

A
PROJECT REPORT ON
**DETECTION AND PREDICTION OF FUTURE MENTAL
DISORDER FROM SOCIAL MEDIA DATA USING
MACHINE LEARNING, ENSEMBLE LEARNING, AND
LARGE LANGUAGE MODELS**

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

By
K S Saravana Kumar
(23BFF00043)

Under the esteemed guidance of

B. Sai Hemanth, MCA
Assistant Professor



DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

**SRI VENKATESWARA COLLEGE OF ENGINEERING
(AUTONOMOUS)**

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)

Accredited by NAAC with 'A' Grade

Opp. LIC Training Centre, Karakambadi Road, TIRUPATI-517507

2023-2025

**SRI VENKATESWARA COLLEGE OF ENGINEERING
(AUTONOMOUS)**

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)

Accredited by NAAC with 'A' Grade

Opp. LIC Training Centre, Karakambadi Road, TIRUPATI-517507

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

2023-2025



CERTIFICATE

This is to certify that the project report entitled ***"DETECTION AND PREDICTION OF FUTURE MENTAL DISORDER FROM SOCIAL MEDIA DATA USING MACHINE LEARNING, ENSEMBLE LEARNING, AND LARGE LANGUAGE MODELS"*** is a bonafide record of the project work done and submitted by

K S Saravana Kumar

(23BFF00043)

for the partial fulfillment of the requirements for the award of **MASTER OF COMPUTER APPLICATIONS** Degree from SRI VENKATESWARA COLLEGE OF ENGINEERING (AUTONOMOUS) Affiliated to JNT University Anantapur.

GUIDE

HEAD OF THE DEPARTMENT

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I am thankful to my guide **B. Sai Hemanth**, Assistant Professor for his valuable guidance and encouragement. His helping attitude and suggestions have helped me in the successful completion of the project.

I would like to express my sincere thanks to **Dr. E. Sreedevi**, Professor, Head of the Department of MCA, for her kind help and encouragement during the course of my study and in the successful completion of the project work.

I express my sincere thanks to **Dr. N. Sudhakar Reddy**, Principal, S.V College of Engineering, Tirupati.

Successful completion of any project cannot be done without proper support and encouragement. My sincere thanks to the Management for providing all the necessary facilities during the Course of my study.

I would like to thank my parents and friends, who have the greatest contributions in all my achievements, for the great care and blessings in making me successful in all my endeavors.

I would like to express my deep gratitude to all those who helped me directly or indirectly to transform an idea into my working project.

K S SARAVANA KUMAR

(23BFF00043)

DECLARATION

I hereby declare that project entitled **“Detection and Prediction of future Mental Disorder from Social Media Data Using Machine Learning ,Ensemble Learning, and Large Language Models”** submitted to the Department of MASTER Of COMPUTER APPLICATIONS, **SRI VENKATESWARA COLLEGE OF ENGINEERING, TIRUPATI** in partial fulfillment of Requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS**.

This project is the result of my own effort and it has not been submitted to any other University or Institution for the award of any degree other than specified above.

SIGNATURE OF THE STUDENT

ABSTRACT

Depression is a common but serious mental health disorder. Still, most people dealing with depression do not approach doctors for this problem. On the other hand, the use of Social Media Sites like Twitter is expanding extremely fast. Nowadays, people tend to rely on these social media platforms to share their emotions and feelings through their feed. Thus, this readily available content on social media has become helpful for us to analyze the mental health of such users. We can apply various machine learning techniques on this social media data to extract the mental health status of a user focusing on Depression. Detecting texts that express negativity in the data is one of the best ways to detect depression. In this paper, we highlighted this problem of depression and discussed various techniques on how to detect it. we implemented a system that can detect if a person on social media is going through depression or not by analyzing the user's data and activities by using various machine learning techniques.

TABLE OF CONTENTS

| S.NO | CONTENTS | PAGE NO |
|------|--------------------------------------|---------|
| 1 | INTRODUCTION | 1 |
| | 1.1 GENERAL | 1 |
| | 1.2 OBJECTIVE OF THE PROJECT | 2 |
| | 1.3 DESCRIPTION OF THE PROJECT | 3 |
| 2 | LITERATURE SURVEY | 4 |
| 3 | SYSTEM ANALYSIS | 7 |
| | 3.1 EXISTING SYSTEM | 7 |
| | 3.2 PROPOSED SYSTEM | 8 |
| | 3.3 HARDWARE & SOFTWARE REQUIREMENTS | 9 |
| 4 | SYSTEM DESIGN | 10 |
| | 4.1 SYSTEM ARCHITECTURE | 10 |
| | 4.2 UML DIAGRAMS | 12 |
| 5 | IMPLEMENTATION | 20 |
| | 5.1 DESCRIPTION OF MODULES | 20 |
| | 5.2 SOFTWARE ENVIRONMENT | 21 |
| 6 | ALGORITHMS | 32 |
| 7 | SYSTEM TESTING | 37 |
| | 7.1 TYPES OF TESTS | 38 |
| | 7.2 SYSTEM TEST | 39 |
| 8 | RESULT | 40 |
| 9 | CONCLUSION AND FUTURE WORK | 42 |

| S.NO | CONTENTS | PAGENO |
|-------------|-------------------|---------------|
| 10 | APPENDIX | 43 |
| | 10.1 SAMPLECODE | 43 |
| | 10.2 SCREEN SHOTS | 49 |
| 11 | REFERENCES | 53 |

LISTOF FIGURES

| FIG NO | NAME OF FIGURES | PAGENO |
|---------------|------------------------|---------------|
| 1 | USE CASE DIAGRAM | 13 |
| 2 | CLASS DIAGRAM | 14 |
| 3 | SEQUENCE DIAGRAM | 15 |
| 4 | ACTIVITY DIAGRAM | 17 |
| 5 | ER DIAGRAM | 19 |

1. INTRODUCTION

1.1 GENERAL

Social media platforms are nowadays used by almost every single person on earth. People use it to express their feelings and attitudes towards everything, including other people, products, weather, social events, and political issues . Natural language processing (NLP) and new AI techniques have also led to the development of many new technological advancements that are used by all people in their ordinary life. Most popular NLP and AI modern techniques used by people worldwide are Machine translation, information extraction, information retrieval, question answering, text memorization, automatic assistance, and recommendation chat-bots and apps, etc. Together Social media platforms with NLP and AI have made a lot of things easier to people, like fast communication across different countries, giving people the ability to know news that happens everywhere on spot, not just that but it gave them the ability to express what they think and feel using posts and comments. Those facts led to having huge chunks of data publicly available on the internet. Those huge chunks of data, especially text data have been used by NLP, AI, and Data Science researchers in many purposes including automatic sentiment analysis, network analysis, information extraction, knowledge graph building, information interpretation, etc. In the last few years, a new direction was added to the prementioned research directions that benefit from data on social media, which is automatic depression detection from social media posts. This new direction had recently gripped the attention of researchers and developed rapidly (see section II). The researchers were not limited to depression detection, but they extended their studies to detect other mental health disorders like ADHD, Bipolar, Depression, and Anxiety. By time this direction of research had also been extended to include studies of future mental disorders prediction for social media users rather than detecting the already existing disorders . This work, unlike the previous attempts in literature, deploys and compares many Machine learning algorithms (ML) and Deep learning models (DL) to detect and predict mental disorders from REDDIT data. REDDIT is a social news forum or platform that is continuously curated by the site members to cover some important data points that includes but not limited to: post title, post body, comments, time stamps, shares, and scores . In this work the proposed approaches are deployed, compared, and then discussed comprehensively to get the best possible insights and conclusions. REDDIT data was used to validate and

evaluate the proposed work. The main advantage of this work over all the state-of-the-art work in literature is the detection of all mental disorders not only depression or anxiety, but it addresses the detection of all kinds of mental disorders which are: ADHD, Bipolar, Anxiety, Depression.

1.2 OBJECTIVE OF THE PROJECT:

The objective of this project is to detect and predict early signs of mental disorders from user-generated social media data. It leverages machine learning and ensemble learning techniques to improve prediction accuracy. Large Language Models (LLMs) are utilized to analyze linguistic patterns and contextual cues in posts. The goal is to enable proactive mental health interventions through automated, data-driven insights.

The key objectives of this project would include:

- Early Detection
- Accurate Diagnosis
- Non-Invasive and Convenient Screening
- Personalized Medicine
- Feature Importance and Interpretability
- Real-Time Monitoring

1.3 Description of the Project:

Mental health challenges are rising globally, yet early detection and intervention remain limited due to stigma, lack of resources, and late clinical presentation. With the widespread use of social media platforms, people often share personal thoughts, feelings, and behaviors online — creating a rich, real-time source of psychological data. This project aims to harness that potential by using **Machine Learning (ML)**, **Ensemble Learning**, and **Large Language Models (LLMs)** to detect and predict early signs of mental health disorders from social media data.

