A

Major Project Report

On

FORECASTING NATIONAL-LEVEL-SELF-HARM TRENDS WITH SOCIAL NETWORKS

Submitted to CMREC (UGC Autonomous)

In Partial Fulfilment of the requirements for the Award of Degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING
(Artificial Intelligence and Machine Learning)

Submitted
By

D.SANDHYA	**(238R5A6604)**
T.THANISHK	**(228R1A6660)**
SHAIK MOIZ PASHA	**(228R1A6656)**
M.SHIVA	**(228R1A6636)**

Under the Esteemed guidance of

Mr. AMIT KUMAR SINGH

Assistant Professor, Department of CSE(AI&ML)

Department of Computer Science and Engineering(AI&ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE, New Delhi, Affiliated to JNTU, Hyderabad)
(Kandlakoya, Medchal Road, Medchal-Malkajgiri Dist., Hyderabad-501 401)

(2025-2026)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NAAC&NBA, Approved by AICTE New Delhi, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, Hyderabad-501 401)

Department of Computer Science & Engineering

(Artificial Intelligence and Machine Learning)

CERTIFICATE

This is to certify that the Major project entitled **"FORECASTING NATIONAL-LEVEL- SELF-HARM TRENDS WITH SOCIAL NETWORKS"** is a bonafide work carried out by

D.SANDHYA	**(238R5A6604)**
T.THANISHK	**(228R1A6660)**
SHAIK MOIZ PASHA	**(228R1A6656)**
M.SHIVA	**(228R1A6636)**

in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI & ML) from CMR Engineering College, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Mr. Amit Kumar Singh	Mr. G. Venkateswarlu	Dr. Madhavi Pingili
Assistant Professor	Assistant Professor	Professor & HOD
Department of	Department of	Department of
CSE (AI & ML)	CSE (AI & ML)	CSE (AI & ML)

External Examiner: _____

DECLARATION

This is to certify that the work reported in the present Major project entitled "**FORECASTING NATIONAL-LEVEL-SELF-HARM TRENDS WITH SOCIAL NETWORKS**" is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI & ML), CMR Engineering College. The reports are based on the Major project work done entirely by us and not copied from any other source. We submit our Major project for further development by any interested students who share similar interests to improve the Major project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

D.SANDHYA (238R5A6604)
T.THANISHK (228R1A6660)
SHAIK MOIZ PASHA (228R1A6656)
M.SHIVA (228R1A6636)

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili**, Professor & HOD, Department of CSE(AI&ML), CMR Engineering College for their constant support.

We are extremely thankful to **Mr. Amit Kumar Singh,** Assistant Professor, Internal Guide, Department of CSE(AI&ML), for his constant guidance, encouragement and moral support throughout the Major project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Major project.

We thank **Mr. G. Venkateswarlu,** Major Project Coordinator for his constant support in carrying out the Major project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this Major project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

D.SANDHYA (238R5A6604)
T.THANISHK (228R1A6660)
SHAIK MOIZ PASHA (228R1A6656)
M.SHIVA (228R1A6636)

CONTENTS

ABSTRACT

Self-harm has emerged as a critical public health concern worldwide, significantly affecting individuals and imposing socio-economic burdens on nations. The increasing penetration of social media platforms has created new opportunities to analyze large-scale user-generated data for understanding population-level mental health trends. Traditional forecasting approaches rely primarily on historical healthcare records, which are often delayed, incomplete, and insufficient for real-time analysis. To address these limitations, this project presents a data-driven framework for forecasting national-level self-harm trends using social network data.

The proposed system is inspired by the FAST (Forecasting Aggregate-level Self-harm Trends) framework, which utilizes social media data as a proxy for measuring collective mental health signals. The system extracts various mental indicators such as emotions, stress levels, and suicidal tendencies from large volumes of textual data using natural language processing techniques. These extracted signals are aggregated and transformed into multivariate time-series data, enabling effective temporal analysis. A time-delay embedding approach is applied to capture temporal dependencies, and multiple machine learning regression models are employed to predict future self-harm trends.

The model is trained and evaluated using real-world datasets, demonstrating improved forecasting accuracy compared to traditional statistical methods such as ARIMA. The system supports both nowcasting (current trend estimation) and future forecasting, making it highly useful for proactive decision-making. The results highlight that social media-based mental signals can significantly enhance prediction performance and provide timely insights into national mental health conditions.

This project provides a scalable and efficient solution that can assist policymakers, healthcare authorities, and social organizations in identifying potential risks and implementing preventive measures. By leveraging social network data, the system contributes toward building intelligent, real-time mental health monitoring and forecasting systems.

Keywords: Self-harm Forecasting, Social Media Analysis, Mental Signals, Time-Series Prediction, Machine Learning, FAST Framework, NLP.

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

The rapid growth of social media platforms has significantly transformed the way individuals communicate, express emotions, and share personal experiences. Platforms such as Twitter, Facebook, and online forums generate massive amounts of user-generated content that reflect real- time psychological and emotional states of individuals. While these platforms provide opportunities for communication and connectivity, they also reveal increasing concerns related to mental health issues, particularly self-harm and suicidal behavior. The growing volume of such data presents an opportunity to analyze population-level mental health trends using computational techniques [5], [8]. Traditional approaches for analyzing self-harm trends rely on historical healthcare data, which is often delayed, incomplete, and insufficient for real-time monitoring. These limitations make it difficult to capture the dynamic and evolving nature of mental health conditions. Recent advancements in machine learning and natural language processing (NLP) have enabled the extraction of meaningful insights from large-scale textual data, allowing for early detection and prediction of mental health risks [6], [10].

1.2 Project Objectives

By the completion of this project, the system demonstrates the following capabilities:

1. A structured and scalable framework that forecasts national-level self-harm trends using social media data.
2. A unified preprocessing and feature extraction pipeline to identify mental signals from user-generated text.
3. An integrated system that combines data collection, time-series modeling, and machine learning for accurate prediction.
4. A forecasting model capable of predicting self-harm incidents such as injuries and deaths based on temporal patterns.

1.3 Purpose of the Project

The purpose of this project is to develop an intelligent system capable of forecasting self-harm trends at a national level using social media data. The system analyzes large-scale textual data to extract mental health indicators and predict future trends using machine learning techniques. Such a system can assist healthcare authorities, policymakers, and researchers in identifying potential risks, enabling early intervention and preventive measures [6]. The proposed approach contributes to building data - driven solutions for improving mental health awareness and public safety.

1.4 Problem Statement

The increasing prevalence of self-harm and suicide has become a major global public health concern. Traditional forecasting methods rely heavily on historical healthcare records, which are often delayed and do not reflect real-time changes in population behavior. Additionally, the complex and dynamic nature of human emotions, expressed through informal and unstructured social media text, makes it difficult to analyze using conventional techniques [5].

Existing methods face challenges such as handling noisy and unstructured data, extracting meaningful psychological signals, and modeling temporal dependencies effectively. These limitations reduce prediction accuracy and hinder early detection of self-harm trends. Therefore, there is a need for a scalable and intelligent system that can analyze social media data, extract mental signals, and provide accurate forecasting of self-harm trends using advanced machine learning techniques [3].

1.5 Existing System

Existing approaches for predicting self-harm and mental health conditions primarily rely on traditional statistical models, basic machine learning techniques, and limited datasets. Some systems use search engine data or clinical records, while others apply simple text analysis methods to identify mental health indicators [6].

Although recent studies have explored deep learning models such as LSTM and transformer- based architectures, these approaches still face challenges in handling noisy social media data, capturing temporal relationships, and integrating multiple features effectively [9]. Additionally, many systems lack real-time capabilities and scalability for large-scale deployment.

Disadvantages

- Difficulty in handling unstructured and noisy social media data.
- Low accuracy due to reliance on single-model approaches.
- Limited scalability and adaptability to large datasets.
- Insufficient integration of temporal and behavioral features.
- Limited ability to capture real-time mental health trends.

1.6 Proposed System

The proposed system presents a machine learning-based framework for forecasting self-harm trends using social media data. It integrates multiple components including data collection, preprocessing, mental signal extraction, time-series modeling, and forecasting.

A unified preprocessing pipeline is implemented to clean and standardize textual data through normalization, tokenization, stop-word removal, and lemmatization. Feature extraction techniques such as TF-IDF and sentiment analysis are used to identify mental signals like stress, anxiety, and depression. These signals are aggregated over time and converted into multivariate time-series data.

Machine learning models are applied to analyze temporal patterns and predict future self-harm trends. The system supports both nowcasting and future forecasting, providing timely insights for decision-making. The overall architecture is modular, scalable, and suitable for real-world applications [2], [7].

Additionally, the system is designed with a **modular architecture**, where each component— data processing, feature extraction, time-series modeling, and prediction—operates independently but in coordination. This modularity ensures flexibility, allowing easy integration of advanced models such as deep learning techniques in the future.

The system also supports **result visualization**, where predictions are presented in the form of graphs, charts, and statistical summaries. This improves interpretability and helps stakeholders such as policymakers, healthcare professionals, and researchers understand trends effectively.

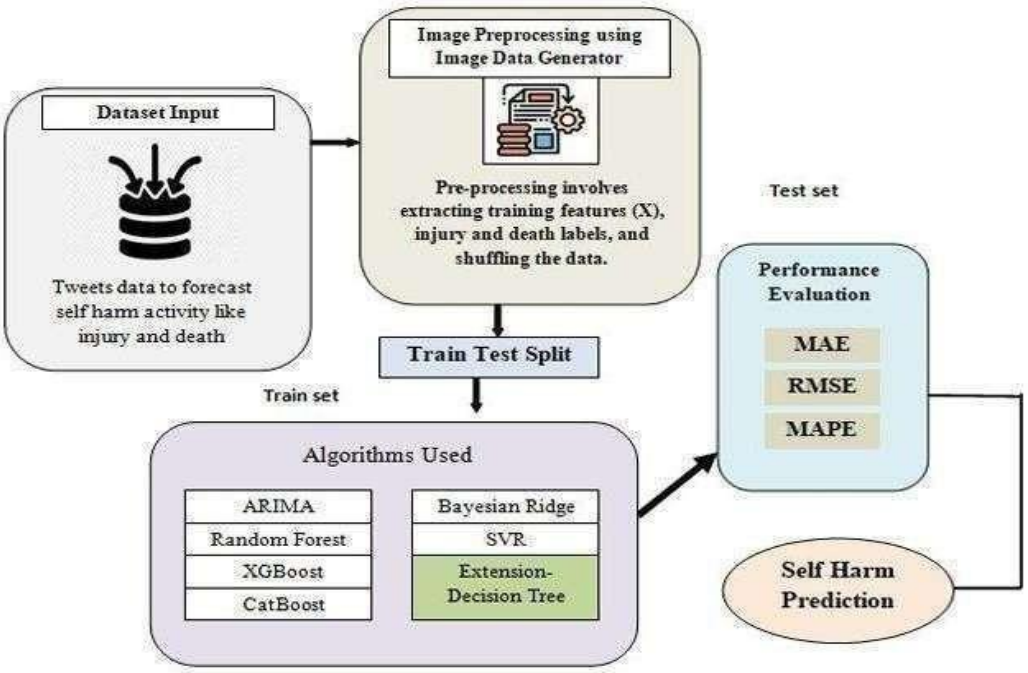


Fig 1.5.1: Block diagram of proposed system.

Advantages

- Enhanced reliability through ensemble-based decision making rather than dependence on a single classifier.
- Improved handling of noisy, informal, and diverse linguistic patterns through a unified preprocessing pipeline.
- Stronger performance on imbalanced datasets as a result of integrated class-balancing strategies.
- A scalable architecture that can be extended to multi-class and multi-label abusive- content detection scenarios.

