM per node, the number of subsets n must strike a balance between memory usage and model performance. In this case, you chose $n = 6$ subsets, which implies a subset size S of:

$$S = \frac{D}{n} = \frac{167,229}{6} \approx 27,872 \text{ records per subset.}$$

The choice of $n = 6$ is reasonable given the following factors:

1. **Memory Constraints**: Google Colab provides 12GB of RAM. Each subset must fit within this memory while accommodating the model's requirements for training and validation. Processing approximately 27,833 records at a time is well-suited to this memory limit for most machine learning models, including Logistic Regression, SVM, LSTM, and Transformer-based models.
2. **Sequential Execution**: Since the architectures are implemented sequentially, the number of subsets n does not need to align with the number of computational nodes. Instead, the goal is to divide the dataset into manageable chunks that reduce training time and memory overhead for each subset.
3. **Performance and Aggregation**: With $n = 6$, both architectures remain computationally feasible. In Architecture 1, $n = 6$ leads to six independent intermediate ensemble models, which are aggregated into the final ensemble. In Architecture 2, models of the same type from all six subsets are combined into six intermediate ensemble models, one for each type of algorithm.

The web application does not use the model for bigger dataset but focuses on the model received from the first intermediate node of Architecture 1. The reason being that the model for the bigger dataset takes a lot of time, more than 20 minutes, if the user chooses for retraining the model after an input.

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 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. 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. 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.

Probable Method for implementation of Distributed Architecture

The dataset is divided into subsets for parallel processing in a distributed system. Data partitioning can be performed using distributed data storage systems such as Hadoop HDFS, Amazon S3, or Google Cloud Storage. Tools like Apache Spark or Dask are particularly effective for splitting datasets into logical subsets based on criteria such as size, record count, or specific data attributes like time ranges, regions, or categories. Each subset is then assigned to a worker node for processing. Alternatively, data sharding can be used at the database level, where SQL queries or shard keys in databases like MongoDB or Cassandra create logical partitions of the dataset. Each worker node is responsible for training base models on its assigned subset. Worker nodes are typically implemented using containers such as Docker or virtual machines, each equipped with necessary machine learning frameworks like Scikit-learn, TensorFlow, or PyTorch. A central master node, acting as the orchestrator, distributes tasks such as training models like Logis-

tic Regression, SVM, and others to these workers. Frameworks such as Apache Spark MLlib or TensorFlow Distributed Strategy facilitate this process. Each worker independently processes its subset, training multiple base models in parallel, while periodically reporting training statuses back to the master node. After training the base models for each subset, a subset-specific ensemble is created. This is typically done by employing a meta learner like Random Forest to aggregate the outputs of the base models within each worker. Frameworks like Scikit-learn or XGBoost can be utilized locally on the workers for this purpose. The trained subset-specific ensembles are then saved to distributed storage systems such as S3 or HDFS for later aggregation into a final ensemble model. To create the final ensemble model, predictions or model weights from all subset-specific ensembles are collected by the master node. The aggregation process can use techniques such as weighted averaging or stacking, implemented through another meta learner. In cases where the number of subset-specific ensembles is large, the aggregation itself can be distributed using paradigms like MapReduce. The final model is saved to shared storage to facilitate deployment. Once the final ensemble model is created, it is deployed using distributed inference platforms like TensorFlow Serving, Kubernetes, or AWS SageMaker, ensuring scalability by serving the model across multiple instances. Monitoring tools such as Prometheus or Grafana are employed to track the performance and resource utilization of the distributed system, ensuring reliable operation. For implementing such a system, various technologies are available. Distributed computing frameworks like Apache Spark or Dask are ideal for parallel data processing and machine learning. TensorFlow Distributed Strategy or PyTorch Distributed are suitable for scalable training of deep learning models. Task orchestration can be managed using tools like Apache Airflow or Prefect, while Kubernetes is used for container orchestration and scaling worker nodes. Storage and sharding can be handled by systems like Hadoop HDFS, Amazon S3, or databases like Cassandra and MongoDB. For managing communication between nodes, message-passing systems such as RabbitMQ or Kafka are employed. Network overhead must be minimized to avoid communication bottlenecks between the master node and workers. Fault tolerance is critical, and distributed frameworks with built-in recovery mechanisms should be used.

The implementation of threading in the web application presents a significant opportunity for future improvements, particularly in enhancing the scalability and efficiency of the system. Threading can address several real-world challenges by allowing the application to handle multiple tasks concurrently, thereby improving responsiveness and reducing bottlenecks. Functions involving heavy I/O operations, such as downloading videos or images, extracting audio, or transcribing speech, can be threaded to overlap network and disk operations. This would minimize the idle time caused by waiting for these tasks to complete, allowing the system to serve multiple user requests simultaneously.

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] Jetli Chung and Jason Teo. Single classifier vs. ensemble machine learning approaches for mental health prediction. *Brain Informatics*, 10(1):1, jan 2023.

[4] 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.

[5] 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.

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

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

[8] Dip Kumar Saha, Tuhin Hossain, Mejdl Safran, Sultan Alfarhood, M. F. Mridha, and Dunren Che. Ensemble of hybrid model based technique for early detecting of depression based on svm and neural networks. *Scientific Reports*, 14(1):25470, oct 2024.

[9] 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.