We propose a system that analyzes users' posts, comments, images, and interaction patterns across platforms like Twitter, Reddit, Instagram, or TikTok to identify linguistic, emotional, and behavioral markers associated with conditions such as depression, anxiety, bipolar disorder, or emerging psychosis. Traditional ML techniques (e.g., SVM, Random Forests, Gradient Boosting) will be combined through ensemble methods to improve prediction robustness, while state-of-the-art LLMs (like GPT, BERT, RoBERTa) will handle complex language patterns, sentiment shifts, and context understanding that simpler models may miss.

The pipeline includes several stages:

- **Data Collection & Preprocessing:** Scraping public social media data (with ethical considerations), anonymization, noise filtering, and structuring text/image data.
- **Feature Engineering:** Extracting linguistic, sentiment, temporal, and behavioral features.
- **Modeling:** Training individual ML classifiers, fine-tuning LLMs on domain-specific corpora, and combining them using ensemble strategies (e.g., stacking, voting).
- **Prediction & Early Warning:** Outputting risk scores or alerts for potential future mental disorder emergence, enabling proactive intervention.

By leveraging the complementary strengths of classic ML, ensemble techniques, and the nuanced understanding of LLMs, this project seeks to build a scalable, accurate, and ethically responsible tool for **early mental health risk detection**. It has applications in public health monitoring, clinical support systems, and digital mental health tools — potentially transforming how we identify and support individuals at risk **before** they reach crisis points.

2.LITERATURE SURVEY

1) Lifespan (Half-Life) of Social Media Posts: Update for 2024

AUTHORS: Scott M. Graffius

Social media is a ubiquitous presence in society, with 4.95 billion users worldwide. Social media platforms enable people to connect with family and friends, share information and ideas, and access news and entertainment. The platforms also play a vital role in business by providing companies with direct channels to reach and engage with their target audiences. Social media analytics provide valuable insights that can support efforts to level-up engagement and results. The relevance and engagement of posts have a limited lifespan. An advantageous objective metric is half-life. It's the time it takes for a post to receive half of its total engagement (such as likes, shares, and comments). Corresponding data can help inform strategic and tactical decisions such as the frequency and scheduling of posts. In 2018, Scott M. Graffius first published data on the lifespan (half-life) of social media posts. Algorithms and other factors on platforms change over time. For that reason, Graffius periodically renews the analysis. Introducing the new report: 'Lifespan (Half-Life) of Social Media Posts: Update for 2024.' The visual at the top of the attached article summarizes the average lifespan (half-life) for posts on Snapchat, X (formerly Twitter), Facebook, Instagram, LinkedIn, YouTube, Pinterest, and blogs. For details or to request a high-resolution version of the image

2) Predicting future mental illness from social media: A big-data approach

AUTHORS: Robert Thorstad

In the present research, we investigated whether people's everyday language contains sufficient signal to predict the future occurrence of mental illness. Language samples were collected from the social media website Reddit, drawing on posts to discussion groups focusing on different kinds of mental illness (clinical subreddits), as well as on posts to discussion groups focusing on nonmental health topics (nonclinical subreddits). As expected, words drawn from the clinical subreddits could be used to distinguish several kinds of mental illness (ADHD, anxiety, bipolar disorder, and depression). Interestingly, words drawn from the nonclinical subreddits (e.g., travel, cooking, cars) could also be used to distinguish different categories of mental illness, implying that the impact of mental illness spills over into topics unrelated to mental

illness. Most importantly, words derived from the nonclinical subreddits predicted future postings to clinical subreddits, implying that everyday language contains signal about the likelihood of future mental illness, possibly before people are aware of their mental health condition. Finally, whereas models trained on clinical subreddits learned to focus on words indicating disorder-specific symptoms, models trained to predict future mental illness learned to focus on words indicating life stress, suggesting that kinds of features that are predictive of mental illness may change over time. Implications for the underlying causes of mental illness are discussed.

3) A Novel Co-Training-Based Approach for the Classification of Mental Illnesses Using Social Media Posts

AUTHORS: Subhan Tariq

Recently, research community of certain domain showing their eagerness towards the use of social media networks to gain constructive knowledge in decision making and automation, such as aid to perform software development activities, crypto-currencies usage, network community detection and recommendation and so on. Recently, besides other domains of eHealth, the use of social media and big data analytics has become hot topic to predict the patient of mental illness involved in either depression, schizophrenia, eating disorders, anxiety or addictive behaviors. Problem: Traditional methods either need enough historic data or to keep the regular monitoring on patient activities for identification of a patient associated with a mental illness disease. Method: In order to address this issue, we propose a methodology to classify the patients associated with chronic mental illness diseases (i.e. Anxiety, Depression, Bipolar, and ADHD (Attention Deficit Hyperactivity Disorder) based on the data extracted from the Reddit, a well-known network community platform. The proposed method is employed through Co-training (type of semi-supervised learning approach) technique by incorporating the discriminative power of widely used classifiers namely Random Forrest (RF), Support Vector Machine (SVM), and Naïve Bayes (NB). We used Reddit API to download posts and top five associated comments for construction of a feature space. Results: The experimental results indicate the effectiveness of Co-training based classification rather than the state of the art classifiers by a margin of 3% on average in par with every state of art technique. In future, the proposed method could be employed to investigate any classification problem of any domain by extracting data from the social media.

4) Ensemble Hybrid Learning Methods for Automated Depression Detection

AUTHORS: LunaAnsar

Changes in human lifestyle have led to an increase in the number of people suffering from depression over the past century. Although in recent years, rates of diagnosing mental illness have improved, many cases remain undetected. Automated detection methods can help identify depressed or individuals at risk. An understanding of depression detection requires effective feature representation and analysis of language use. In this article, text classifiers are trained for depression detection. The key objective is to improve depression detection performance by examining and comparing two sets of methods: hybrid and ensemble. The results show that ensemble models outperform the hybrid model classification results. The strength and effectiveness of the combined features demonstrate that better performance can be achieved by multiple feature combinations and proper feature selection.

5) Depression Detection from Social Media Posts Using Multinomial Naive Theorem

AUTHORS: Rinki Chatterjee, Rajeev Kumar Gupta , Bhavana Gupta

All 350+ million people worldwide are suffering from mental disorder called Depression. An individual who is suffering from depression functions below average in life, is vulnerable to other diseases and in the worst-case, depression leads to suicide. There are many restraints preventing expert care from reaching people suffering from depression in time. Impediments such as social stigma associated with mental disorders, lack of trained health-care professionals and ignorance of the signs of depression owing to a lack of awareness of the disease. The World Health Organization (WHO) suggests that people who are depressed are regularly not correctly diagnosed and others who are misdiagnosed are prescribed antidepressants. Thus, there is a strong need to automatically assess the risk of depression. Social media platforms increasingly come closer to become a true digitization of the human social experience. In many cases people would in fact prefer to express themselves online than offline. In this paper, we are using Facebook comments as data set, and based on it categorizing the users as depressed or non-depressed

3. SYSTEM ANALYSIS

EXISTING SYSTEM:

Existing semi supervised learning approach combined with the broadly used supervised ML classifiers like Support Vector Machine (SVM), Naïve Bayes, and Random Forest to detect and classify different mental disorders from social media posts. The authors used Reddit to download posts and their associated comments to train and test their classifiers. The authors confirmed that the combination between the semi supervised approach and the supervised classifiers obtained better results, they also confirmed that the SVM with co-training achieved the better F1-score of 0.84. But this study was also limited to detecting depression and anxiety based on the detection of negative feelings.

DRAWBACKS

- The complexity of data: Most of the existing machine learning models must be able to accurately interpret large and complex datasets for Detection and prediction of Future Mental disorder.
- Data availability: Most machine learning models require large amounts of data to create accurate predictions. If data is unavailable in sufficient quantities, then model accuracy may suffer.
- Incorrect labeling: The existing machine learning models are only as accurate as the data trained using the input dataset. If the data has been incorrectly labeled, the model cannot make accurate predictions.