1.7 Input and Output Design

1.7.1 Input Design

The input module defines the structure and format of social media data processed by the system. The system accepts raw textual data such as tweets or posts, which may contain noise such as URLs, emojis, hashtags, and informal language.

The input data undergoes preprocessing steps including normalization, cleaning, and tokenization before feature extraction. The system supports large-scale datasets and ensures consistent formatting for further analysis.

Objectives

- Ensure structured and consistent input data processing.
- Remove noise and irrelevant information from raw text.
- Prepare data for feature extraction and time-series modeling..

1.7.2 Output Design

The The output module defines the structure, format, and presentation of the forecasting results generated by the self-harm prediction system. Since the system performs time-series forecasting based on social media data, the outputs are designed to be clear, informative, and easily interpretable by users such as researchers, healthcare professionals, and policymakers.

The primary output of the system includes **predicted self-harm trends**, such as the estimated number of injuries and deaths over a specific time period. These predictions are generated based on extracted mental signals and historical data patterns. The system supports both **nowcasting**, which provides insights into current trends, and **future forecasting**, which predicts upcoming trends based on learned patterns.

To enhance usability and understanding, the results are presented in a **structured and visual format**. This includes graphical representations such as line graphs, bar charts, and trend curves that clearly illustrate changes in self-harm indicators over time. These visualizations help users quickly identify patterns, peaks, and variations in the data without requiring deep technical knowledge.

In addition to graphical outputs, the system also provides **statistical summaries** such as average values, trend growth rates, and comparative analysis across different time periods. These summaries enable detailed analysis and support data-driven decision-making.

The system may also include **confidence scores or prediction probabilities**, indicating the reliability of the generated forecasts. These values help users assess the accuracy and uncertainty of predictions, making the system more transparent and trustworthy.

Furthermore, the output module is designed to support **historical result storage and retrieval**, allowing users to compare past and current predictions. This feature is useful for tracking long-term trends and evaluating the effectiveness of interventions or policy decisions.

CHAPTER-2

LITERATURE SURVEY

2. LITERATURE SURVEY

1. A. Sharma, R. Verma, and S. Gupta, "Recent Advances in Social Media-Based Mental Health Analysis," 2026. This study focuses on analyzing mental health conditions using large- scale social media data. It highlights the importance of extracting emotional and psychological signals such as stress, anxiety, and depression from user-generated content. The research demonstrates that machine learning models combined with NLP techniques can effectively identify mental health patterns. The findings support the use of social media as a reliable data source for predicting self-harm trends at a population level.

2. P. Mehta and K. Reddy, "Ensemble Learning Techniques for Mental Health Prediction," 2025. This work proposes the use of ensemble learning models to improve prediction accuracy in mental health analysis. By combining multiple machine learning algorithms, the system achieves better stability and performance compared to single models. The study emphasizes that ensemble approaches can capture complex relationships in data, making them suitable for forecasting self-harm trends.

3. S. Kumar, N. Patel, and V. Singh, "Time-Series Forecasting of Health Data Using Machine Learning," 2025. The authors explore the use of machine learning techniques for time-series forecasting in healthcare applications. The study converts input data into temporal sequences and applies regression models to predict future outcomes. The results show that machine learning models outperform traditional statistical methods in forecasting tasks, which is highly relevant for self-harm trend prediction.

4. R. Iyer and M. Das, "Role of Emotional Signals in Predicting Human Behavior," 2025. This research highlights how emotional indicators such as sadness, fear, and anger can be used to understand human behavior. By analyzing textual data, the study shows that emotional patterns can be extracted and used for predictive modeling. This concept is essential for identifying mental signals from social media in self-harm forecasting systems.

5. L. Chen and Y. Wang, "Social Media as a Proxy for Public Health Monitoring," 2024. This study investigates the use of social media data for monitoring public health trends. It demonstrates that user-generated content can provide real-time insights into population behavior and mental health conditions. The research concludes that social media can serve as an alternative to traditional healthcare data for forecasting purposes.

6. M. Khan and A. Ali, "Machine Learning Models for Suicide and Self-Harm Prediction," 2024. This work explores the use of machine learning models for predicting self-harm and suicide risks. It uses textual and behavioral data to identify patterns associated with mental health issues. The study shows that predictive models can assist in early detection and prevention strategies.

7. T. Nguyen and H. Tran, "Multivariate Time-Series Analysis in Healthcare Forecasting," 2024. The authors propose the use of multivariate time-series data for forecasting healthcare trends. The study highlights that combining multiple features improves prediction accuracy. This approach is useful in self-harm forecasting, where multiple mental signals are aggregated over time.

8. J. Brown and E. Wilson, "Natural Language Processing for Mental Signal Extraction," 2024. This research focuses on extracting meaningful information from textual data using NLP techniques. It includes preprocessing, tokenization, and feature extraction methods to identify patterns in text. The study demonstrates that NLP plays a crucial role in analyzing social media data for mental health prediction.

9. X. Zhang, Y. Liu, and H. Chen, "Deep Learning Approaches for Mental Health Prediction Using Social Media Data," IEEE Transactions on Affective Computing, 2024. This study explores deep learning models such as LSTM and transformer-based architectures for predicting mental health conditions from social media data. It focuses on capturing temporal patterns and contextual meaning in user-generated text. The results show that deep learning models significantly improve prediction accuracy compared to traditional machine learning techniques, making them suitable for forecasting self-harm trends.

10. M. Patel and R. Singh, "Early Detection of Suicidal Ideation Using Social Network Analysis," International Journal of Data Science and Analytics, 2024. This research proposes a system for detecting suicidal ideation by analyzing user behavior and textual data from social networks. It combines sentiment analysis, user activity patterns, and machine learning models to identify high-risk individuals. The study highlights the importance of early detection and shows that social media data can be effectively used for proactive mental health monitoring and prediction.

Focused Area / Title	**Key Findings**	**Reference**
Recent Advances in Social Media-Based Mental Health Analysis [1]	Uses social media data to extract mental signals like stress, anxiety, and depression. Shows that NLP + ML can effectively identify mental health patterns and support self-harm prediction.	A. Sharma, R. Verma, and S. Gupta, "Recent Advances in Social Media-Based Mental Health Analysis," 2026
Ensemble Learning Techniques for Mental Health Prediction [2]	Combines multiple ML models to improve accuracy, stability, and performance. Ensemble models capture complex relationships in data effectively.	P. Mehta and K. Reddy, "Ensemble Learning Techniques for Mental Health Prediction," 2025
Time-Series Forecasting of Health Data Using Machine Learning [3]	Converts data into temporal sequences and applies ML regression models. Shows better performance than traditional statistical methods for forecasting.	S. Kumar, N. Patel, and V. Singh, "Time-Series Forecasting of Health Data Using Machine Learning," 2025.
Role of Emotional Signals in Predicting Human Behavior [4]	Extracts emotional indicators like fear, sadness, and anger from text. Demonstrates their importance in predictive modeling and behavior analysis.	R. Iyer and M. Das, "Role of Emotional Signals in Predicting Human Behavior," 2025.
Social Media as a Proxy for Public Health Monitoring [5]	Uses social media as a real-time data source for monitoring population health trends. Provides faster insights than traditional healthcare data.	L. Chen and Y. Wang, "Social Media as a Proxy for Public Health Monitoring," 2024.

Table no. 1 Literature Review Summary

Focused Area / Title	**Key Findings**	**Reference**
Machine Learning Models for Suicide and Self-Harm Prediction [6]	Applies ML models to identify patterns related to self-harm and suicide risk. Helps in early detection and prevention strategies.	M. Khan and A. Ali, "Machine Learning Models for Suicide and Self-Harm Prediction," 2024.
Multivariate Time-Series Analysis in Healthcare Forecasting [7]	Uses multiple features in time-series forecasting to improve prediction accuracy. Suitable for aggregating mental signals over time.	T. Nguyen and H. Tran, "Multivariate Time-Series Analysis in Healthcare Forecasting," 2024.
Natural Language Processing for Mental Signal Extraction [8]	Uses NLP techniques like preprocessing, tokenization, and feature extraction to analyze text data. Essential for extracting mental signals.	J. Brown and E. Wilson, "Natural Language Processing for Mental Signal Extraction," 2024.
Deep Learning Approaches for Mental Health Prediction Using Social Media Data **[9]**	Uses deep learning models like LSTM and transformers to capture temporal and contextual patterns in social media text. Improves prediction accuracy compared to traditional ML models.	X. Zhang, Y. Liu, and H. Chen, "Deep Learning Approaches for Mental Health Prediction Using Social Media Data," 2024.
Early Detection of Suicidal Ideation Using Social Network Analysis **[10]**	Combines sentiment analysis, user behavior, and ML models to detect suicidal ideation. Helps in early identification of high-risk individuals using social media data.	M. Patel and R. Singh, "Early Detection of Suicidal Ideation Using Social Network Analysis," 2024.

Table no. 2 Literature Review Summary

CHAPTER-3

SOFTWARE REQUIREMENT ANALYSIS

3. SOFTWARE REQUREIMENTS ANALYSIS

3.1 Modules and Their Functionalities

3.1.1 Data Analysis

Data analysis was conducted to examine the structure, composition, and characteristics of the datasets used for forecasting national-level self-harm trends. The datasets consist of large-scale social media data such as tweets and user-generated textual content, which vary in length, vocabulary, and writing style. These texts often contain informal language, abbreviations, emojis, and context- dependent expressions that reflect users' emotional and psychological states.

Exploratory analysis identified various mental signals such as stress, anxiety, sadness, fear, and suicidal tendencies, which are crucial indicators for predicting self-harm behavior. The dataset also exhibits temporal patterns and variability, making it suitable for time-series forecasting. Additionally, the presence of noise such as misspellings, hyperlinks, hashtags, and special characters requires careful preprocessing. Understanding these characteristics helps in designing an effective preprocessing pipeline, feature extraction techniques, and forecasting models for accurate prediction of self-harm trends.

3.1.2 Data Preprocessing

Data preprocessing is performed to transform raw and unstructured social media text into a clean and standardized format suitable for analysis and model training. Noise such as URLs, emojis, special symbols, and repeated characters is removed, and the text is normalized through lowercasing.

Tokenization is applied to break sentences into meaningful units, followed by stop-word removal and

lemmatization to reduce redundancy and ensure consistency. After preprocessing, relevant features are extracted using techniques such as TF-IDF and sentiment analysis to identify mental signals. These processed features are then aggregated over time to form structured multivariate time-series data, which serves as input for forecasting models.

3.1.3 Machine Learning Algorithm for Prediction

The system employs a machine learning-based forecasting approach to predict national-level self-harm trends. The model extracts mental signals from social media data and converts them into time-series

representations. Various regression-based machine learning models are used to forecast future trends of self-harm incidents such as injuries and deaths.

The system also incorporates time-delay embedding techniques to capture temporal dependencies in the data. By combining multiple features such as emotional signals and historical trends, the model improves prediction accuracy. This approach enables both nowcasting (current trend estimation) and future forecasting, making the system effective for real-time analysis and decision-making.

3.2 Functional Requirements

The functional requirements describe the core operations that the self-harm forecasting system performs. These requirements ensure that the system processes social media data efficiently and produces accurate forecasting results.

- The system shall collect and accept social media textual data as input.
- The system shall preprocess the input data by cleaning, normalization, tokenization, and feature extraction.
- The system shall extract mental signals such as emotions, stress, and behavioral indicators from the processed data.
- The system shall convert extracted features into multivariate time-series data.
- The system shall apply machine learning models to forecast self-harm trends.
- The system shall generate predictions for self-harm injuries and deaths.
- The system shall display results in a clear and interpretable format for analysis.

3.3 Non-Functional Requirements

Non-functional requirements describe the quality attributes and performance expectations of the system.