[10] Hongzhi Zhang and M. Omair Shafiq. Survey of transformers and towards ensemble learning using transformers for natural language processing. *Journal of Big Data*, 11(1):25, feb 2024.

APPENDIX A - Prototype

Below are the flow diagrams for the various options that the web application provides.

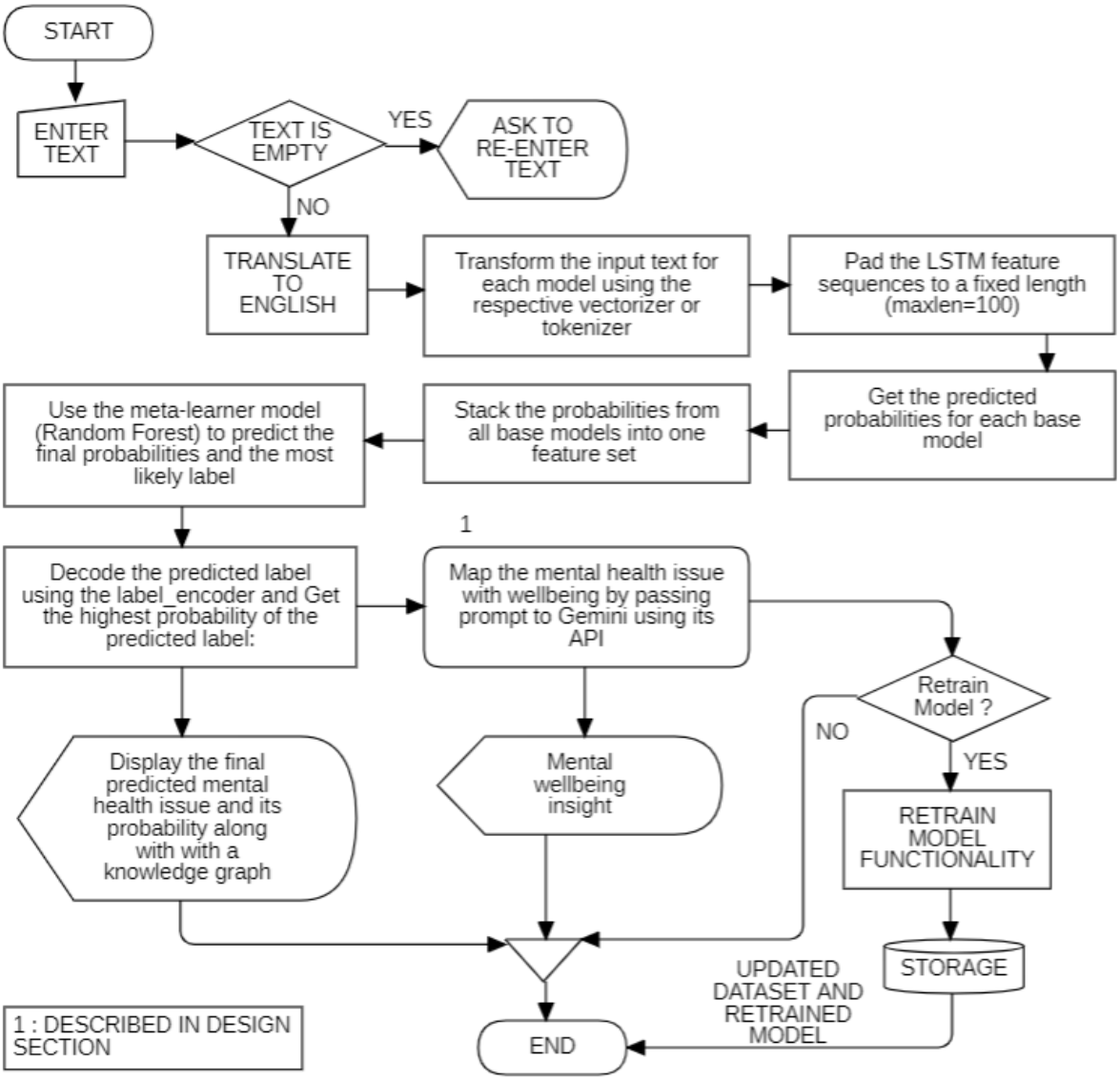


Figure 51: Text Classification Flow Diagram

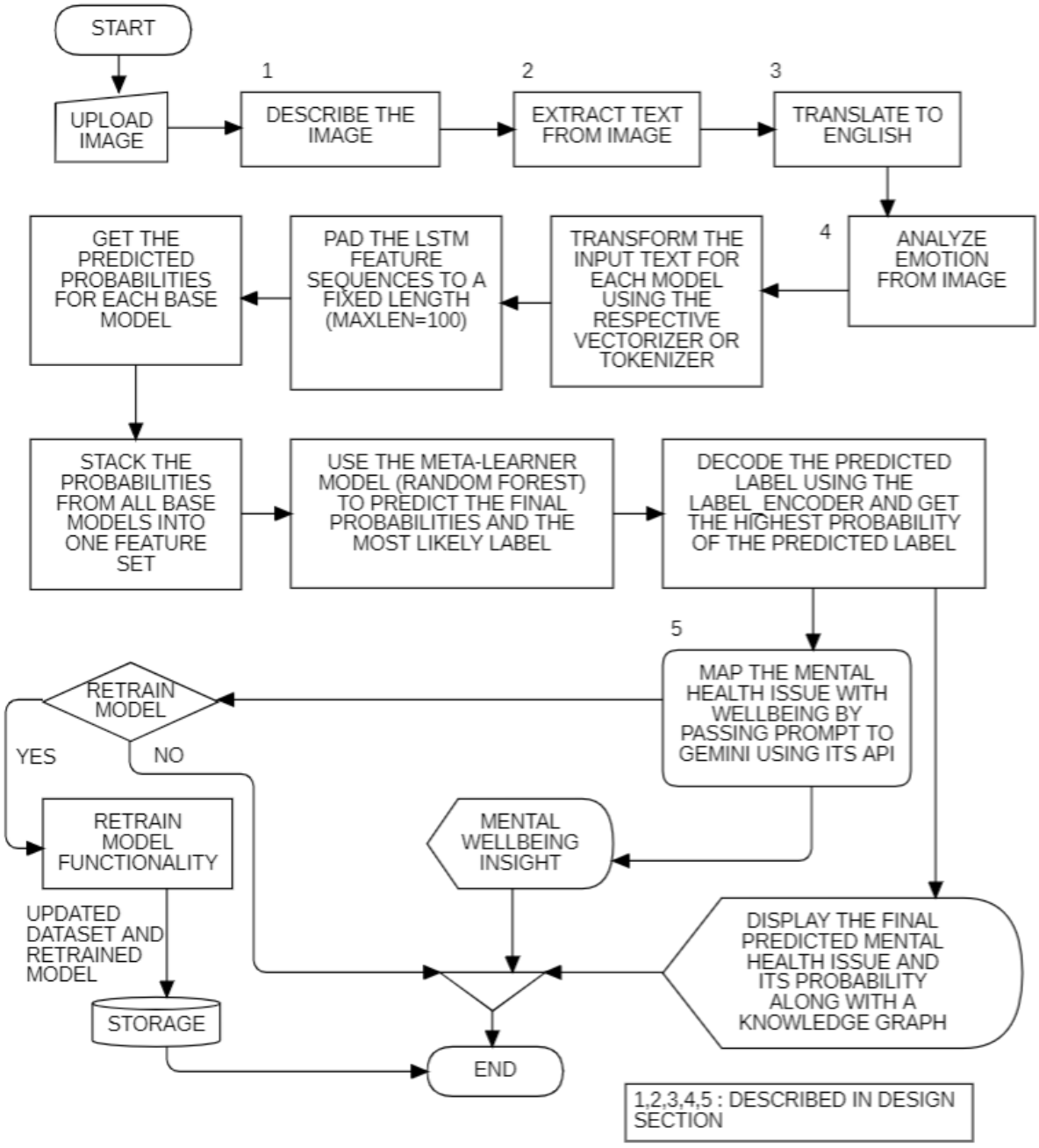


Figure 52: Image Classification Flow Diagram