PROPOSED SYSTEM:

- The proposed work aims at the early detection and even the prediction of potential future mental disorder from social media data. This successful approach can be used for effectively diagnosing mental disorders of social media users without asking them to cooperate in the diagnosis process. In this work three different studies were conducted to cover all possible aspects in the field of analyzing social media posts to obtain statistical data about users' mental cases. The proposed work covers the four well-known mental disorders which are depression, anxiety, bipolar, and ADHD. The three different studies are:
 - 1) mental disorder detection from clinical data,
 - 2) mental disorder detection from non-clinical data which is a harder problem because no mental disorders are discussed or mentioned in this ordinary type of social media data, and
 - 3) mental disorder prediction from non-clinical data which is further harder than the two previously mentioned studies

Advantages:

- In the proposed system, all mental disorders which are ADHD, Anxiety, Bipolar, and Depression were considered, and this is another added value for this work, as this is the second study that aimed at detecting and predicting all mental disorders not just one or two of them.
- In the proposed system also represents the first attempt that builds and compares wide diversity of models to detect and predict mental disorder in different contexts of social media data

3.3. HARDWARE & SOFTWARE REQUIREMENTS:

HARDWARE REQUIREMENTS :

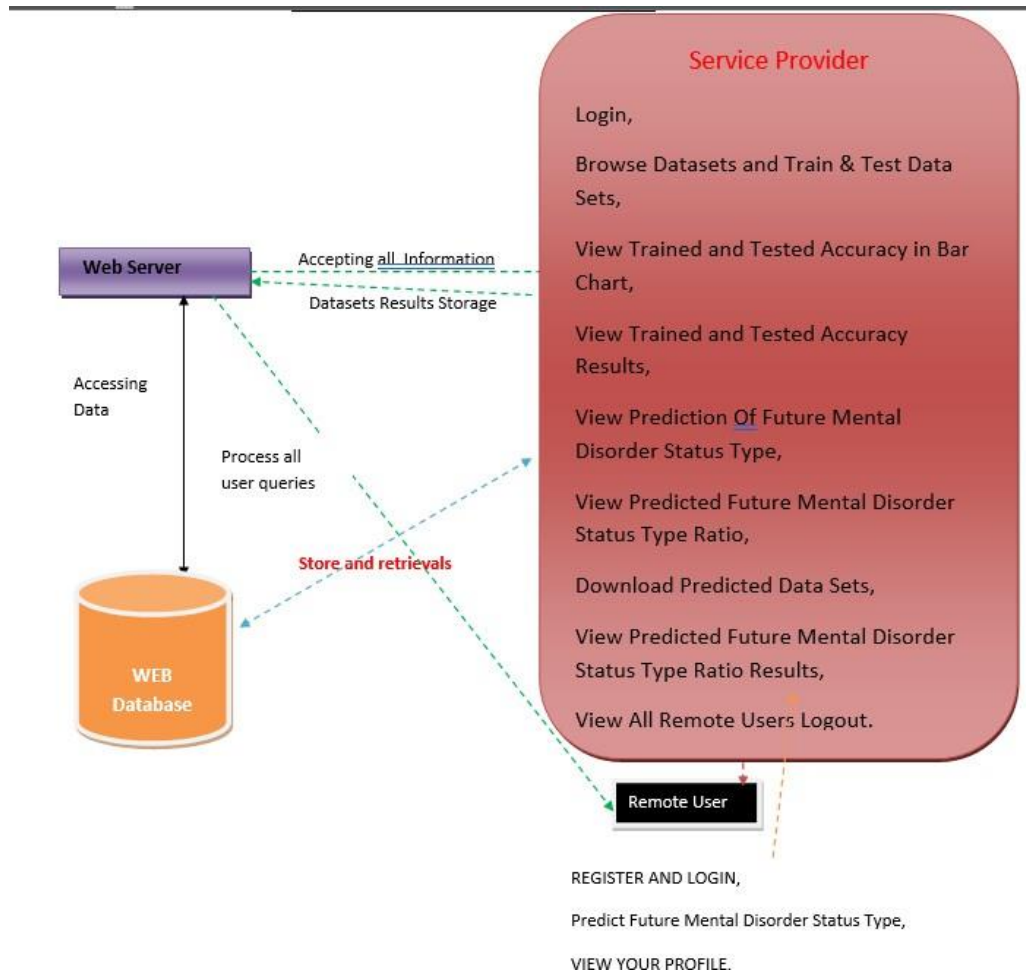
| | |
|-----------|-----------------------------|
| Processor | - Pentium –IV |
| RAM | - 4 GB (min) |
| Hard Disk | - 20 GB |
| Key Board | - Standard Windows Keyboard |
| Mouse | - Two or Three Button Mouse |
| Monitor | - SVGA |

SOFTWARE REQUIREMENTS:

| | |
|------------------|--------------------------|
| Operating system | : Windows 7 Ultimate.+ |
| Coding Language | : Python. |
| Front-End | : Python. |
| Back-End | : Django-ORM |
| Designing | : Html, css, javascript. |
| Data Base | : MySQL (WAMP Server). |

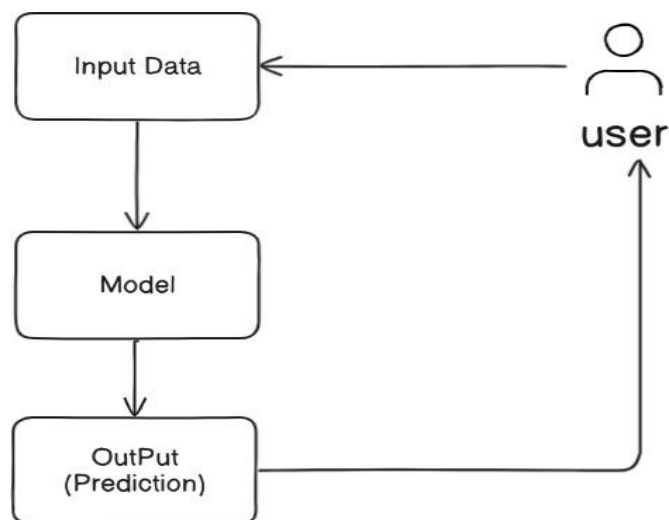
4. SYSTEM DESIGN

SYSTEM ARCHITECTURE:

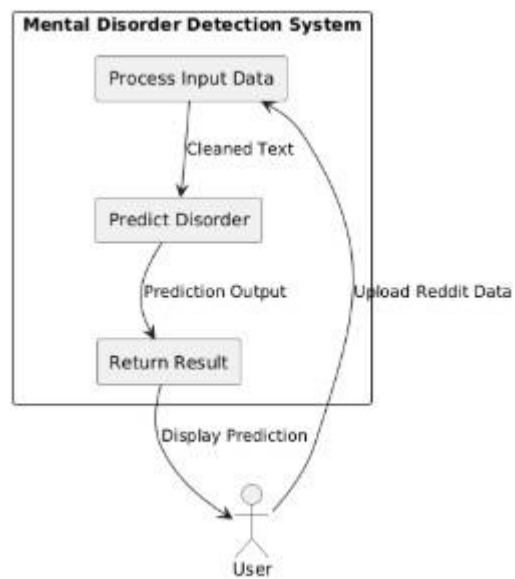


DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



Data Flow Diagram Level 0



Data Flow Diagram Level 1

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

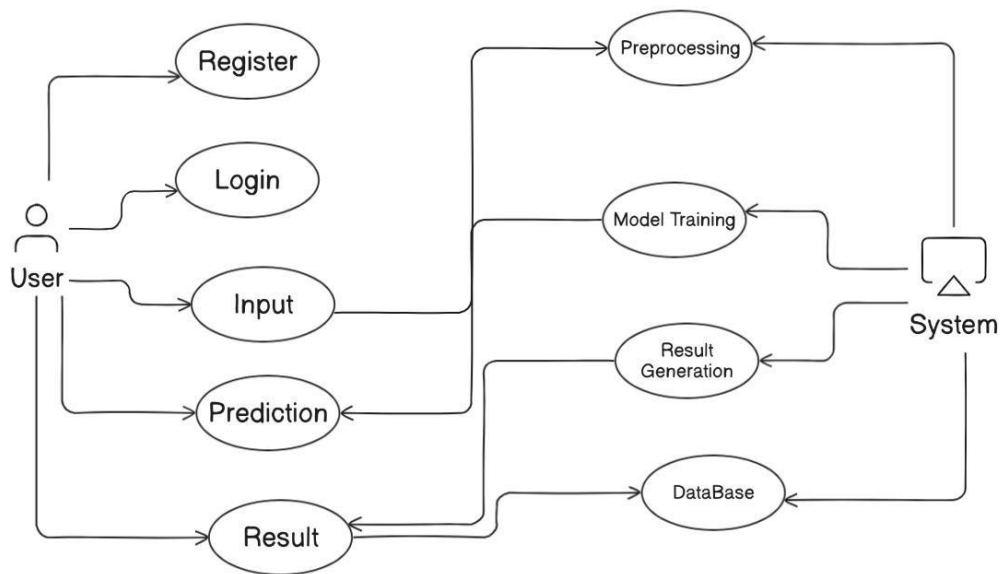
1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms.