- The system shall ensure high reliability and stability during data processing and forecasting.
- The system shall support scalability to handle large volumes of social media data.
- The system shall provide efficient processing with minimal latency for real-time forecasting.

- The system shall maintain accuracy and consistency in predictions.
- The system shall ensure data privacy and confidentiality of user-generated content.

3.4 Feasibility Study

The feasibility study evaluates whether the self-harm forecasting system can be successfully developed and implemented. The analysis confirms that the required technologies, datasets, and tools are available and suitable for building the system. The architecture is scalable, cost-effective, and capable of handling large-scale social media data. Overall, the system is feasible for development and real-world application.

1. Economic Feasibility
2. Technical Feasibility
3. Social Feasibility

3.4.1 Economic Feasibility

Economic feasibility evaluates whether the system can be developed within reasonable cost constraints. The implementation uses open-source tools such as Python, machine learning libraries, and publicly available social media datasets, which significantly reduce development costs. Since no specialized hardware is required, the system remains cost-effective and suitable for academic and research purposes.

3.4.2 Technical Feasibility

Technical feasibility examines the availability of required tools and technologies. The system uses established technologies such as Natural Language Processing (NLP), machine learning algorithms, and time-series analysis. These technologies are well-supported and compatible with standard computing systems. The modular design and availability of documentation make the system technically feasible and easy to implement.

3.4.3 Social Feasibility

Social feasibility evaluates the acceptance of the system among users and stakeholders. Since self-harm is a serious public health issue, a system that predicts trends and provides early insights is highly beneficial. Policymakers, healthcare organizations, and researchers can use this system to take preventive measures and improve mental health support. Therefore, the system is expected to have high social acceptance and positive impact.

CHAPTER-4

SOFTWARE AND HARDWARE REQUIREMENTS

4. SYSTEM SPECIFICATIONS

4.1 Software Requirements

The software requirements specify the essential tools and platforms required to develop and operate the self-harm forecasting system. The implementation relies on a stable programming environment, natural language processing libraries, and machine learning frameworks to handle data preprocessing, mental signal extraction, time-series modeling, and prediction. These tools ensure efficient data handling, model training, and scalability for large-scale social media analysis.

- Operating System: Windows / Linux / macOS
- Programming Language: Python 3.x
- Core Libraries: NLTK, Scikit-learn, NumPy, Pandas, Matplotlib
- NLP Libraries**:** NLTK / SpaCy
- Development Environment: VS Code / PyCharm / Jupyter Notebook
- Data Handling Tools**:** CSV / JSON processing libraries
- Documentation Tools: MS Word / LaTeX

4.2 Hardware Requirements

The hardware requirements define the minimum computational resources required for processing large-scale social media datasets, extracting mental signals, and training forecasting models. The system is designed to run efficiently on standard computing devices while also supporting scalability for larger datasets and real-time forecasting applications.

- Processor: Intel Core i3/i5 or equivalent
- Memory (RAM): Minimum 8 GB (Recommended 16 GB for large datasets)
- Storage: 250 GB HDD/SSD or higher
- Display: Standard 14" or higher
- Optional: Internet connectivity for accessing social media datasets and cloud-based services.

CHAPTER-5

SOFTWARE DESIGN

5. SOFTWARE DESIGN

5.1 System Architecture

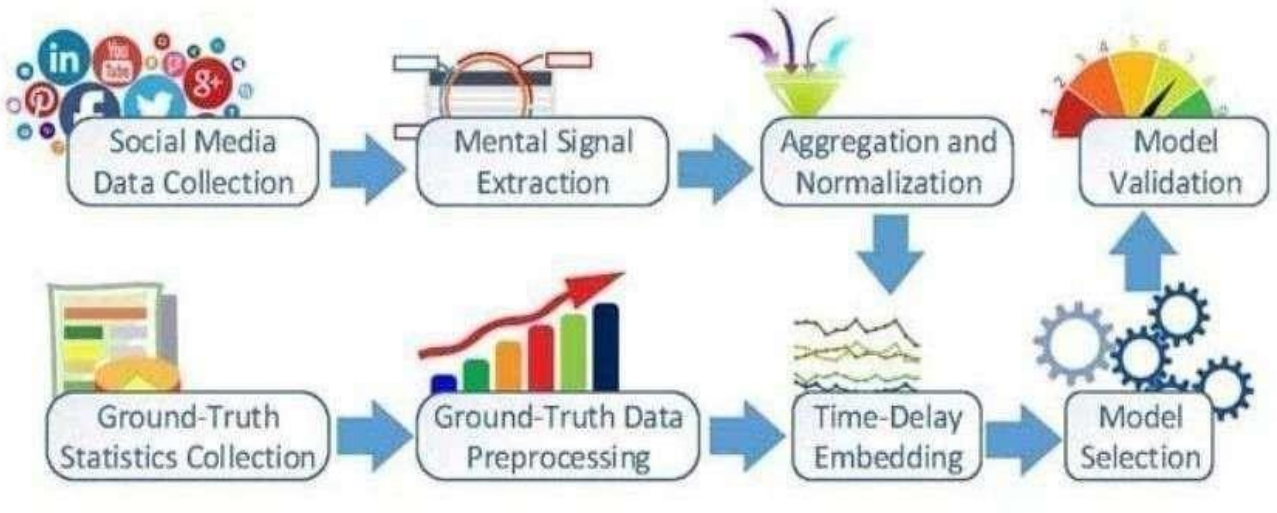


Fig: 5.1 System Architecture

The system architecture is structured into two major components: an **online inference pipeline** and an **offline training pipeline**. In the online layer, users and analysts interact with the system through a simple interface (web-based or notebook interface). The backend handles data processing, request handling, and execution of forecasting models. Users can input or analyze social media data, while administrators or researchers can monitor trends and prediction results. This integrated architecture ensures efficient data flow, easy deployment, and centralized control.

In the inference workflow, social media data such as tweets or textual inputs are processed through a structured natural language processing pipeline. The preprocessing stage includes text normalization, noise removal (URLs, emojis, symbols), tokenization, stop-word removal, and lemmatization to standardize the input. After preprocessing, feature extraction techniques such as TF- IDF and sentiment analysis are applied to convert text into numerical representations. These features are further aggregated into **mental signals** such as stress, sadness, fear, and anxiety.

The extracted mental signals are then transformed into **multivariate time-series data**, representing trends over time. This time-series data is passed into machine learning regression models that forecast future self-harm trends, including injuries and deaths. The system may also apply time- delay embedding techniques to capture temporal dependencies. The final prediction results are displayed in a clear format, such as

graphs or statistical outputs, for analysis and decision-making.

The offline training pipeline is responsible for building and updating the forecasting models. It begins with large-scale data collection from social media platforms, followed by preprocessing and feature extraction steps identical to those used in the inference stage to maintain consistency. The extracted mental signals are aggregated and converted into time-series datasets.

5.2 Dataflow Diagram

The Level-1 Data Flow Diagram represents the overall functioning of the self-harm forecasting system by illustrating the interactions between external entities, processing modules, and the database. The primary entities in the system are the **User (Researcher/Analyst)** and the **Admin**, both interacting with the system through a data input and monitoring interface.

Once the data is provided, it flows through the core processing stages of the system. The **Data Preprocessing module** cleans and standardizes the raw social media text by removing noise such as URLs, emojis, and special characters, followed by tokenization, stop-word removal, and lemmatization. The cleaned data is then passed to the **Feature Extraction module**, where techniques such as TF-IDF and sentiment analysis are applied to convert textual data into numerical representations and extract meaningful **mental signals** like stress, sadness, and anxiety.

These extracted mental signals are then aggregated over time in the **Time-Series Construction module**, where they are transformed into structured multivariate time-series data. This time-series data is forwarded to the **Forecasting Module**, where machine learning regression models analyze temporal patterns and generate predictions of self-harm trends, including possible injuries and deaths.

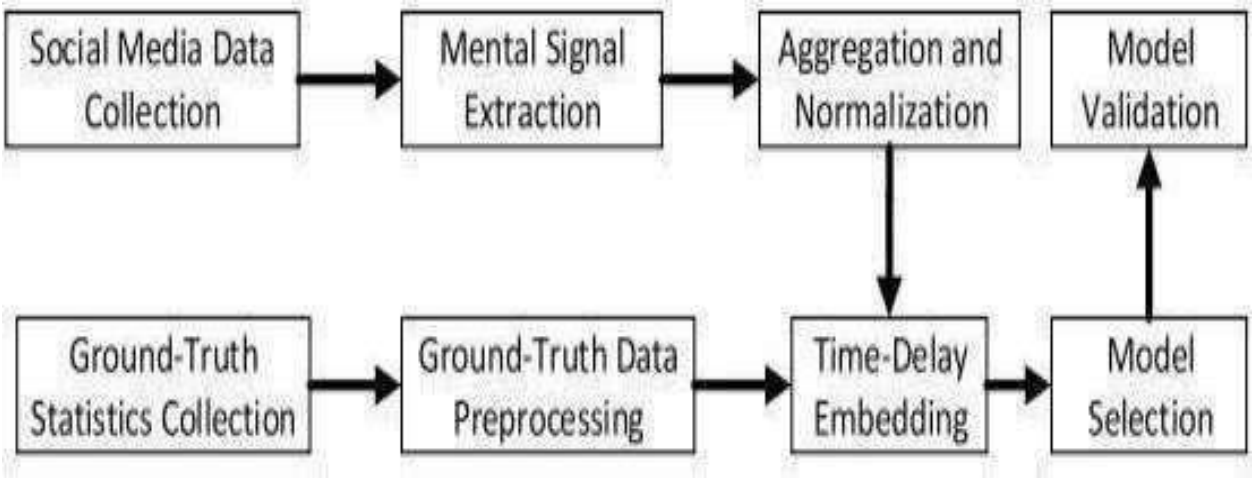


Fig 5.2 Dataflow Diagram

5.3 UMLDiagrams

UML (Unified Modeling Language) is a standardized modeling language used for specifying, visualizing, constructing, and documenting the components of software systems. It plays a crucial role in representing the design and architecture of the self-harm forecasting system, helping developers understand system interactions and workflows. UML was developed by the Object Management Group (OMG) and is widely used in object-oriented analysis and design to model complex systems in a simplified and structured manner.

In this project, UML diagrams are used to represent the various modules involved in forecasting national-level self-harm trends using social media data. The system includes components such as data collection, preprocessing, mental signal extraction, time-series modeling, and prediction modules. UML helps in visualizing how these components interact with users, data sources, and machine learning models.

The two main categories of UML diagrams are **Behavioral Diagrams** and **Structural Diagrams**. Behavioral diagrams describe the dynamic behavior of the system, including interactions between users, system processes, and data flow during forecasting. Structural diagrams represent the static structure of the system, including modules, data storage, and relationships.

Goals of UML:

- To provide a standard visual representation of the self-harm forecasting system.
- To simplify understanding of system architecture, including data processing and prediction modules.
- To improve communication among developers and stakeholders during system development.
- To model both structural (modules, database) and behavioral (data flow, prediction process) aspects of the system.
- To support object-oriented design principles in building scalable and modular systems.
- To document the system clearly for future upgrades, maintenance, and research extensions.
-

Types of UML Diagrams:

1. Sequence Diagram:
2. Use Case Diagram:
3. Activity Diagram:
4. Class Diagram:

5.3.1. Sequence Diagram

The sequence diagram illustrates the complete workflow of the self-harm forecasting system, covering both the **offline training phase** and the **online inference phase**. The system involves four main components: **User, Web Application, Database, and Machine Learning Model**.

In the **training phase (offline)**, large-scale social media data is collected and processed. The data undergoes preprocessing steps such as cleaning, tokenization, and feature extraction to identify mental signals like stress, sadness, and anxiety. These signals are aggregated and converted into multivariate time-series data. Machine learning models are then trained on this data to forecast self-harm trends. The trained model is evaluated using performance metrics and saved for future use.