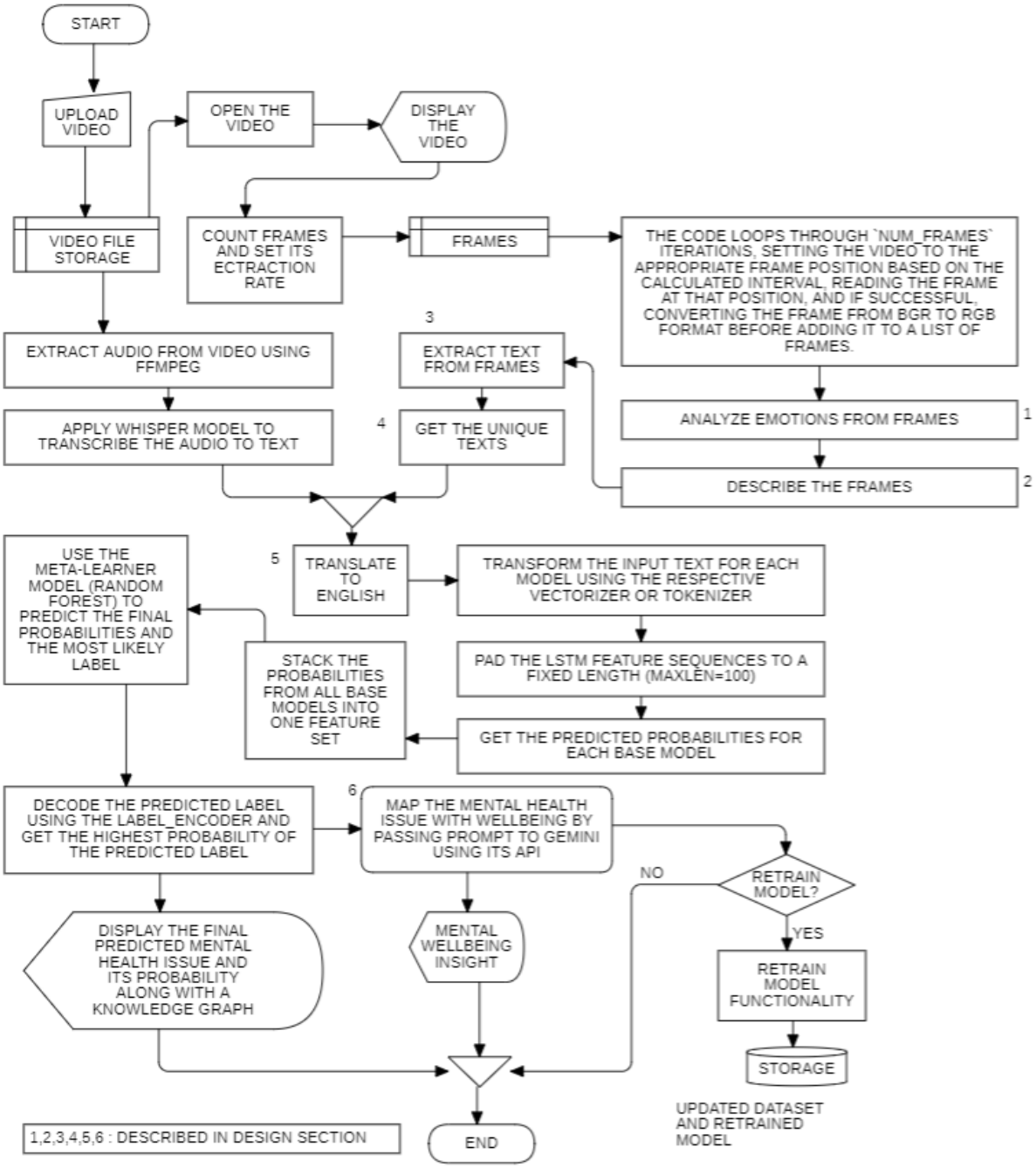


Figure 53: Video Classification Flow Diagram

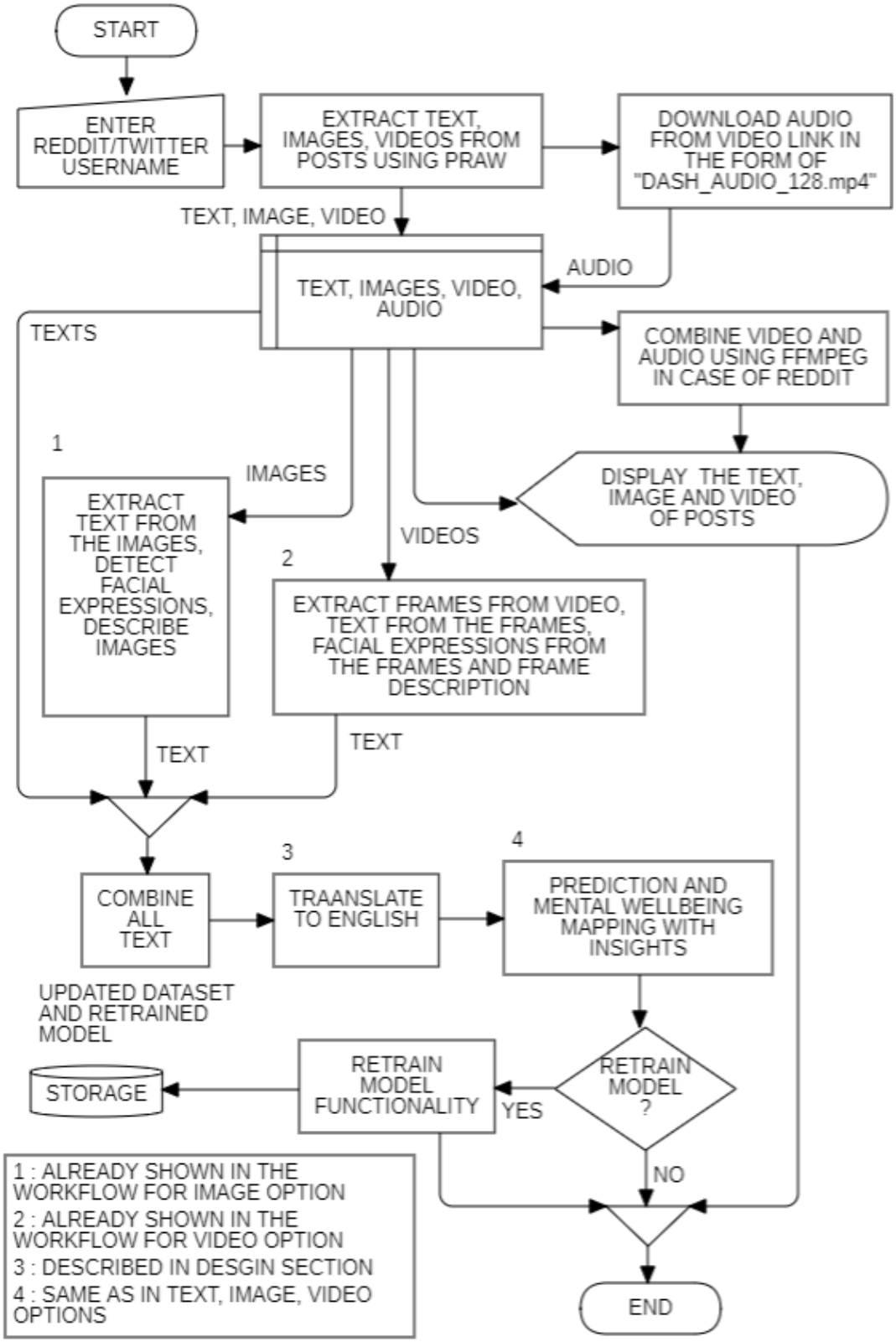


Figure 54: Reddit and Twitter username Classification Flow Diagram

Below are some screenshots from the web application.