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



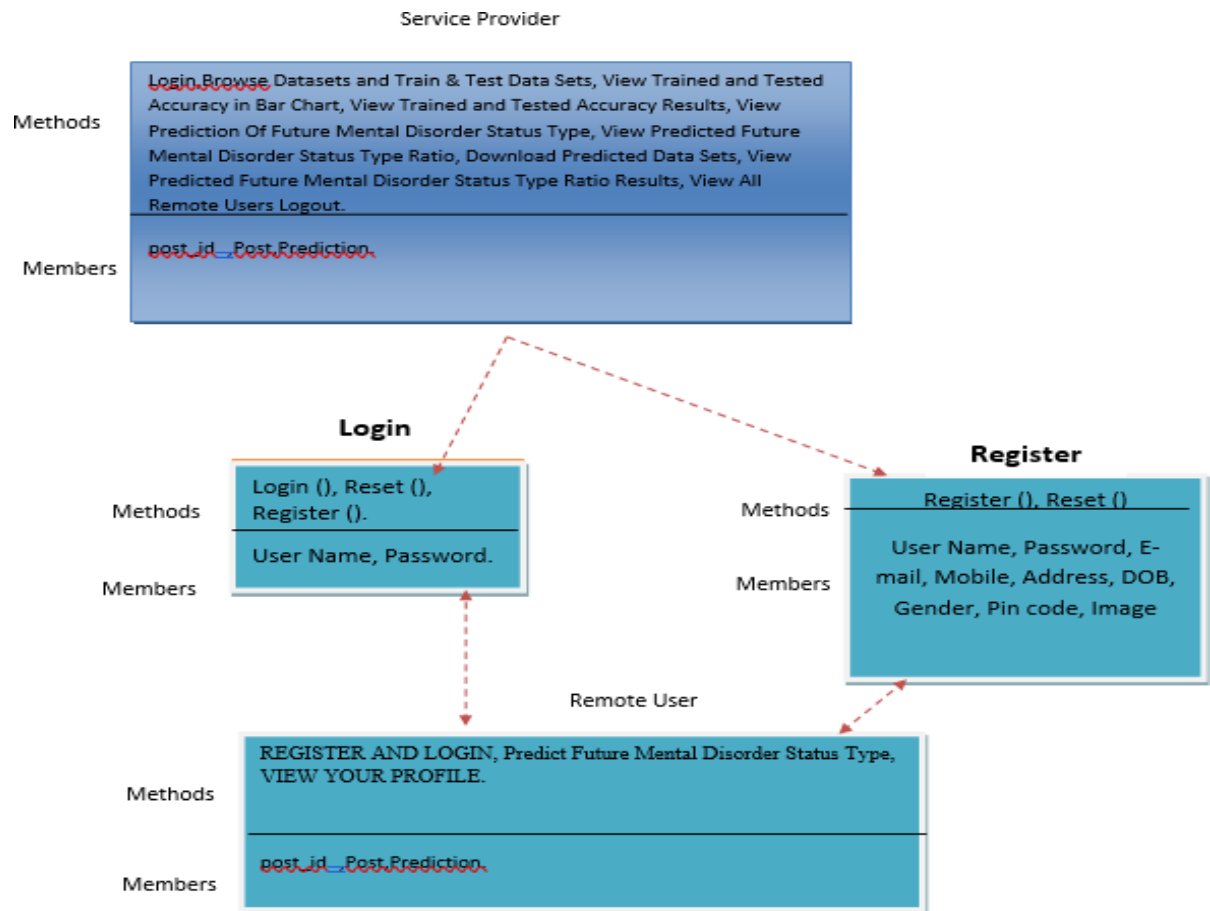
Use Case Diagram

CLASS DIAGRAM:

UML class diagrams model static class relationships that represent the fundamental architecture of the system. Note that these diagrams describe the relationships between classes, not those between specific objects instantiated from those classes. Thus the diagram applies to all the objects in the system.

A class diagram consists of the following features:

- **Classes:** These titled boxes represent the classes in the system and contain information about the name of the class, fields, methods and access specifies. Abstract roles of the Class in the system can also be indicated.
- **Interfaces:** These titled boxes represent interfaces in the system and contain information about the name of the interface and its methods. Relationship Lines that model the relationships between classes and interfaces in the system.
- **Dependency:** A dotted line with an open arrowhead that shows one entity depends on the behavior of another entity. Typical usages are to represent that one class instantiates another or that it uses the other as an input parameter
- **Aggregation:** Represented by an association line with a hollow diamond at the tail end. An aggregation models the notion that one object uses another object without "owning" it and thus is not responsible for its creation or destruction.
- **Inheritance:** A solid line with a solid arrowhead that points from a sub-class to a super class or from a sub-interface to its super-interface.
- **Implementation:** A dotted line with a solid arrowhead that points from a class to the interface that it implement.
- **Composition:** Represented by an association line with a solid diamond at the tail end. A composition models the notion of one object "owning" another and thus being responsible for the creation and destruction of another object.



Class Diagram

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behaviour of a system.

Elements of sequence diagram

The sequence diagram is an element that is used primarily to showcase the interaction that occurs between multiple objects. This interaction will be shown over certain period of time. Because of this, the first symbol that is used is one that symbolizes the object.

Lifeline

A lifeline will generally be generated, and it is a dashed line that sits vertically, and the top will be in the form of a rectangle. This rectangle is used to indicate both the instance and the class. If the lifeline must be used to denote an object, it will be underlined.

Messages

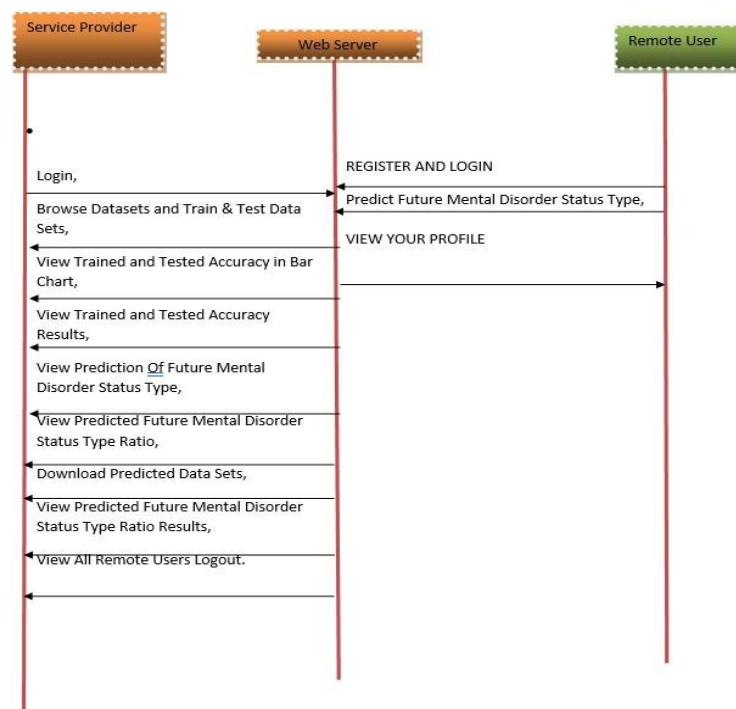
To showcase an interaction, messages will be used. These messages will come in the form of horizontal arrows, and the messages should be written on top of the arrows. If the arrow has a full head, and it's solid, it will be called a synchronous call. If the solid arrow has a stick head, it will be an asynchronous call. Stick heads with dash arrows are used to represent return messages.

Objects

Objects will also be given the ability to call methods upon themselves, and they can add net activation boxes. Because of this, they can communicate with others to show multiple levels of processing. Whenever an object is eradicated or erased from memory, the "X" will be drawn at the lifeline's top, and the dash line will not be drawn beneath it. This will often occur as a result of a message. If a message is sent from the outside of the diagram, it can be used to define a message that comes from a circle that is filled in. Within a UML based model, a Super step is a collection of steps which result from outside stimuli.