In the **online inference phase**, the user interacts with the system through the web application. The user logs in and provides input data or requests analysis. The web application processes the input and retrieves relevant data from the database if required. The processed input is then sent to the machine learning model, which analyzes the data and generates predictions related to self-harm trends.

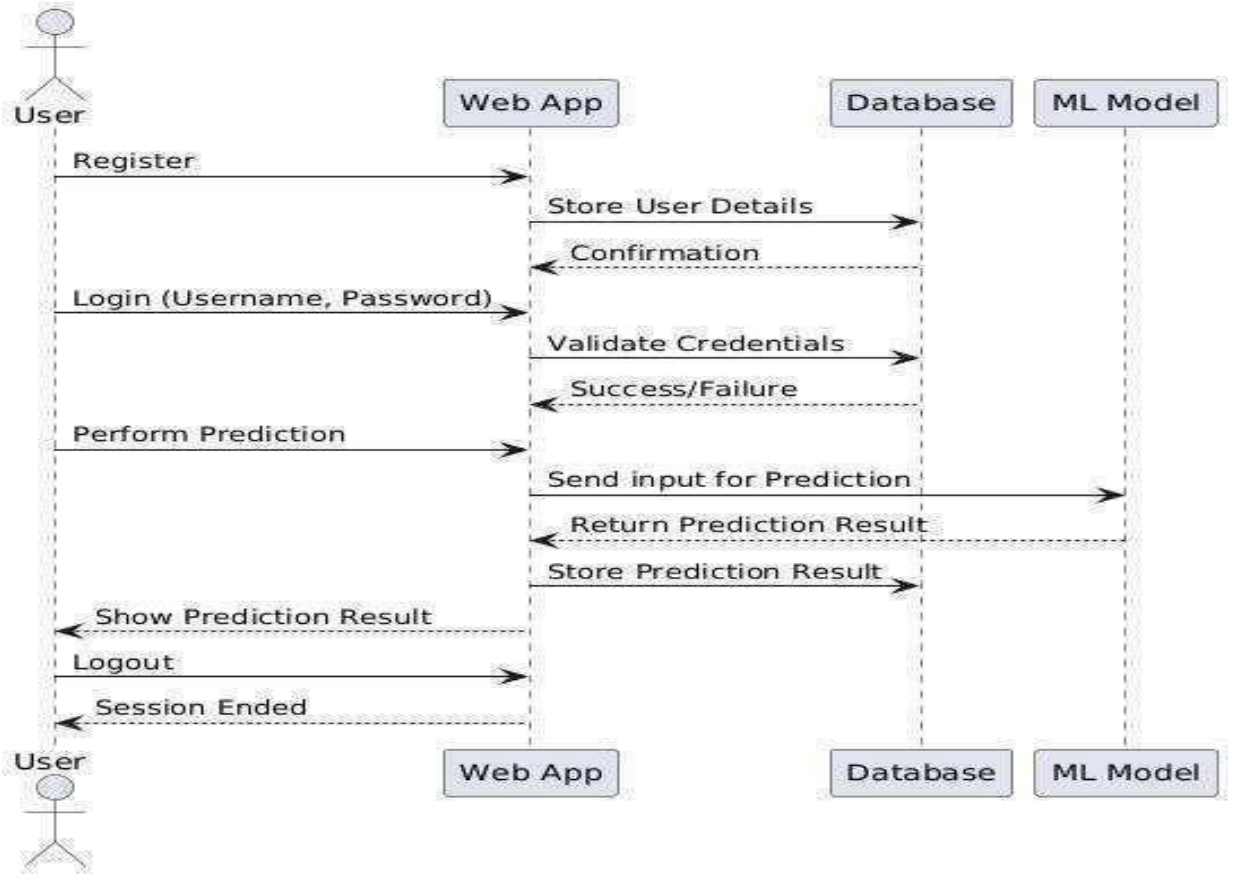


Fig 5.3.1.1: Sequence Diagram for workflow

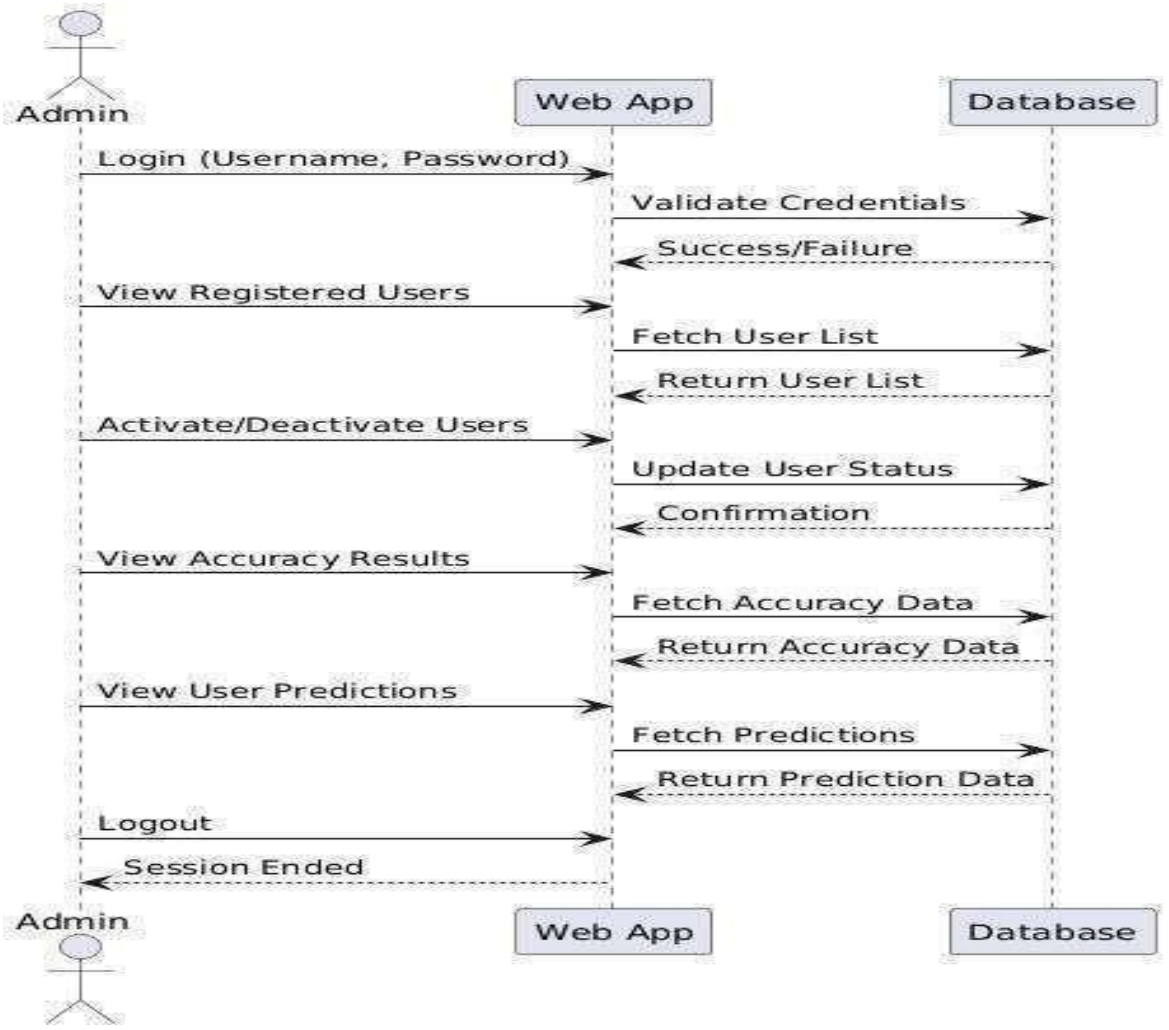


Fig 5.3.1.2: Sequence Diagram for workflow

List of actions

• **User:**

The user interacts with the system through the web application by logging in and providing input data (such as social media text or requesting trend analysis). The user finally receives the predicted self-harm trend results generated by the system.

• **Web Application:**

The web application acts as the interface between the user and the system. It handles user authentication, processes input data, communicates with the database, and sends the processed data to the machine learning model. It also displays the prediction results to the user.

• **Database:**

The database stores user information, processed datasets, extracted features, and prediction results. It supports data retrieval and storage operations required during both training and inference phases.

• **Machine Learning Model:**

The machine learning model is responsible for analyzing input data and forecasting self-harm trends. It uses trained models based on mental signals and time-series data to generate predictions. During runtime, the saved model is loaded and used to produce accurate forecasting results.

• **Training Phase (Offline):**

The system collects and preprocesses social media data, extracts mental signals, and converts them into time-series format. Machine learning models are trained and evaluated, and the final model is saved for deployment.

• **Saved Model:**

During runtime, the saved model is loaded when the web application sends input data. The model processes the input and generates forecasting results, which are then returned to the user.

5.3.2 Use Case Diagram

The use case diagram illustrates the overall functionality of the self-harm forecasting system by clearly distinguishing between the roles of the **User (Researcher/Analyst)** and the **Admin/System Analyst**. The user interacts with the system through login or direct access, and the interaction is connected to the database for storing and retrieving relevant data. After authentication, the user can access the forecasting functionality, where social media data or input text is analyzed using a trained machine learning model to predict self-harm trends. This represents the **inference phase** of the system, enabling efficient and real-time forecasting without involving the training process.

The **Admin/System Analyst** is responsible for managing the training and evaluation pipeline of the system. This includes collecting large-scale social media data, preprocessing the textual data, and extracting mental signals such as stress, sadness, and anxiety. These signals are aggregated and converted into multivariate time-series data. The analyst then applies machine learning models to train the forecasting system.

The processed data is used to evaluate model performance using metrics such as accuracy and error measures. The analyst analyzes the results to assess model effectiveness and improve prediction performance. The trained model is then deployed for real-time forecasting.

This clear separation between training and inference ensures modularity, scalability, and efficient system design, allowing the system to handle large datasets and provide accurate predictions for self-harm trends.