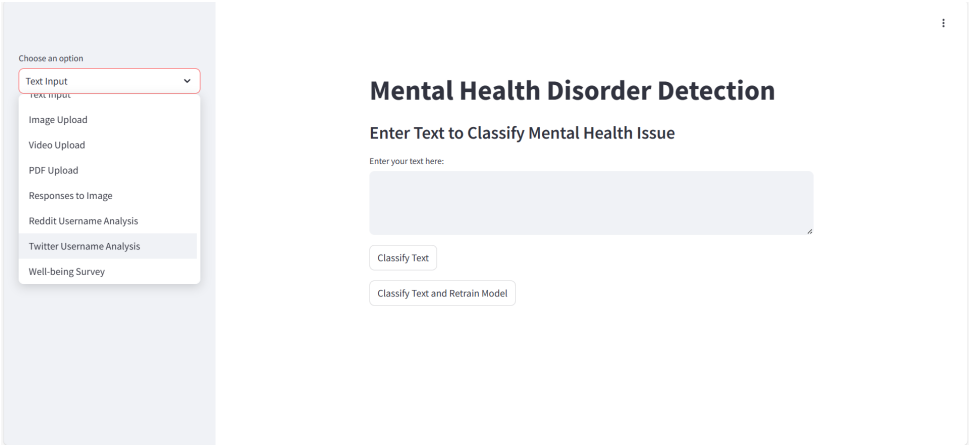


Figure 55: Website with all options

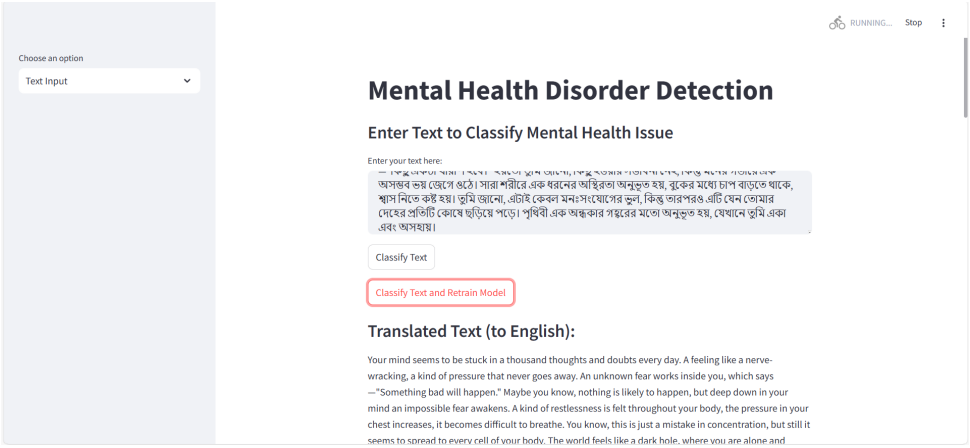


Figure 56: Entering Text for classification

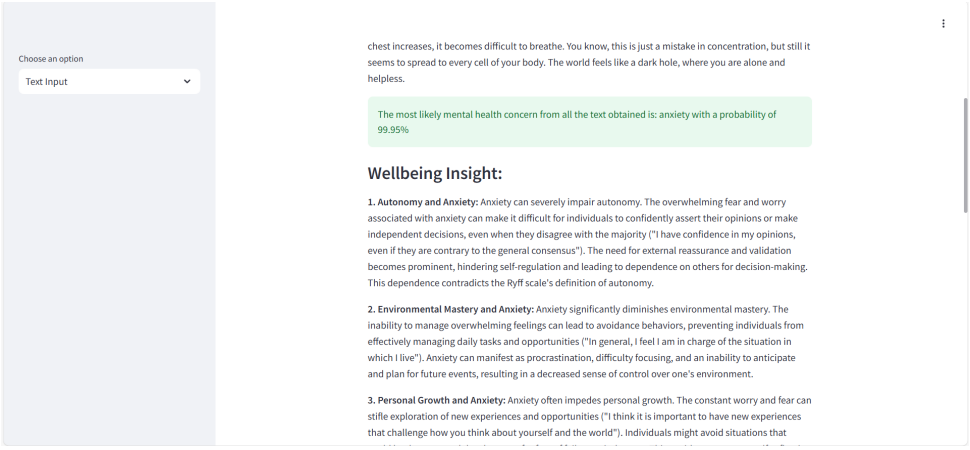


Figure 57: Text Classification Result

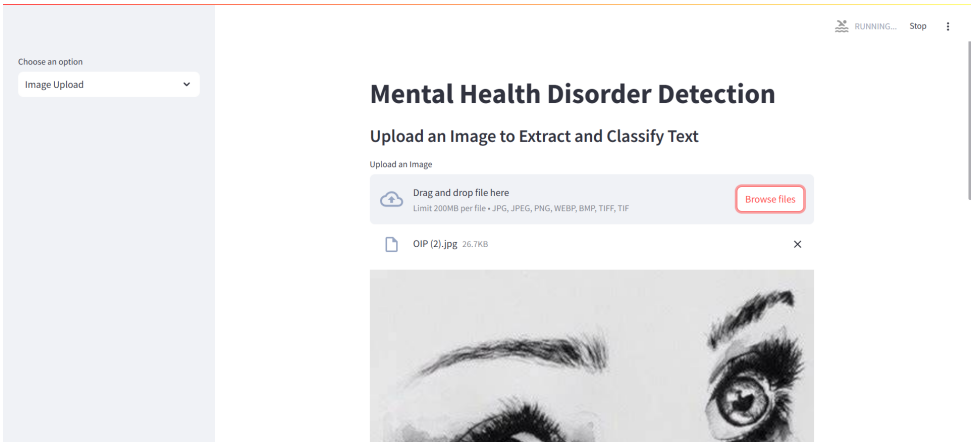