Steps to Create a Sequence Diagram

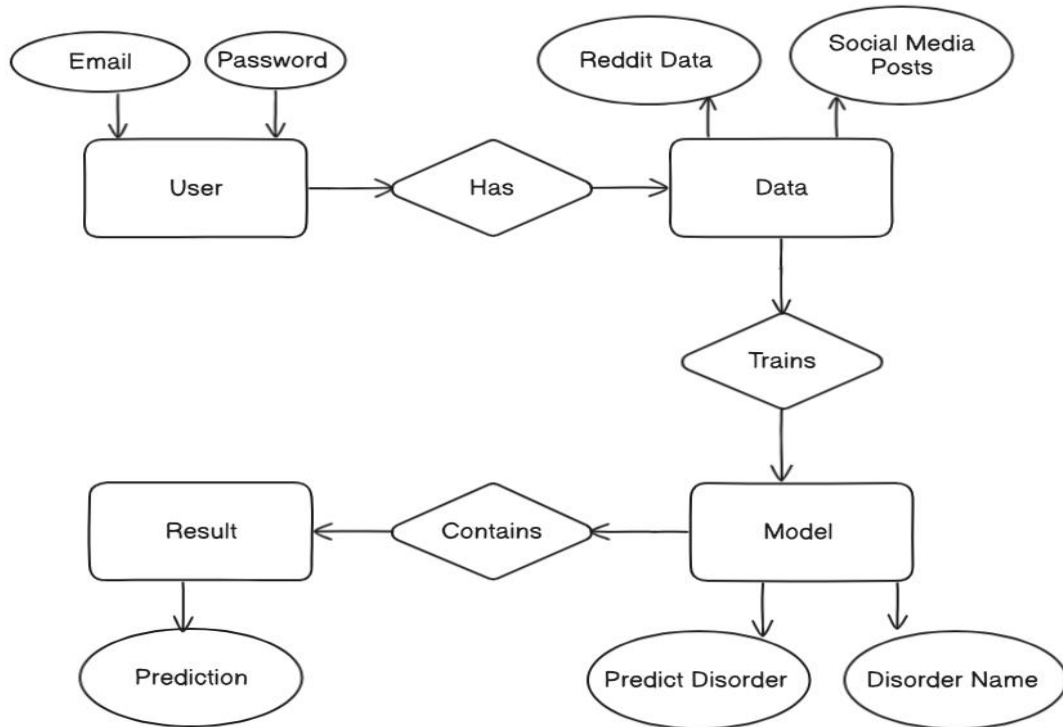
- Set the context for the interaction, whether it is a system, subsystem, operation or class.
- Set the stage for the interaction by identifying which objects play a role in interaction.
- Set the lifetime for each object.
- Start with the message that initiates the interaction.
- Visualize the nesting of messages or the points in time during actual computation.
- Specify time and space constraints, adorn each message with timing mark and attach suitable time or space constraints.
- Specify the flow of control more formally, attach pre and post conditions to each message.



Sequence Diagram

ER DIAGRAM

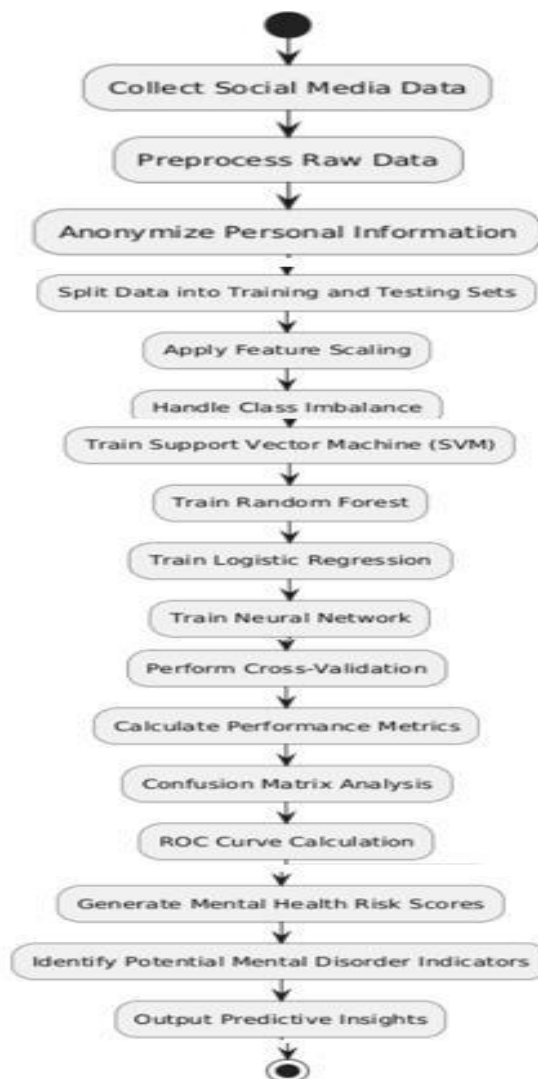
- Users provide their email and password to access the system, which utilizes social media and Reddit data.
- The data is used to train a model that predicts mental health disorders based on user-generated posts.
- The model's results provide disorder predictions, which are returned to the user as the system's output.



ER Diagram

ACTIVITY DIAGRAM:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.



Activity Diagram

5.IMPLEMENTATION

MODULES:

- ❖ Data collection (Reddit data)
- ❖ Data Preprocessing,
- ❖ Feature extraction.
- ❖ Model training (Logistic Regression, Bagging, XGBoost, Light GBM, RoBERTa, GPT),
- ❖ Evaluation
- ❖ Performance comparison
- ❖ Prediction

5.1.MODULES DESCRIPTION:

- ❖ **Data Collection and Preprocessing:** The system collects Reddit posts and comments about mental health.
- ❖ **Feature Extraction:** It cleans and processes the text using techniques like TF-IDF or word embeddings.
- ❖ **Model Training:** Various models, including Logistic Regression and large language models like ROBERT and GPT, are trained on the data.
- ❖ **Model Evaluation:** The models are evaluated using accuracy, precision, recall, and F1-score.
- ❖ **Performance Comparison:** compares the effectiveness of different models across clinical and non-clinical datasets, highlighting the best-performing models for each disorder.
- ❖ **Prediction Module:** The best models are used to predict mental health disorders from new Reddit posts.

5.2.SOFTWARE ENVIRONMENT

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games..

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include –

- **Easy-to-learn –Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org>.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.
- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
Python2.4.3(#1,Nov112010,13:34:43)
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!")**; However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active. Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file –

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods

| Sr.No | Methods & Description |
|-------|---|
| 1 | GET Sends data in unencrypted form to the server. Most common method. |
| 2 | HEAD Same as GET, but without response body |
| 3 | POST Used to send HTML form data to server. Data received by POST method is not cached by server. |
| 4 | PUT Replaces all current representations of the target resource with the uploaded content. |
| 5 | DELETE Removes all current representations of the target resource given by a URL |

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>

</form>

</body>

</html>
```

Now enter the following script in Python shell.

```
from flask import Flask, redirect, url_for, request

app=Flask(__name__)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name

@app.route('/login',methods=['POST','GET'])

def login():

if request.method=='POST':

user=request.form['nm']

return redirect(url_for('success',name= user))
```

```
else:

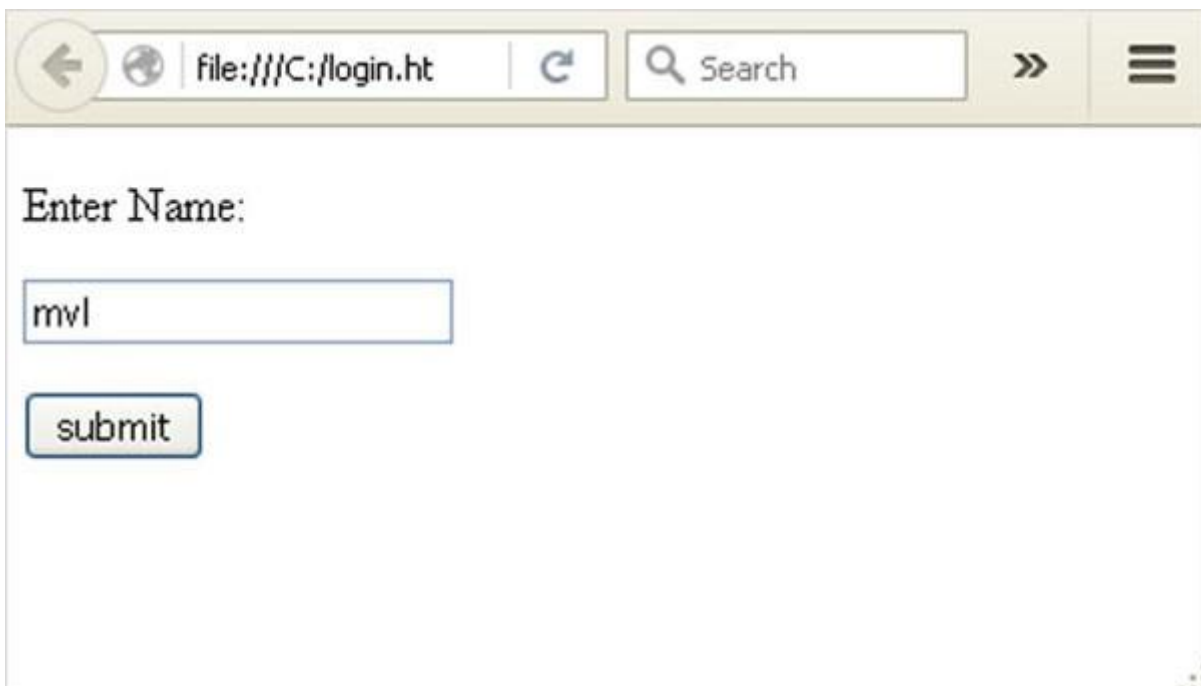
user=request.args.get('nm')