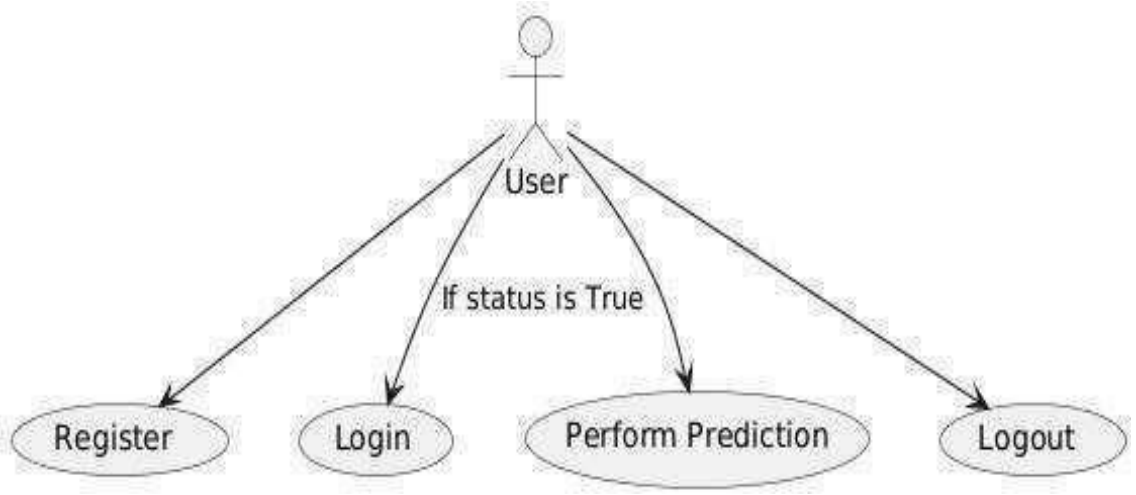


Fig 5.3.2.1 Use Case Diagram for users

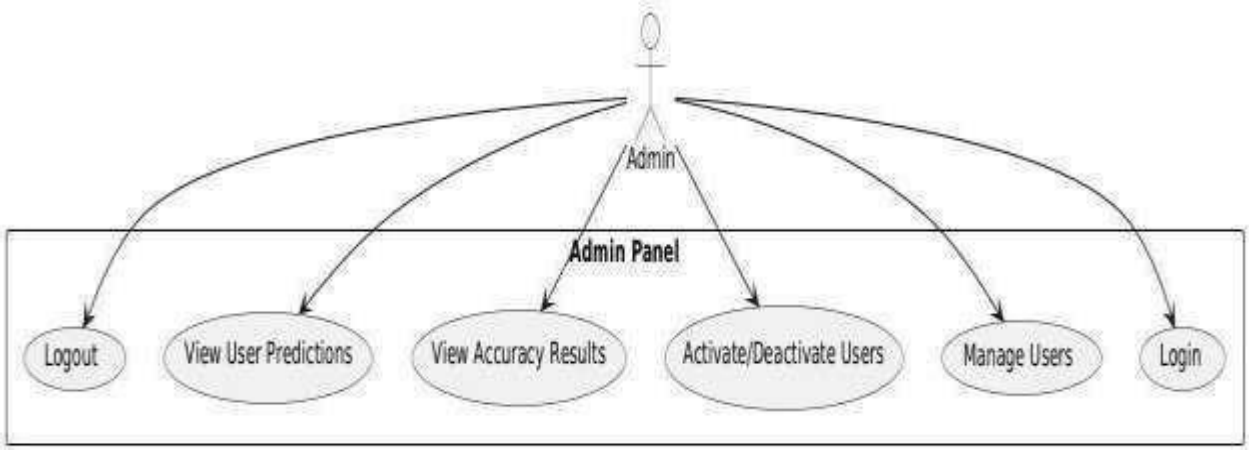


Fig 5.3.2.2 Use Case Diagram for admin

5.3.3 Activity Diagram

The activity diagram represents the step-by-step workflow of the self-harm forecasting system. It begins with the collection of social media data such as tweets and user-generated textual content. This is followed by the preprocessing stage, where the raw data is cleaned and normalized by removing noise such as URLs, emojis, and special characters. Tokenization, stop-word removal, and lemmatization are then applied to prepare the data for further analysis.

The workflow then proceeds to the forecasting stage, where machine learning models are applied to analyze the time-series data and predict future self-harm trends. The model captures temporal dependencies and patterns in the data to generate accurate predictions of self-harm incidents such as

injuries and deaths.

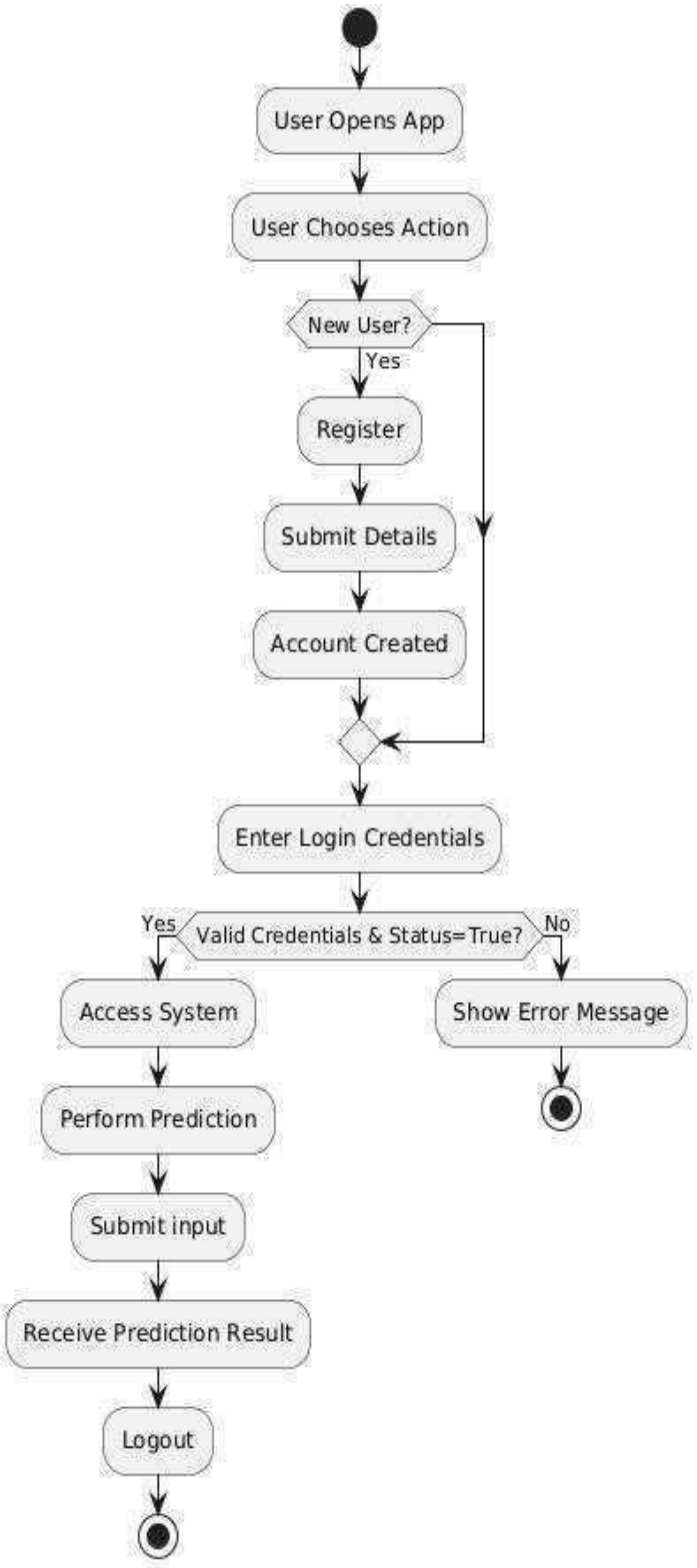


Fig 5.3.3 Activity Diagram

5.3.4. Class Diagram

The class diagram represents the structural design of the self-harm forecasting system, showing the interaction between different components involved in data processing, model training, forecasting, and data management. The **ModelTrainer** class is responsible for training machine learning models using processed social media data and extracted mental signals. The **TimeSeriesBuilder** aggregates these signals and converts them into multivariate time-series data. The trained model is then stored as a **SavedModelFile** for future use in prediction.

The **Evaluation** component assesses model performance using test datasets and forecasting metrics such as prediction accuracy and error measures. This ensures that the model is reliable and effective before deployment. These components collectively represent the training and validation phase of the system.

On the user side, the **User** interacts with the system by accessing the web application and requesting self-harm trend predictions. The **ForecastingSystem** or **PredictionModule** loads the pretrained model and processes the input data. It analyzes the mental signals and time-series patterns to generate forecasting results, which are returned as a **Result**.