Figure 58: Upload Image

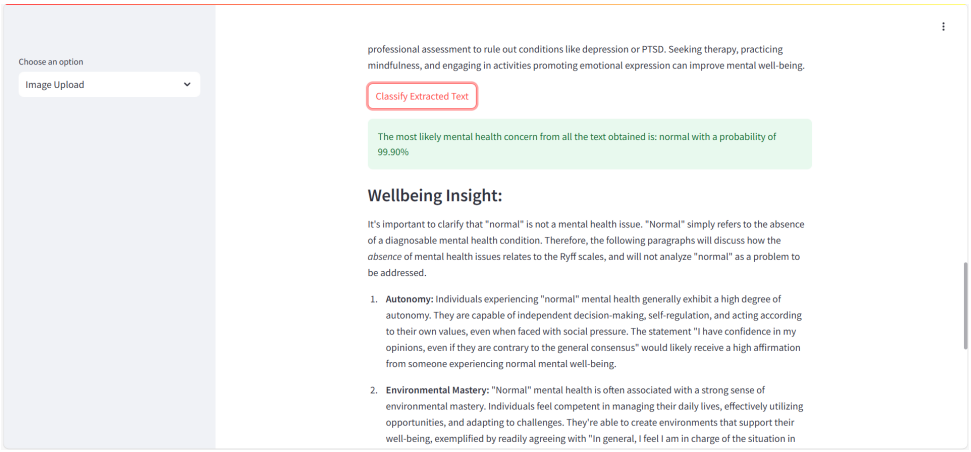


Figure 59: Image Classification Result

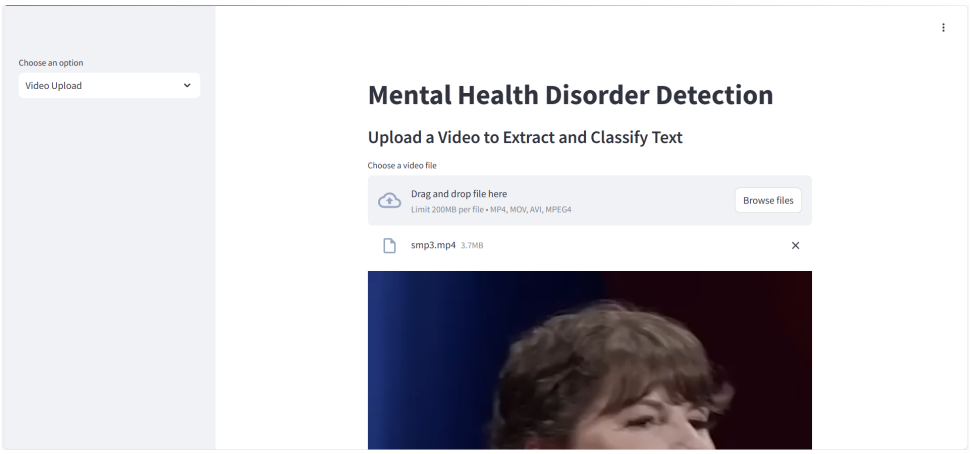


Figure 60: Upload Video

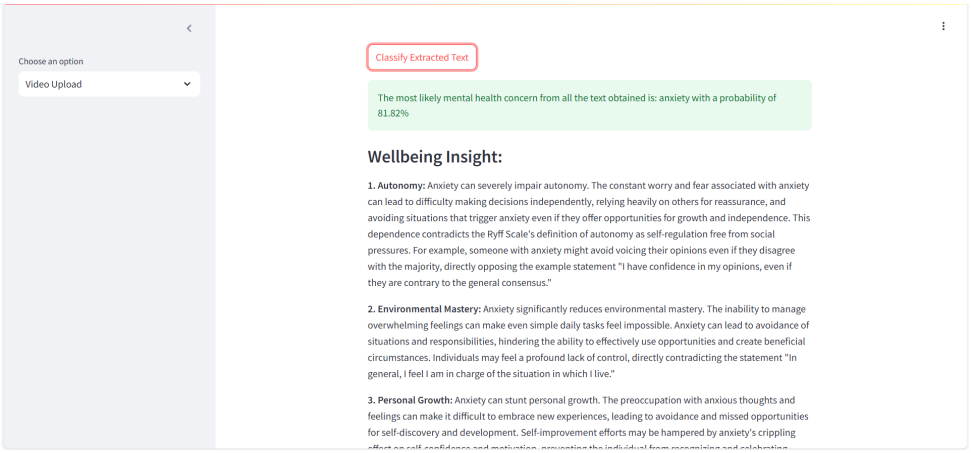


Figure 61: Video Classification Result

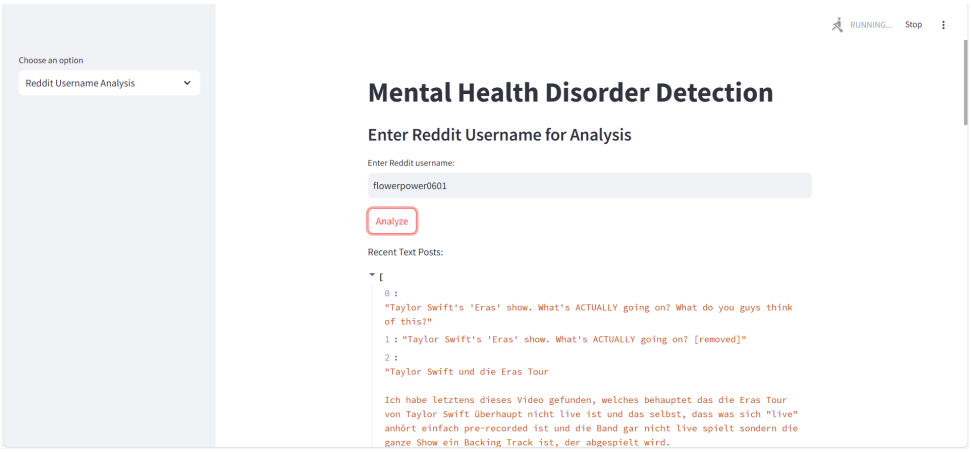