return redirect(url_for('success',name= user))

if __name__=='__main__':

app.run(debug =True)
```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

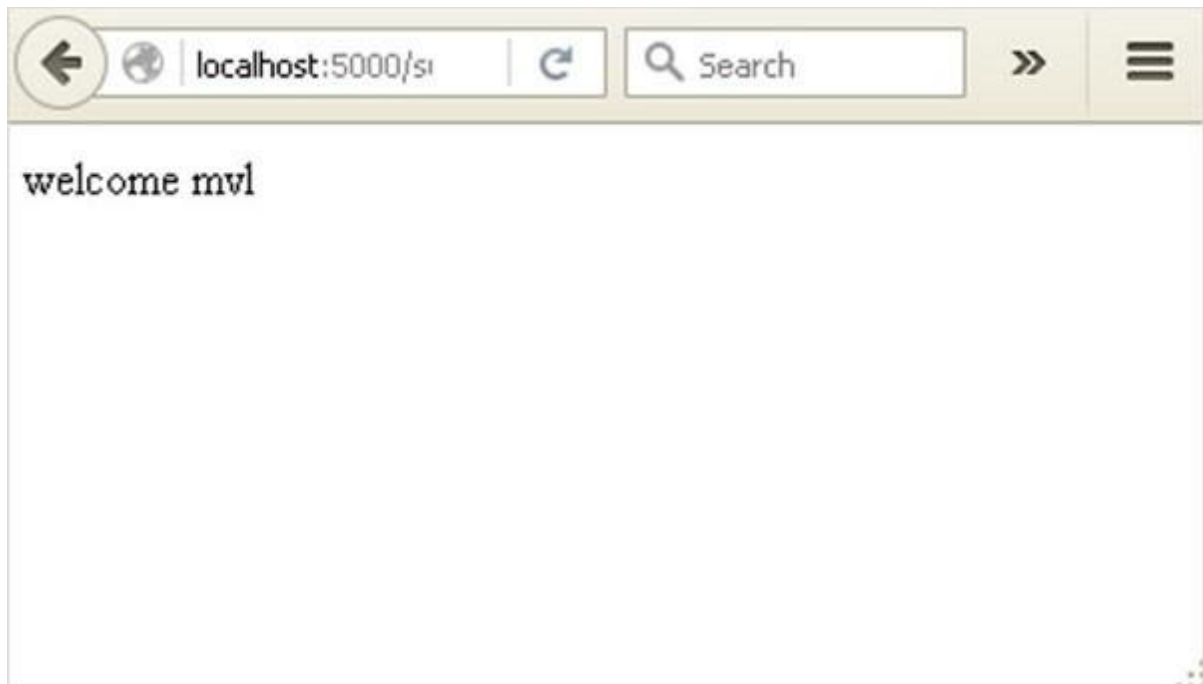
A screenshot of a web browser window. The address bar shows the file path 'file:///C:/login.ht'. The main content area displays a simple login form. It starts with the text 'Enter Name:' followed by a text input field. The input field contains the text 'mvl'. Below the input field is a button labeled 'submit'.

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by –

```
user = request.form['nm']
```

It is passed to **‘/success’** URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to '**GET**' in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by –

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to '/success' URL as before.

What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

Python Quick start

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
```

```
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users*Your Name*>python myfile

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
```

```
print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

```
#This is a comment.
```

```
print("Hello, World!")
```

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

6. ALGORITHMS

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C_1, C_2, \dots, C_k is as follows:

Step 1 : If all the objects in S belong to the same class, for example C_i , the decision tree for S consists of a leaf labeled with this class

Step 2 : Otherwise, let T be some test with possible outcomes O_1, O_2, \dots, O_n . Each object in S has one outcome for T so the test partitions S into subsets S_1, S_2, \dots, S_n where each object in S_i has outcome O_i for T . T becomes the root of the decision tree and for each outcome O_i we build a subsidiary decision tree by invoking the same procedure recursively on the set S_i .

Gradient boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

K-Nearest Neighbors (KNN)

- Simple, but a very powerful classification algorithm
- Classifies based on a similarity measure
- Non-parametric
- Lazy learning
- Does not “learn” until the test example is given
- Whenever we have a new data to classify, we find its K-nearest neighbors from the training data

Example

- Training dataset consists of k-closest examples in feature space
- Feature space means, space with categorization variables (non-metric variables)
- Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset

Logistic regression Classifiers

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name logistic regression is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name multinomial logistic regression is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar.

Logistic regression competes with discriminant analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does.

Comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values and provides ROC curves to help determine the best cutoff point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

Naïve Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b and RapidMiner 4.6.0). We try above all to understand the obtained results.

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

SVM

In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed (iid)* training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point x and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to *genetic algorithms (GAs)* or *perceptrons*, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes' meeting this requirement.

7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub- assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS:

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and

responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8. RESULT

- ❖ **Linguistic and Behavioural Markers:** AI models analyze linguistic patterns, behavioural changes in social media activity, and temporal trends to identify potential indicators of mental distress.
- ❖ **Language and Platform Consistency:** Some studies indicate that AI model performance can be consistent across different languages and social media platforms.
- ❖ **Varying Accuracy Across Crisis Types:** The accuracy of detection can vary depending on the specific type of mental health crisis, with suicidal ideation sometimes showing higher detection rates compared to anxiety crises.
- ❖ **Importance of Hybrid Models:** Combining different deep learning architectures like CNNs and LSTMs can improve accuracy by capturing both local and long-range dependencies in the text data.
- ❖ **Data Augmentation:** Techniques to increase the diversity of training data are used to improve the robustness and accuracy of the models.

9. CONCLUSION AND FUTURE WORK

CONCLUSION

The proposed work aims at the early detection and even the prediction of potential future mental disorder from social media data. This successful approach can be used for effectively diagnosing mental disorders of social media users without asking them to cooperate in the diagnosis process. The successful diagnosis then can be further used to give advice or recommendations for early treatment and prevention of mental disorders.

In this work three different studies were conducted to cover all possible aspects in the field of analyzing social media posts to obtain statistical data about users' mental cases. The proposed work covers the four well-known mental disorders which are depression, anxiety, bipolar, and ADHD. The three different studies are: 1) mental disorder detection from clinical data, 2) mental disorder detection from non-clinical data which is a harder problem because no mental disorders are discussed or mentioned in this ordinary type of social media data, and 3) mental disorder prediction from non-clinical data which is further harder than the two previously mentioned studies.

REDDIT social media platform was used to train and evaluate the models used in this work, because it is the largest and most up-to-date publicly available social media data. With six classical machine learning classifiers, nine ensemble learning models, and four language models, this is the first study in literature that builds, trains, evaluates, and compares this large number of models to address mental disorder detection and prediction from social media data.

Large Language Models (LLMs) outperformed the classical machine learning classifiers, and the ensemble learning models in the task of detecting mental disorders from clinical data. The champion models overall the LLMs used in this task are BERT, RoBERTa, and OpenAI GPT. The champion models were chosen based on the average proved to be the best in detecting depression (see table 1). F1-score taken from the F1-scores obtained for every disorder, but some models were better than others per every disorder. BERT proved to be the best in detecting anxiety and bipolar, meanwhile ROBERT proved to be the best in detecting ADHD, and OpenAI GPT.

On the other hand, ML classifiers and ensemble learning models have outperformed Large Language Models (LLMs) in the tasks of detecting and predicting mental disorders from non-clinical data. The champion model overall the ML classifiers in both tasks is the logistic regression (LR), and the champion models overall the ensemble learning models used in the detection of already existing mental disorder are Bagging estimator, XGBoost, and LightGBM. LR and Bagging estimator are the best in detecting depression, XGBoost is the best in detecting anxiety and bipolar, and LightGBM is the best in detecting ADHD. The champion model that surpassed all the ensemble learning models used in the task of predicting future mental disorder is the Bagging estimator. LR is the best in predicting depression, Bagging estimator is the best in predicting ADHD, and both models can equally predict possible anxiety and bipolar.

As a future work a comprehensive error analysis should be done to analyze the results obtained by every model especially the champion models to be able to further enhance the results of mental disorder detection and prediction either by changing the settings and hyperparameters of the models or by changing the data pre-processing part. Another important future work suggestion is the comprehensive fine-tuning of the hyper parameters which requires a lot of time and computational power, specially that feature selection step would be very important in that case to avoid overfitting in such a large dataset. Cross validation should also be considered in the future work to validate the obtained results, validate the hyperparameters, finetune them based on validation results, and accordingly improve the testing results. Developing an end-to-end software tool which helps social media users with existing or potential mental disorder would be also a great added step to this work. Possible scenarios of the software tool could be a chatbot that helps mental disorder patients based on psychology knowledgebase. It could also be just a recommendation system that set the user aware of his/her case and give possible advice. It could be connected to psychology analysis exam to give more accurate diagnosis, and it might also be connected to contacts and data of psychiatrists recommended based on the case. To be able to deploy such a software tool, a quantization step for the LLMs would be of great added value to shrink the size of the models and accordingly make data handling easier.

10. APPENDIX

10.1 SAMPLECODE:

Remort_User.PY

```
from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
import numpy as np
# Create your views here.