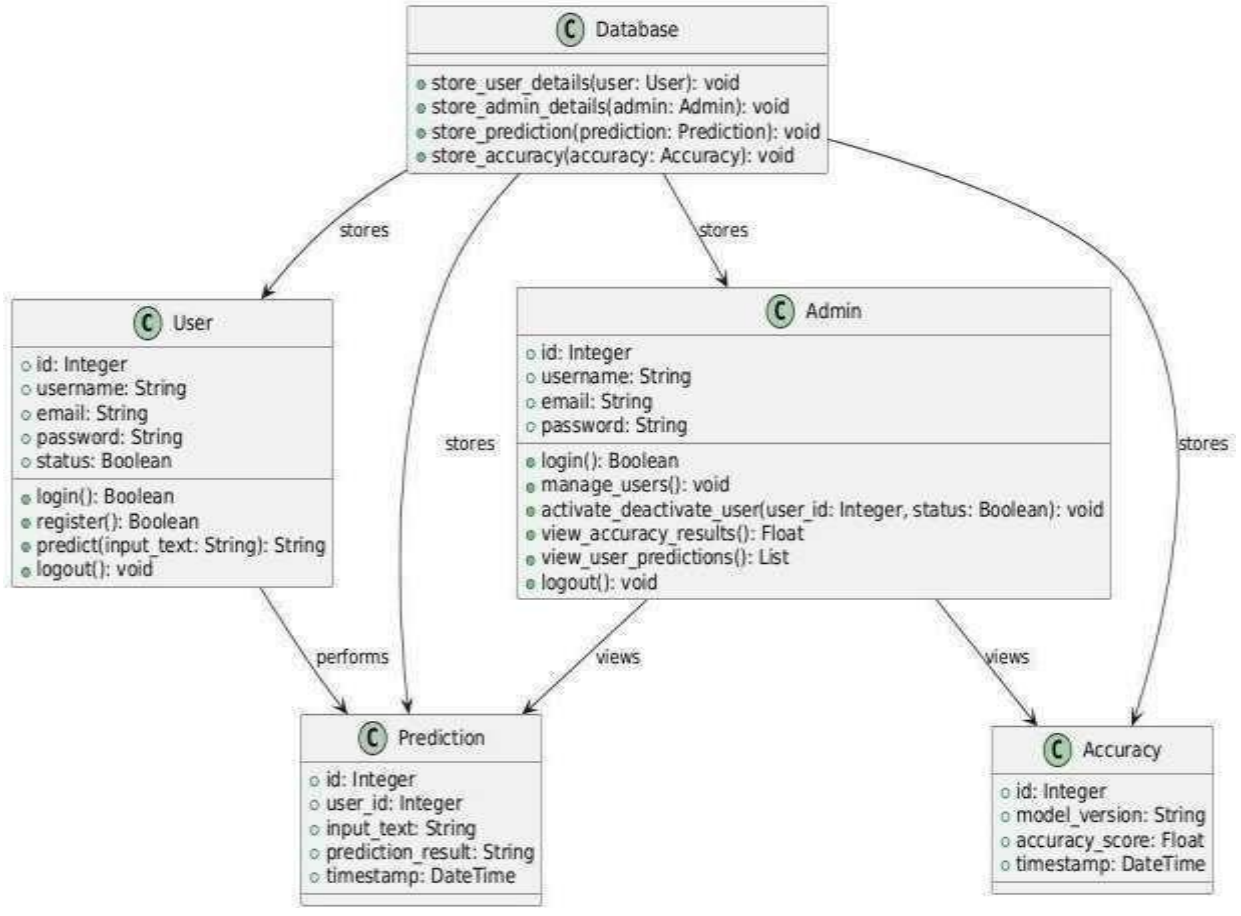


Fig 5.3.4 Class Diagram

CHAPTER-6

CODING AND IMPLEMENTATION

6. CODING AND IMPLEMENTATION

6.1 Source Code

Views.py:

```
from django.contrib import admin

# Register your models here.
from django.apps import AppConfig


class AdminsConfig(AppConfig):
default_auto_field = 'django.db.models.BigAutoField' name = 'Admins'
from django.db import models

# Create your models here.
from django.test import TestCase

# Create your tests here. from django.urls import path from Admins.views import * urlpatterns =

[
path('adminhome/', adminhome, name='adminhome'), path('admin_update_userstatus/<int:user_id>/',
admin_update_userstatus,
name='admin_update_userstatus'),
path('adminuserpredictions/', adminuserpredictions, name='adminuserpredictions'),
path('adminresultratio/', adminresultratio, name='adminresultratio'), path('adminaccuracy/',
adminaccuracy, name='adminaccuracy'),
]
from django.shortcuts import render, redirect from django.contrib.auth.models import User from
django.contrib import messages
from User.models import PredictionResult from django.core.paginator import Paginator

from django.http import HttpResponse
from django.core.files.storage import FileSystemStorage import os from
xgboost import XGBRegressor
from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.model_selection import
train_test_split
from sklearn.metrics import mean_squared_error, r2_score import pandas as pd
```

```
    def adminhome(request):
    users = User.objects.filter(is_staff=False, is_superuser=False) return render(request,
    "Admin/adminhome.html", {"users": users})

def admin_update_userstatus(request, user_id): try: user
    = User.objects.get(id=user_id)

    # Toggle the is_active status user.is_active = not user.is_active user.save() #

    Display message based on the action if user.is_active:
        messages.success(request, f"User {user.username} has been activated.") else:
    messages.success(request, f"User {user.username} has been deactivated.")

        return redirect('adminhome') # Redirect back to the admin home page except User.DoesNotExist:
    messages.error(request, "User not found.") return redirect('adminhome')

    def adminuserpredictions(request):
    predictions = PredictionResult.objects.all().order_by('-created_at')

    paginator = Paginator(predictions, 7) page_number = request.GET.get('page') page_obj =
    paginator.get_page(page_number)

        return render(request, 'Admin/adminuserpredictions.html', {'page_obj': page_obj}) from collections import

    Counter
    import json

    def adminresultratio(request):
    # Fetch all PredictionResult objects predictions = PredictionResult.objects.all()

    # Calculate ratios for sentiment, emotion, and suicidal results sentiment_counts =
    Counter(pred.sentiment_result for pred in predictions) emotion_counts = Counter(pred.emotion_result for
    pred in predictions) suicidal_counts = Counter(pred.suicidal_result for pred in predictions)

    # Convert counts to JSON format for use in templates sentiment_data = {
```

```
    "labels": list(sentiment_counts.keys()), "data": list(sentiment_counts.values())
    }
    emotion_data = {
    "labels": list(emotion_counts.keys()), "data": list(emotion_counts.values())
    }
    suicidal_data = {
    "labels": list(suicidal_counts.keys()), "data": list(suicidal_counts.values())
    }

    context = {
    "sentiment_data": json.dumps(sentiment_data), "emotion_data": json.dumps(emotion_data), "suicidal_data":
    json.dumps(suicidal_data),
    }

        return render(request, 'Admin/adminresultratio.html', context) def adminaccuracy(request):
    metrics = {}
if request.method == 'POST' and request.FILES['uploaded_file']: # Save the uploaded file
    uploaded_file = request.FILES['uploaded_file'] fs = FileSystemStorage()
    file_path = fs.save(uploaded_file.name, uploaded_file) file_full_path = fs.path(file_path)

    try:
    # Load uploaded CSV file
    combined_data = pd.read_csv(file_full_path)

    # TF-IDF Vectorization
    tfidf_vectorizer = TfidfVectorizer(max_features=5000)
    X = tfidf_vectorizer.fit_transform(combined_data['Text'].fillna(""))

    y_sentiment = combined_data['sentiment'] y_emotion = combined_data['emotion'] y_suicidal =
    combined_data['suicidal_label']

    # Split data
X_train_sentiment, X_test_sentiment, y_train_sentiment, y_test_sentiment = train_test_split( X, y_sentiment,
    test_size=0.3, random_state=42
    )
    X_train_emotion, X_test_emotion, y_train_emotion, y_test_emotion = train_test_split(
```

```
    X, y_emotion, test_size=0.3, random_state=42
    )
X_train_suicidal, X_test_suicidal, y_train_suicidal, y_test_suicidal = train_test_split( X, y_suicidal,
    test_size=0.3, random_state=42
    )

    # Train XGBoost for Sentiment
    xgb_model_sentiment = XGBRegressor(random_state=42, verbosity=1)
    xgb_model_sentiment.fit(X_train_sentiment, y_train_sentiment) sentiment_predictions =
    xgb_model_sentiment.predict(X_test_sentiment)     sentiment_mse     =
    mean_squared_error(y_test_sentiment, sentiment_predictions) sentiment_r2 = r2_score(y_test_sentiment,
    sentiment_predictions)

    # Train XGBoost for Emotion
    xgb_model_emotion = XGBRegressor(random_state=42, verbosity=1)
    xgb_model_emotion.fit(X_train_emotion, y_train_emotion) emotion_predictions =
    xgb_model_emotion.predict(X_test_emotion) emotion_mse = mean_squared_error(y_test_emotion,
    emotion_predictions) emotion_r2 = r2_score(y_test_emotion, emotion_predictions)

    # Train XGBoost for Suicidal
    xgb_model_suicidal = XGBRegressor(random_state=42, verbosity=1)
    xgb_model_suicidal.fit(X_train_suicidal, y_train_suicidal) suicidal_predictions =
    xgb_model_suicidal.predict(X_test_suicidal) suicidal_mse = mean_squared_error(y_test_suicidal,
    suicidal_predictions) suicidal_r2 = r2_score(y_test_suicidal, suicidal_predictions)

    # Store metrics metrics = {
    "Sentiment MSE": sentiment_mse, "Sentiment R2": sentiment_r2, "Emotion MSE": emotion_mse,
    "Emotion R2": emotion_r2, "Suicidal MSE": suicidal_mse, "Suicidal R2": suicidal_r2,
    }

except Exception as e: metrics['error'] = str(e) finally:
    # Clean up uploaded file os.remove(file_full_path)

    return render(request, "Admin/adminaccuracy.html", {"metrics": metrics})
```

Backend.py:

```
from django.shortcuts import render, redirect from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout from django.contrib import messages """
ASGI config for Backend project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/ """

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Backend.settings') application =

get_asgi_application()

"""
Django settings for Backend project.

Generated by 'django-admin startproject' using Django 4.2.17.

For more information on this file, see https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see https://docs.djangoproject.com/en/4.2/ref/settings/ """

import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(_file_).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-+$qij7fh0ho@j_#%53bjmm03xx6m_&n(!5c@k(eesc^&^l19=x'
```

```
# SECURITY WARNING: don't run with debug turned on in production! DEBUG = True

ALLOWED_HOSTS = []


# Application definition INSTALLED_APPS = [
'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions',
'django.contrib.messages', 'django.contrib.staticfiles', 'User',
'Admins',
]

MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'Backend.urls' TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [BASE_DIR /
'templates'],
'APP_DIRS': True, 'OPTIONS': {
'context_processors': [ 'django.template.context_processors.debug',
'django.template.context_processors.request', 'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
```

```
]

WSGI_APPLICATION = 'Backend.wsgi.application'


# Database
#  https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
'default': {
'ENGINE': 'django.db.backends.sqlite3', 'NAME': BASE_DIR / 'db.sqlite3',
}
}


# Password validation
#    https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
{
'NAME':  'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{
'NAME':  'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
'NAME':   'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
'NAME':  'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]


# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/ LANGUAGE_CODE = 'en-us' TIME_ZONE

= 'UTC'

USE_I18N = True USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
#    https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'),]

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field DEFAULT_AUTO_FIELD =

'django.db.models.BigAutoField'

"""
WSGI config for Backend project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/ """

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Backend.settings') application =

get_wsgi_application()
"""
URL configuration for Backend project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
    Examples:
    Function views
1. Add an import:  from my_app import views
        2. Add a URL to urlpatterns:  path('', views.home, name='home') Class-based views
1. Add an import:  from other_app.views import Home
        2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home') Including another URLconf
1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
```

```
"""
from django.contrib import admin from django.urls import path, include from django.conf import settings
from django.conf.urls.static import static from Backend.views import *

urlpatterns = [
path('admin/', admin.site.urls), path('User/', include('User.urls')), path('Admins/',
include('Admins.urls')),

path('', index, name='index'), path('home_page/', index, name='home_page'), path('login_page/',
login_page, name='login_page'), path('register_page/', register_page,
name='register_page'), path('user_logout/', user_logout, name='user_logout'), path('user_login/',
user_login, name='user_login'), path('user_registration/', user_registration, name='user_registration')

]
if settings.DEBUG:
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

def index(request):
return render(request, "index.html")

def login_page(request):
return render(request, "login.html")

def register_page(request):
return render(request, "register.html")

# Define the login function def user_login(request):
if request.method == "POST":
username = request.POST.get('username') password = request.POST.get('password')

# Authenticate user
user = authenticate(request, username=username, password=password)

if user is not None:
if not user.is_active: # User is inactive
```

```
    messages.error(request, "Your account is inactive. Please contact the admin.") return redirect('login_page')

    # Login the user login(request, user) if

    user.is_staff or user.is_superuser:
    # Redirect to admin home if user is staff return redirect('adminhome')
    else:
    # Redirect to user home if user is not staff return redirect('userhome') else:
    # Invalid username or password
    messages.error(request, "Invalid username or password.") return redirect('login_page') return

    render(request, 'login.html')

    # Define the user registration function def user_registration(request):
    if request.method == "POST":
    username = request.POST.get('username') email = request.POST.get('email') password =
    request.POST.get('password')
    confirm_password = request.POST.get('confirm_password') first_name =
    request.POST.get('first_name')
    last_name = request.POST.get('last_name')

    # Check if passwords match
if password != confirm_password: messages.error(request, "Passwords do not match.") return
    redirect('register_page')

    # Check if username already exists
if User.objects.filter(username=username).exists(): messages.error(request, "Username already exists.") return
    redirect('register_page')

    # Check if email already exists
if User.objects.filter(email=email).exists(): messages.error(request, "Email already exists.") return
    redirect('register_page')

    # Create the user with is_active set to False user = User.objects.create_user(
```

```
username=username, email=email, password=password, first_name=first_name, last_name=last_name
)
user.is_active = False  # Set is_active to False by default user.save()

messages.success(request, "Registration successful! Please wait for admin approval.") return
redirect('login_page')

   return render(request, 'register.html') # Define the logout function def
user_logout(request):
logout(request)
messages.success(request, "You have been logged out successfully.") return redirect('login_page')
```

Frontend:

Base.html:

```
<!DOCTYPE html>
{% load static%}
<html lang="en">

<head>
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<title>Forecasting National-Level Self-Harm Trends with Social Networks</title>
<meta name="description" content="">
<meta name="keywords" content="">

<!-- Fonts -->
<link href="https://fonts.googleapis.com" rel="preconnect">
<link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
          <link
href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,90
0;1,100;
1,300;1,400;1,500;1,700;1,900&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700
;0,800; 0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Raleway:ital,wght@0,100;0,20
0;0,300;
0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display
=swap" rel="stylesheet">

<!-- Vendor CSS Files -->
<link href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
<link href="{% static 'vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">
```

```
<link href="{% static 'vendor/aos/aos.css' %}" rel="stylesheet">
<link href="{% static 'vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">
<link href="{% static 'vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">

<!-- Main CSS File -->
<link href="{% static 'css/main.css' %}" rel="stylesheet">


</head>

<body class="index-page">

<header id="header" class="header d-flex align-items-center light-background sticky-top">
<div class="container-fluid position-relative d-flex align-items-center justify-content-between">

<a class="logo d-flex align-items-center me-auto me-xl-0">
<!-- Uncomment the line below if you also wish to use an image logo -->
<!-- <img src="assets/img/logo.png" alt=""> -->
<h1>Self-Harm Trends with Social Networks</h1>
</a>

<nav id="navmenu" class="navmenu">
<ul>
<li><a href="{% url 'home_page' %}">Home</a></li>
<li><a href="{% url 'login_page' %}">Login</a></li>
</ul>
<i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
</nav>

<div class="header-social-links">
</div>

</div>
</header>

<main class="main">

{%block contents%}

{%endblock%}

</main>
```

```
<!-- Scroll Top -->
<a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-center"><i
class="bi bi-arrow-up-short"></i></a>

<!-- Preloader -->
<div id="preloader"></div>

<!-- Vendor JS Files -->
<script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
<script src="{% static 'vendor/php-email-form/validate.js' %}"></script>
<script src="{% static 'vendor/aos/aos.js' %}"></script>
<script src="{% static 'vendor/waypoints/noframework.waypoints.js' %}"></script>
<script src="{% static 'vendor/purecounter/purecounter_vanilla.js' %}"></script>
<script src="{% static 'vendor/swiper/swiper-bundle.min.js' %}"></script>
<script src="{% static 'vendor/glightbox/js/glightbox.min.js' %}"></script>
<script src="{% static 'vendor/imagesloaded/imagesloaded.pkgd.min.js' %}"></script>
<script src="{% static 'vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>

<!-- Main JS File -->
<script src="{% static 'js/main.js' %}"></script>

</body>

</html>
```