Figure 62: Reddit User Analysis

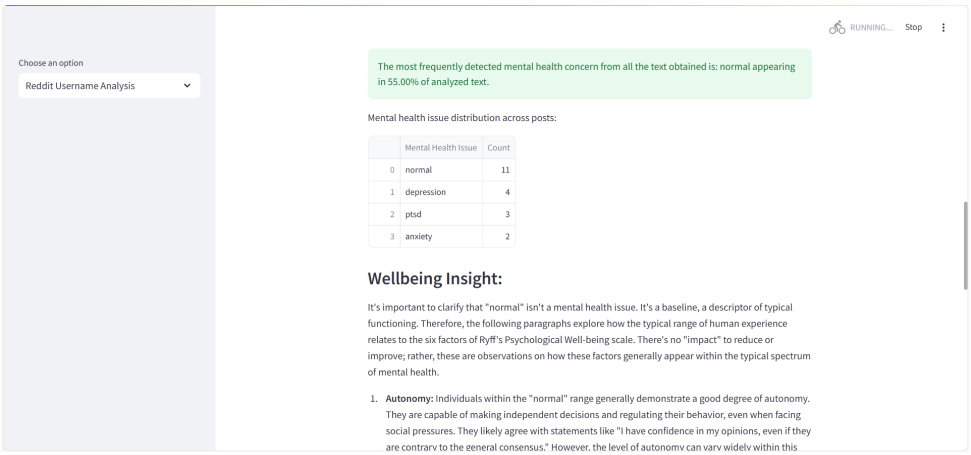


Figure 63: Result from Reddit Posts Analysis

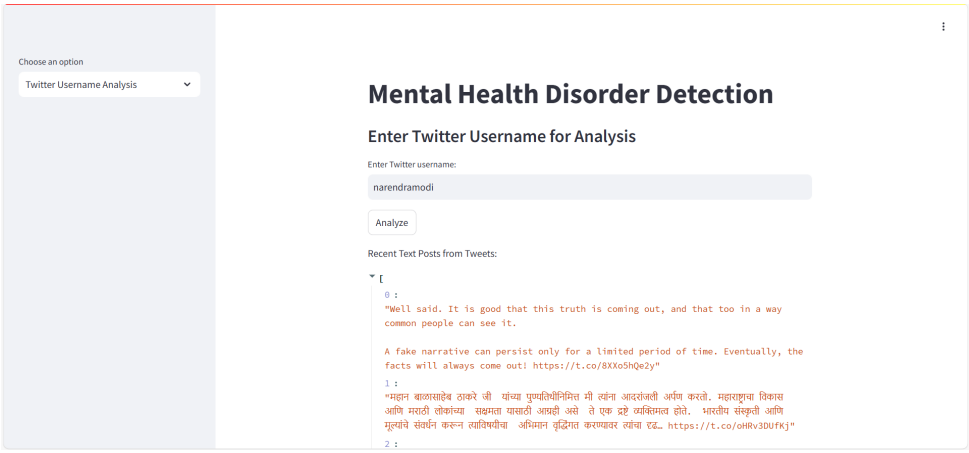


Figure 64: Twitter User Analysis

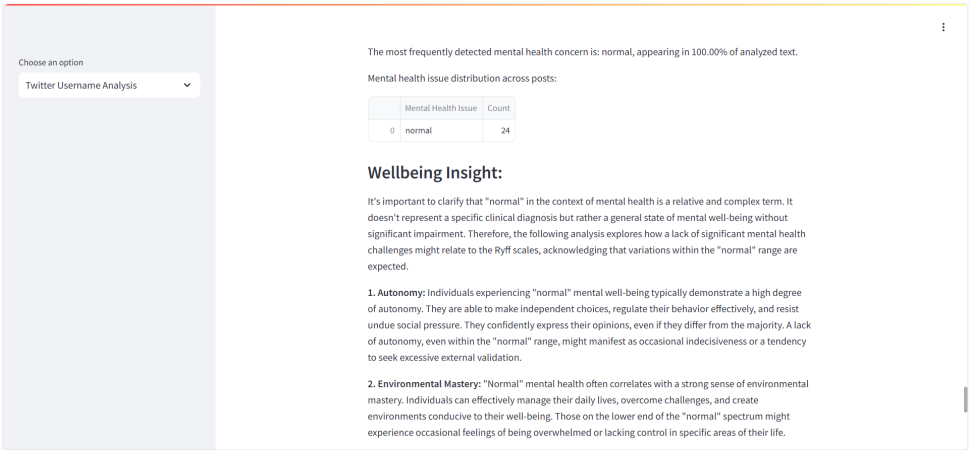


Figure 65: Result from Twitter Posts Analysis

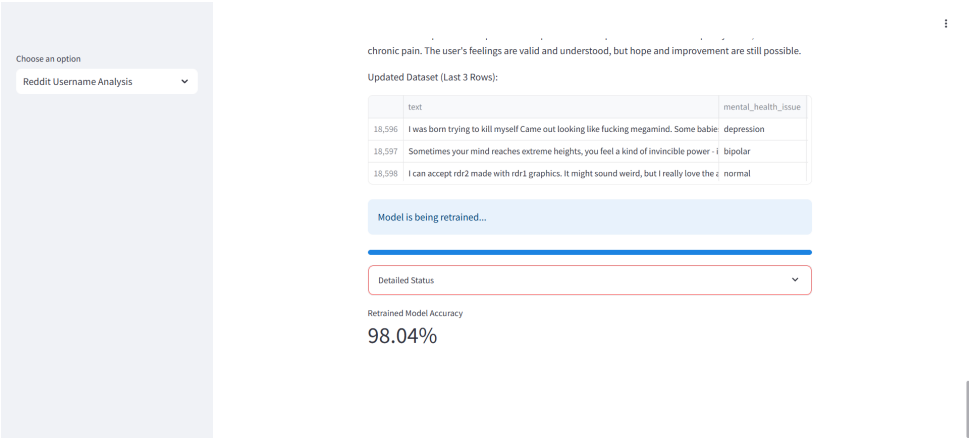