From Remote_User.models import
ClientRegister_Model,predict_future_mental_disorder,detection_ratio,detection_accuracy

def login(request):
    if request.method == "POST" and 'submit1' in request.POST:
        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            enter = ClientRegister_Model.objects.get(username=username,password=password)
            request.session["userid"] = enter.id
            return redirect('ViewYourProfile')
        except:
            pass
    return render(request,'RUser/login.html')

def index(request):
    return render(request, 'RUser/index.html')

def Add_DataSet_Details(request):
```

```
return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data": ""})

def Register1(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
        state = request.POST.get('state')
        city = request.POST.get('city')
        address = request.POST.get('address')
        gender = request.POST.get('gender')

        ClientRegister_Model.objects.create(username=username,email=email,password=password,
        phoneno=phoneno, country=country, state=state, city=city,address=address,gender=gender)
        obj = "Registered Successfully"
        return render(request, 'RUser/Register1.html',{'object':obj})
    else:
        return render(request,'RUser/Register1.html')

def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request,'RUser/ViewYourProfile.html',{'object':obj})

def Future_Mental_Disorder_Prediction(request):
    if request.method == "POST":
        if request.method == "POST":
            post_id=request.POST.get('post_id')
            Post=request.POST.get('Post')
            df = pd.read_csv('Datasets.csv', encoding='latin-1')
            def apply_response(Label):
                if (Label == 0):
```

```
        return 0 # Mental Disorder Not Found

    elif (Label == 1):
        return 1 # Mental Disorder Found

df['results'] = df['label'].apply(apply_response)

X = df['Post']
y = df['results']

print("Post")
print(X)
print("Results")
print(y)

cv = CountVectorizer()
# X = cv.fit_transform(X)

X = cv.fit_transform(X)

models = []
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
X_train.shape, X_test.shape, y_train.shape

print("Naive Bayes")

from sklearn.naive_bayes import MultinomialNB

NB = MultinomialNB()
NB.fit(X_train, y_train)
predict_nb = NB.predict(X_test)
```

```
naivebayes = accuracy_score(y_test, predict_nb) * 100
print("ACCURACY")
print(naivebayes)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_nb))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_nb))
models.append(('naive_bayes', NB))

# SVM Model
print("SVM")
from sklearn import svm

lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print("ACCURACY")
print(svm_acc)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))

print("Logistic Regression")

from sklearn.linear_model import LogisticRegression

reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)
y_pred = reg.predict(X_test)
print("ACCURACY")
```

```
print(accuracy_score(y_test, y_pred) * 100)
print("CLASSIFICATION REPORT")

print(classification_report(y_test, y_pred))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, y_pred))
models.append(('logistic', reg))

print("Decision Tree Classifier")
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
dtcpredict = dtc.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, dtcpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, dtcpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, dtcpredict))
models.append(('DecisionTreeClassifier', dtc))

classifier = VotingClassifier(models)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

Post1 = [Post]
vector1 = cv.transform(Post1).toarray()
predict_text = classifier.predict(vector1)

pred = str(predict_text).replace("[", "")
pred1 = pred.replace("]", "")

prediction = int(pred1)
```



```
if (prediction == 0):
    val = 'Mental Disorder Not Found'
elif (prediction == 1):
    val = 'Mental Disorder Found'

print(val)
print(pred1)

predict_future_mental_disorder.objects.create(
    post_id=post_id,
    Post=Post,
    Prediction=val)