Index.html:

```
{% extends 'base.html' %}
{% load static %}

{% block contents %}


<!-- Hero Section -->
<section id="hero" class="hero section">

<img src="{% static 'img/Sentiment-Analysis-Instagram-scaled.jpg' %}" alt="" data-aos="fade-in">

<div class="container text-center" data-aos="zoom-out" data-aos-delay="100">
<div class="row justify-content-center">
<div class="col-lg-8">
<h2>Forecasting National-Level Self-Harm Trends with Social Networks</h2>
<p></p>

</div>
```

```
</div>
</div>
</section><!-- /Hero Section -->

{% endblock %}
```

Login.html:

```
{% extends "base.html" %}

{% block contents %}
  <!-- Login Section -->
  <section id="login" class="login section">

    <!-- Section Title -->
    <div class="container section-title" data-aos="fade-up">
      <h2>Login</h2>
      <p>Please enter your credentials to log in.</p>
    </div><!-- End Section Title -->

    <div class="container" data-aos="fade-up" data-aos-delay="100">

      <div class="row gy-4 justify-content-center">

        <div class="col-lg-5">
          {% if messages %}
          <div class="row">
            <div class="col-md-12">
              {% for message in messages %}
                <div class="alert alert-success" role="alert">
                  {{ message }}
                </div>
              {% endfor %}
            </div>
          </div>
        {% endif %}

          <form method="post" action="{% url 'user_login' %}"  data-aos="fade-up" data-aos-

            {% csrf_token %}
            <div class="row gy-4">

              <div class="col-md-12">
                <label for="username-field" class="pb-2">Username</label>
                <input type="text" name="username" id="username-field" class="form-control"
```

```
            </div>
            <div class="col-md-12">
              <label for="password-field" class="pb-2">Password</label>
              <input type="password" name="password" id="password-field" class="form-control"

            </div>

            <div class="col-md-12 text-center">
              <button type="submit" class="btn btn-primary">Login</button>
            </div>

            <div class="col-md-12 text-center">
              <p>Don't have an account? <a href="{% url 'register_page' %}">Register here</a>.</p>
            </div>

          </div>
        </form>
      </div><!-- End Login Form -->

    </div>

  </div>

</section><!-- /Login Section -->
{% endblock %}
```

register.html:

```
{% extends "base.html" %}

{% block contents %}
  <!-- Register Section -->
  <section id="register" class="register section">

    <!-- Section Title -->
    <div class="container section-title" data-aos="fade-up">
      <h2>Register</h2>
      <p>Create your account by filling out the details below.</p>
    </div><!-- End Section Title -->

    <div class="container" data-aos="fade-up" data-aos-delay="100">

      <div class="row gy-4 justify-content-center">
```

```
<div class="col-lg-6">
   {% if messages %}
  <div class="row">
     <div class="col-md-12">
        {% for message in messages %}
           <div class="alert alert-success" role="alert">
              {{ message }}
           </div>
        {% endfor %}
     </div>
  </div>
{% endif %}
 <form method="post" action="{% url 'user_registration' %}" data-aos="fade-up" data-aos-

   {% csrf_token %}
   <div class="row gy-4">

    <div class="col-md-6">
     <label for="first-name-field" class="pb-2">First Name</label>
     <input type="text" name="first_name" id="first-name-field" class="form-control"

    </div>

    <div class="col-md-6">
     <label for="last-name-field" class="pb-2">Last Name</label>
     <input type="text" name="last_name" id="last-name-field" class="form-control"

    </div>

    <div class="col-md-12">
     <label for="username-field" class="pb-2">Username</label>
     <input type="text" name="username" id="username-field" class="form-control"

    </div>

    <div class="col-md-12">
     <label for="email-field" class="pb-2">Email</label>
     <input type="email" name="email" id="email-field" class="form-control" required="">
    </div>

    <div class="col-md-12">
     <label for="password-field" class="pb-2">Password</label>
     <input type="password" name="password" id="password-field" class="form-control"

    </div>
```

```
 class="form-control"
   required="">
            </div>

            <div class="col-md-12 text-center">
              <button type="submit" class="btn btn-primary">Register</button>
            </div>

            <div class="col-md-12 text-center">
              <p>Already have an account? <a href="{% url 'login_page' %}">Login here</a>.</p>
            </div>

          </div>
        </form>
      </div>

    </div>

  </div>

   </section>
{% endblock %}
```

admin.html:

```
{% extends "Admin/adminbase.html" %}

{% block contents %}
<div class="container mt-5">
  <h1 class="text-center mb-4">Upload a File to Train the Model</h1>
  <div class="card p-4 shadow">
            <form id="upload-form" method="POST" enctype="multipart/form-data"
onsubmit="showLoadingScreen()">
      {% csrf_token %}
      <div class="mb-3">
        <label for="file" class="form-label">Choose a CSV file:</label>
        <input type="file" id="file" name="uploaded_file" class="form-control" required>
      </div>
      <div class="text-center">
        <button type="submit" class="btn btn-primary">Upload and Train</button>
        </div>
      </form>
  </div>

  {% if metrics %}
```

```
        <h2 class="text-center">Training Metrics</h2>
        <div class="table-responsive">
            <table class="table table-bordered table-striped table-hover mt-3">
                <thead class="table-primary">
                    <tr>
                        <th>Metric</th>
                        <th>Value</th>
                    </tr>
                </thead>
                <tbody>
                    {% for key, value in metrics.items %}
                        <tr>
                            <td>{{ key }}</td>
                            <td>{{ value }}</td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
  {% endif %}
</div>

<!-- Loading Screen -->
<div id="loading-screen" class="loading-screen d-none">
  <div class="spinner-border text-primary" role="status">
    <span class="visually-hidden">Loading...</span>
  </div>
  <p class="mt-3">Processing, please wait...</p>
</div>

<script>
  function showLoadingScreen() {
    // Show the loading screen
    document.getElementById('loading-screen').classList.remove('d-none');
  }
</script>
<style>
  .loading-screen
    { position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
```

```
        display: flex;
        justify-content: center;
        align-items: center;
        flex-direction: column;
        background-color: rgba(255, 255, 255, 0.8);
        z-index: 9999;
    }
</style>

<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Forecasting National-Level Self-Harm Trends with Social Networks</title>
  <meta name="description" content="">
  <meta name="keywords" content="">

  <!-- Fonts -->
  <link href="https://fonts.googleapis.com" rel="preconnect">
  <link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
          <link
href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,90
0;1,100;
1,300;1,400;1,500;1,700;1,900&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700
;0,800;
0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Raleway:ital,wght@0,100;0,20
0;0,300;
0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display
=swap" rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/aos/aos.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">

  <!-- Main CSS File -->
  <link href="{% static 'css/main.css' %}" rel="stylesheet">


</head>

<body class="index-page">
```

```
<header id="header" class="header d-flex align-items-center light-background sticky-top">
<div class="container-fluid position-relative d-flex align-items-center justify-content-between">
<a  class="logo d-flex align-items-center me-auto me-xl-0">
<!-- Uncomment the line below if you also wish to use an image logo -->
<!-- <img src="assets/img/logo.png" alt=""> -->
<h1>Self-Harm Trends with Social Networks</h1>
</a>
<nav id="navmenu" class="navmenu">
<ul>
<li><a href="{% url 'adminhome' %}">Users</a></li>
<li><a href="{% url 'adminuserpredictions' %}">User Predictions</a></li>
<li><a href="{% url 'adminresultratio' %}">Graph</a></li>
<li><a href="{% url 'adminaccuracy' %}">Accuracy</a></li>
<li><a href="{% url 'user_logout' %}">Logout</a></li>
</ul>
<i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
</nav>

<div  class="header-social-links">
</div>

</div>
</header>

<main class="main">

{%block contents%}

{%endblock%}

</main>


<!-- Scroll Top -->
        <a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-
center"><i class="bi bi-arrow-up-short"></i></a>

<!-- Preloader -->
<div id="preloader"></div>

<!-- Vendor JS Files -->
<script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
<script src="{% static 'vendor/php-email-form/validate.js' %}"></script>
<script src="{% static 'vendor/aos/aos.js' %}"></script>
```

```
<script src="{% static 'vendor/waypoints/noframework.waypoints.js' %}"></script>
<script src="{% static 'vendor/purecounter/purecounter_vanilla.js' %}"></script>
<script src="{% static 'vendor/swiper/swiper-bundle.min.js' %}"></script>
<script src="{% static 'vendor/glightbox/js/glightbox.min.js' %}"></script>
<script src="{% static 'vendor/imagesloaded/imagesloaded.pkgd.min.js' %}"></script>
<script src="{% static 'vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>
<!-- Main JS File -->
<script src="{% static 'js/main.js' %}"></script>

</body>

</html>
{% extends "Admin/adminbase.html" %}

{% block contents %}
<div class="container mt-5">
<h2 class="text-center mb-4">Registered Users</h2>

<div class="table-responsive">
<table class="table table-striped table-hover">
<thead class="thead-dark">
<tr>
<th>ID</th>
<th>Username</th>
<th>Email</th>
<th>First Name</th>
<th>Last Name</th>
<th>Date Joined</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
{% for user in users %}
<tr>
<td>{{ user.id }}</td>
<td>{{ user.username }}</td>
<td>{{ user.email }}</td>
<td>{{ user.first_name }}</td>
<td>{{ user.last_name }}</td>
<td>{{ user.date_joined }}</td>
<td>{{ user.is_active|yesno:"Active,Inactive"  }}</td>
<td>
{% if user.is_active %}
```

```
sm">De activate<
/a>
sm">Act
{% else
%}
<a href="{
% url
'admin_
update_
userstatu s'
user.id
%}"
class="b tn
btn-success
btn-

{% endif
%}
</td>
</tr>
{% empty %}
<tr>
<td colspan="8" class="text-center">No users found.</td>
</tr>
{% endfor %}
</tbody>
</table>
</div>
</div>
{% endblock %}
{% extends 'Admin/adminbase.html' %}

{% block contents %}
<div class="container mt-4">
<h2 class="text-center">Prediction Results Analysis</h2>

<!-- Row for the first set of charts -->
<div class="row">
<div class="col-lg-6 col-md-12 mb-4">
<div class="card">
<div class="card-body">
<h4 class="card-title text-center">Sentiment Analysis (Pie Chart)</h4>
<canvas id="sentimentPieChart"></canvas>
</div>
</div>
</div>
</div>
```

```
<div class="col-lg-6 col-md-12 mb-4">
<div class="card">
<div class="card-body">
<h4 class="card-title text-center">Emotion Analysis (Pie Chart)</h4>
<canvas id="emotionPieChart"></canvas>
</div>
</div>
</div>
</div>

<!-- Row for the second set of charts -->
<div class="row">
<div class="col-lg-6 col-md-12 mb-4">
<div class="card">
<div class="card-body">
<h4 class="card-title text-center">Suicidal Analysis (Pie Chart)</h4>
<canvas id="suicidalPieChart"></canvas>
</div>
</div>
</div>
<div class="col-lg-6 col-md-12 mb-4">
<div class="card">
<div class="card-body">
<h4 class="card-title text-center">Sentiment Trends (Line Chart)</h4>
<canvas id="sentimentLineChart"></canvas>
</div>
</div>
</div>
</div>
</div>

<!-- Include Chart.js -->
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
// Load data from context
const sentimentData = JSON.parse('{{ sentiment_data|escapejs }}'); const emotionData =
JSON.parse('{{ emotion_data|escapejs }}'); const suicidalData =
JSON.parse('{{ suicidal_data|escapejs }}');

// Pie Chart Configurations
const createPieChart = (ctx, data) => { return new Chart(ctx, { type:
'pie', data: {
labels: data.labels, datasets: [{ data:
data.data, backgroundColor: [
```

```
'rgba(255, 99, 132, 0.7)',
'rgba(54, 162, 235, 0.7)',
'rgba(255, 206, 86, 0.7)',
'rgba(75, 192, 192, 0.7)',
'rgba(153, 102, 255, 0.7)'
],
}],
},
});
};

// Initialize Pie Charts createPieChart(document.getElementById('sentimentPieChart'),
sentimentData); createPieChart(document.getElementById('emotionPieChart'), emotionData);
createPieChart(document.getElementById('suicidalPieChart'),  suicidalData);

// Line Chart Configuration for Sentiment Trends const createLineChart = (ctx, data) =>
{ return new Chart(ctx, { type: 'line', data: {
labels: data.labels, datasets: [{
label: 'Sentiment Trends Over Time', data: data.data,
backgroundColor: 'rgba(75, 192, 192, 0.2)',
borderColor: 'rgba(75, 192, 192, 1)',
borderWidth: 1, fill: true,
}],
},
});
};

// Initialize Line Chart createLineChart(document.getElementById('sentimentLineChart'),
sentimentData);
</script>
{% endblock %}
{% extends "Admin/adminbase.html" %}

{% block title %} Admin User Predictions
{% endblock %}

{% block contents %}
<div class="container my-5">
<h2 class="mb-4">User Predictions</h2>

<table class="table table-striped table-hover">
<thead class="table-dark">
<tr>
<th>#</th>
<th>Input Text</th>
```

```
<th>Sentiment</th>
<th>Emotion</th>
<th>Suicidal Risk</th>
<th>Created At</th>
</tr>
</thead>
<tbody>
{% for prediction in page_obj %}
<tr>
<td>{{ forloop.counter }}</td>
<td>{{ prediction.input_text|truncatechars:50 }}</td>
<td>{{ prediction.sentiment_result }}</td>
<td>{{ prediction.emotion_result }}</td>
<td>{{ prediction.suicidal_result }}</td>
<td>{{ prediction.created_at|date:"Y-m-d H:i:s" }}</td>
</tr>
{% empty %}
<tr>
<td colspan="6" class="text-center">No predictions found.</td>
</tr>
{% endfor %}
</tbody>
</table>

<!-- Pagination Controls -->
<nav aria-label="Page navigation example">
<ul class="pagination justify-content-center">
{% if page_obj.has_previous %}
<li class="page-item">
<a class="page-link" href="?page={{ page_obj.previous_page_number }}">Previous</a>
</li>
{% endif %}
{% for num in page_obj.paginator.page_range %}
<li class="page-item {% if page_obj.number == num %}active{% endif %}">
<a class="page-link" href="?page={{ num }}">{{ num }}</a>
</li>
{% endfor %}
{% if page_obj.has_next %}
<li class="page-item">
<a class="page-link" href="?page={{ page_obj.next_page_number }}">Next</a>
</li>
{% endif %}
</ul>
</nav>
</div>
```

```
{% endblock %}
```