Figure 66: Result from Reddit Analysis and Model Retraining

Figure 67: Result from the emotion analysis of facial expression

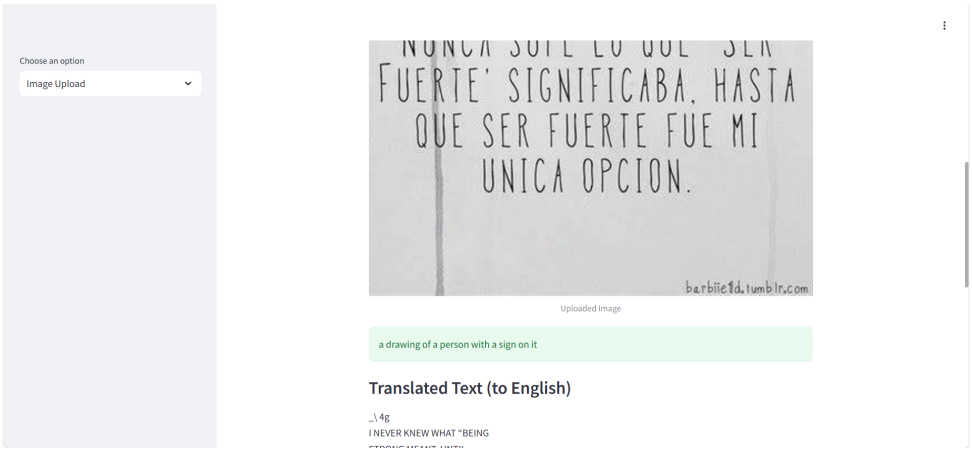


Figure 68: Generate Image Caption

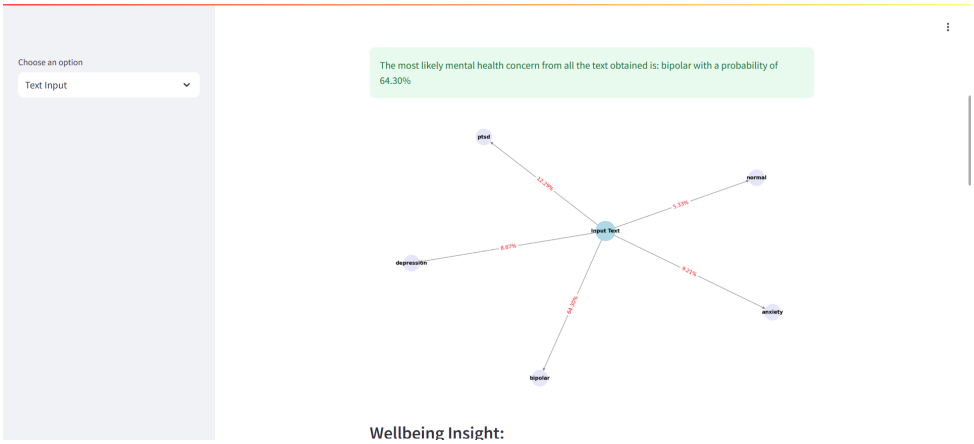


Figure 69: Knowledge Graph from classification