return render(request, 'RUser/Future_Mental_Disorder_Prediction.html',{'objs': val})
return render(request, 'RUser/Future_Mental_Disorder_Prediction.html')
```

10.3 SCREEN SHOTS OF MODULES

The screenshot displays a web application interface. At the top, a browser window shows the URL '127.0.0.1:8000' and the page title 'Home Page'. The main heading of the application is 'Detection and Prediction of Future Mental Disorder From Social Media Data Using Machine Learning, Ensemble Learning, and Large Language Models'. Below this, a navigation bar contains links for 'Home', 'Remote User', and 'Service Provider'. A large image of a person covering their face with their hands is featured. The interface is divided into two main sections. The top section, titled 'PREDICTION OF FUTURE MENTAL DISORDER TYPE !!!', contains a form labeled 'ENTER DATASET DETAILS HERE !!!'. This form has two input fields: 'Enter post_id' and 'Enter Post', followed by a 'Predict' button. The bottom section, titled 'PREDICTED FUTURE MENTAL DISORDER TYPE', displays the result 'Mental Disorder Found'.

Home Page

127.0.0.1:8000

Detection and Prediction of Future Mental Disorder From Social Media Data Using Machine Learning, Ensemble Learning, and Large Language Models

Home | Remote User | Service Provider

PREDICTION OF FUTURE MENTAL DISORDER TYPE !!!

ENTER DATASET DETAILS HERE !!!

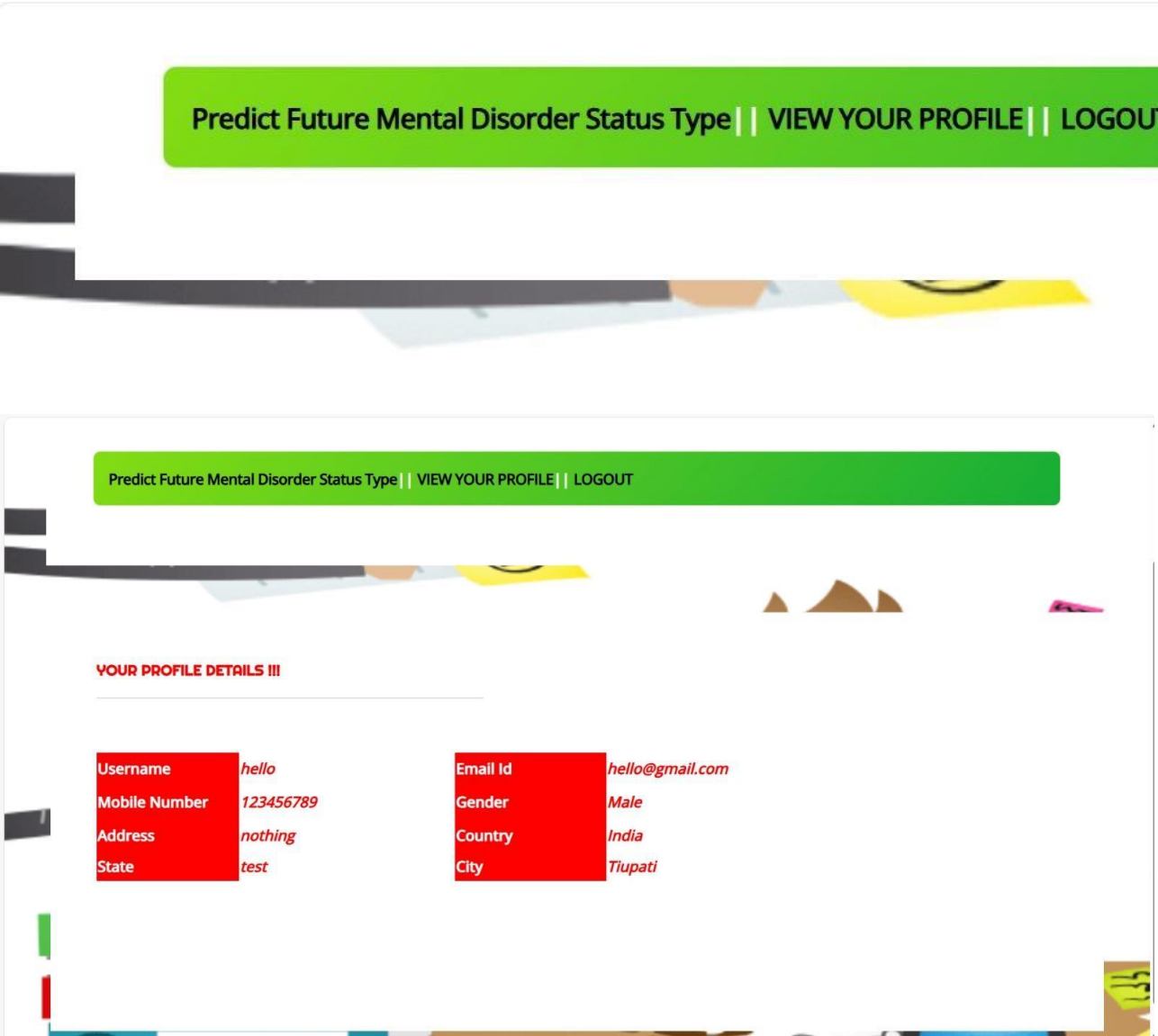
Enter post_id

Enter Post

Predict

PREDICTED FUTURE MENTAL DISORDER TYPE

Mental Disorder Found



Detection and Prediction of Future Mental Disorder From Social Media Data Using Machine Learning, Ensemble Learning, and Large Language Models

[Browse Datasets and Train & Test Data Sets](#)
[View Trained and Tested Accuracy in Bar Chart](#)
[View Trained and Tested Accuracy Results](#)

[View Prediction Of Future Mental Disorder Status Type](#)
[View Predicted Future Mental Disorder Status Type Ratio](#)
[Download Predicted Data Sets](#)

[View Predicted Future Mental Disorder Status Type Ratio Results](#)
[View All Remote Users](#)
[Logout](#)

VIEW ALL REMOTE USERS !!!

| USER NAME | EMAIL | Gender | Address | Mob No | Country | State | City |
|-----------|-----------------------|--------|-----------------------------|------------|---------|-----------|-----------|
| Govind | Govind123@gmail.com | Male | #8928,4th Cross,Vijayanagar | 9535866270 | India | Karnataka | Bangalore |
| Manjunath | tmksmanju14@gmail.com | Male | #8928,8th Cross,Vijayanagar | 9535866270 | India | Karnataka | Bangalore |
| test | test@gmail.com | Male | test | 123456789 | test | test | test |
| hello | hello@gmail.com | Male | nothing | 123456789 | India | test | Tiupati |

[Browse Datasets and Train & Test Data Sets](#)
[View Trained and Tested Accuracy in Bar Chart](#)
[View Trained and Tested Accuracy Results](#)

[View Prediction Of Future Mental Disorder Status Type](#)
[View Predicted Future Mental Disorder Status Type Ratio](#)
[Download Predicted Data Sets](#)

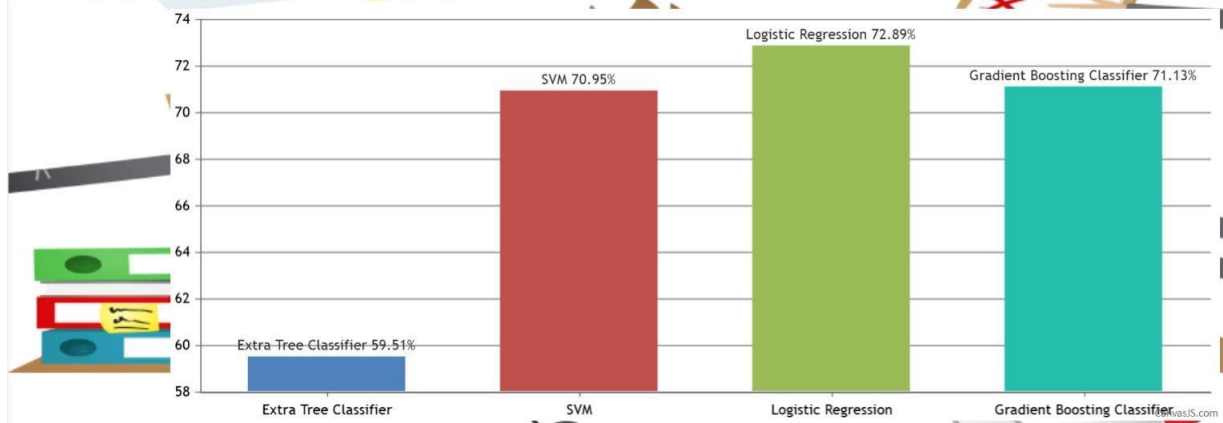
[View Predicted Future Mental Disorder Status Type Ratio Results](#)
[View All Remote Users](#)
[Logout](#)

Datasets Trained and Tested Results

| Model Type | Accuracy |
|------------------------------|-------------------|
| Extra Tree Classifier | 59.50704225352113 |
| SVM | 70.95070422535211 |
| Logistic Regression | 72.88732394366197 |
| Gradient Boosting Classifier | 71.12676056338029 |

Detection and Prediction of Future Mental Disorder From Social Media Data Using Machine Learning, Ensemble Learning, and Large Language Models

[Browse Datasets and Train & Test Data Sets](#) [View Trained and Tested Accuracy in Bar Chart](#) [View Trained and Tested Accuracy Results](#)
[View Prediction Of Future Mental Disorder Status Type](#) [View Predicted Future Mental Disorder Status Type Ratio](#) [Download Predicted Data Sets](#)
[View Predicted Future Mental Disorder Status Type Ratio Results](#) [View All Remote Users](#) [Logout](#)



11. REFERENCES

- [1] Islam, A., Hossain, M. B., Mondal, M. A. H., Ahmed, M. T., Hossain, M. A., Monir, M. U., ... & Awual, M. R. (2021). Energy challenges for a clean environment: Bangladesh's experience. *Energy Reports*, 7, 3373-3389. <https://doi.org/10.1016/j.egy.2021.05.066>.
- [2] Sanguesa, J. A., Torres-Sanz, V., Garrido, P., Martinez, F. J., & Marquez-Barja, J. M. (2021). A review on electric vehicles: Technologies and challenges. *Smart Cities*, 4(1), 372-404. <https://doi.org/10.3390/smartcities4010022>
- [3] Muratori, M., Alexander, M., Arent, D., Bazilian, M., Cazzola, P., De de, E. M., ... & Ward, J. (2021). The rise of electric vehicles—2020 status and future expectations. *Progress in Energy*, 3(2), 022002. <https://doi.org/10.1088/2516-1083/abe0ad>
- [4] Liu, Y., Wei, L., Fan, Z., Wang, X., & Li, L. (2023). Road slope estimation based on acceleration adaptive interactive multiple model algorithm for commercial vehicles. *Mechanical Systems and Signal Processing*, 184, 109733. <https://doi.org/10.1016/j.ymssp.2022.109733>
- [5] Ajanovic, A., & Haas, R. (2019). Economic and environmental prospects: Fuel cells, 19 (5), 515-529. <https://doi.org/10.1002/fuce.201800171>
- [6] Lei, Z., Qin, D., Zhao, P., Li, J., Liu, Y., & Chen, Z. (2020). A real-time blended energy management strategy of plug-in hybrid electric vehicles considering driving conditions. *Journal of Cleaner Production*, 252, 119735. <https://doi.org/10.1016/j.jclepro.2019.119735>
- [7] Kuutti, S., Bowden, R., Jin, Y., Barber, P., & Fallah, S. (2020). A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2), 712-733. <https://doi.org/10.1109/TITS.2019.2962338>
- [8] Montanaro, U., Dixit, S., Fallah, S., Dianati, M., Stevens, A., Oxtoby, D., & Mouzakitis, A. (2019). Towards connected autonomous driving: review of use-cases. *Vehicle system dynamics*, 57(6), 779-814. <https://doi.org/10.1080/00423114.2018.1492142>
- [9] Mangan, S., Wang, J., & Wu, Q. H. (2002, September). Measurement of the road gradient using an inclinometer mounted on a moving vehicle. In *Proceedings. IEEE International Symposium on Computer Aided Control System Design* (pp. 80-85). IEEE. <https://doi.org/10.1109/CACSD.2002.1036933>
- [10] Kock, P., Weller, R., Ordys, A. W., & Collier, G. (2014). Validation methods for digital road maps in predictive control. *IEEE Transactions on Intelligent Transportation Systems*, 16(1), 339-351. <https://doi.org/10.1109/TITS.2014.2332520>