user.html:

```
<!DOCTYPE html>
{% load static%}
<html lang="en">

<head>
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<title>Forecasting National-Level Self-Harm Trends with Social Networks</title>
<meta name="description" content="">
<meta name="keywords" content="">

<!-- Fonts -->
<link href="https://fonts.googleapis.com" rel="preconnect">
<link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
        <link
href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,90
0;1,100;
1,300;1,400;1,500;1,700;1,900&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700
;0,800; 0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Raleway:ital,wght@0,100;0,20
0;0,300;
0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display
=swap" rel="stylesheet">

<!-- Vendor CSS Files -->
<link href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
<link href="{% static 'vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">
<link href="{% static 'vendor/aos/aos.css' %}" rel="stylesheet">
<link href="{% static 'vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">
<link href="{% static 'vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">

<!-- Main CSS File -->
<link href="{% static 'css/main.css' %}" rel="stylesheet">


</head>

<body class="index-page">

<header id="header" class="header d-flex align-items-center light-background sticky-top">
<div class="container-fluid position-relative d-flex align-items-center justify-content-between">
```

```
<a  class="logo d-flex align-items-center me-auto me-xl-0">
<!-- Uncomment the line below if you also wish to use an image logo -->
<!-- <img src="assets/img/logo.png" alt=""> -->
<h1>Self-Harm Trends with Social Networks</h1>
</a>

<nav id="navmenu" class="navmenu">
<ul>
<li><a href="{% url 'userhome' %}">Profile</a></li>
<li><a href="{% url 'userpredict' %}">Predict</a></li>
<li><a href="{% url 'user_logout' %}">Logout</a></li>
</ul>
<i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
</nav>

<div class="header-social-links">
</div>

</div>
</header>

<main class="main">

{%block contents%}

{%endblock%}

</main>


<!-- Scroll Top -->
        <a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

<!-- Preloader -->
<div id="preloader"></div>

<!-- Vendor JS Files -->
<script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
<script src="{% static 'vendor/php-email-form/validate.js' %}"></script>
<script src="{% static 'vendor/aos/aos.js' %}"></script>
<script src="{% static 'vendor/waypoints/noframework.waypoints.js' %}"></script>
<script src="{% static 'vendor/purecounter/purecounter_vanilla.js' %}"></script>
<script src="{% static 'vendor/swiper/swiper-bundle.min.js' %}"></script>
<script src="{% static 'vendor/glightbox/js/glightbox.min.js' %}"></script>
```

```
<script src="{% static 'vendor/imagesloaded/imagesloaded.pkgd.min.js' %}"></script>
<script src="{% static 'vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>

<!-- Main JS File -->
<script src="{% static 'js/main.js' %}"></script>

</body>

</html>
{% extends "User/userbase.html" %}

{% block contents %}
<div class="container mt-5">
<h1 class="text-center">Welcome, {{ user.username }}!</h1>

<div class="user-details mt-4 p-4 border rounded shadow-sm bg-light">
<h2 class="text-center mb-4">Your Account Details</h2>
<table class="table table-striped">
<thead class="thead-light">
<tr>
<th>Field</th>
<th>Details</th>
</tr>
<th>Username</th>
<td>{{ user.username }}</td>
</tr>
<tr>
<th>First Name</th>
<td>{{ user.first_name }}</td>
</tr>
<tr>
<th>Last Name</th>
<td>{{ user.last_name }}</td>
</tr>
<tr>
<th>Email</th>
<td>{{ user.email }}</td>
</tr>
<tr>
<th>Date Joined</th>
<td>{{ user.date_joined }}</td>
</tr>
<tr>
<th>Last Login</th>
<td>{{ user.last_login }}</td>
```

```
</tr>
<tr>
<th>Status</th>
{% block contents %}
<div class="container mt-5">
<!-- Page Title -->
        <h2 class="text-center mb-4" data-aos="fade-up" style="font-weight: 600; color: #343a40;">Enter
Text for Prediction</h2>

<!-- Form Section -->
<form method="POST" action="{% url 'userpredict' %}" data-aos="fade-up" data-aos-delay="100">
{% csrf_token %}
<div class="form-group">
<label for="inputText" style="font-weight: 500; color: #495057;">Your Message:</label>
          <textarea id="inputText" name="input_text" class="form-control" rows="5"
placeholder="Enter your text here..." required></textarea>
</div>
<button type="submit" class="btn btn-primary mt-3">Submit</button>
</form>

<!-- Prediction Results Section -->
{% if result %}
<div class="mt-5 p-4 rounded shadow-sm bg-light" data-aos="fade-up" data-aos-delay="200">
<h3 class="mb-3" style="font-weight: 600; color: #343a40;">Prediction Results</h3>
<p><strong>Sentiment:</strong> {{ result.sentiment }}</p>
<p><strong>Emotion:</strong> {{ result.emotion }}</p>
<p><strong>Suicidal:</strong> {{ result.suicidal }}</p>
</div>
{% endif %}
</div>
{% endblock %}
```

Main.js:

```
(function() { "use strict";

/**
* Apply .scrolled class to the body as the page is scrolled down
*/
function toggleScrolled() {
const selectBody = document.querySelector('body'); const selectHeader =
document.querySelector('#header');
        if (!selectHeader.classList.contains('scroll-up-sticky') &&
!selectHeader.classList.contains('sticky-top') && !selectHeader.classList.contains('fixed-top')) return;
```

```
window.scrollY > 100 ? selectBody.classList.add('scrolled') : selectBody.classList.remove('scrolled');
}

document.addEventListener('scroll', toggleScrolled); window.addEventListener('load',




toggleScrolled);
/**
* Mobile nav toggle
*/
const mobileNavToggleBtn = document.querySelector('.mobile-nav-toggle');

function mobileNavToogle() { document.querySelector('body').classList.toggle('mobile-nav-active');
mobileNavToggleBtn.classList.toggle('bi-list'); mobileNavToggleBtn.classList.toggle('bi-x');
}
mobileNavToggleBtn.addEventListener('click', mobileNavToogle);

/**
* Hide mobile nav on same-page/hash links
*/
document.querySelectorAll('#navmenu a').forEach(navmenu => { navmenu.addEventListener('click', ()
=> {
if (document.querySelector('.mobile-nav-active')) { mobileNavToogle();
}
});

});

/**
* Toggle mobile nav dropdowns
*/
document.querySelectorAll('.navmenu .toggle-dropdown').forEach(navmenu =>
{ navmenu.addEventListener('click', function(e) {
e.preventDefault(); this.parentNode.classList.toggle('active');
this.parentNode.nextElementSibling.classList.toggle('dropdown-active');
e.stopImmediatePropagation();
});
});

/**
* Preloader
*/
```

```
  /**
  * Scroll top button
  */
  let scrollTop = document.querySelector('.scroll-top');

function toggleScrollTop() { if (scrollTop) {
  window.scrollY > 100 ? scrollTop.classList.add('active') : scrollTop.classList.remove('active');
  }
  }
scrollTop.addEventListener('click', (e) => { e.preventDefault();
window.scrollTo({ top: 0,
  behavior: 'smooth'
  });
  });

  window.addEventListener('load', toggleScrollTop); document.addEventListener('scroll',
  toggleScrollTop);

  /**
  * Animation on scroll function and init
  */
function aosInit() { AOS.init({ duration: 600, easing:
  'ease-in-out', once: true,
  mirror: false
  });
  }
  window.addEventListener('load',  aosInit);

  /**
  * Animate the skills items on reveal
  */
  let skillsAnimation = document.querySelectorAll('.skills-animation');  skillsAnimation.forEach((item)
  => {
new Waypoint({ element: item, offset: '80%', handler:
  function(direction) {
  let progress = item.querySelectorAll('.progress .progress-bar'); progress.forEach(el =>
  { el.style.width = el.getAttribute('aria-valuenow') + '%';
```

```
  });
  }
  });
  });

  /**
  * Initiate Pure Counter
  */
  new PureCounter();

  /**
  * Init swiper sliders
  */
  function initSwiper() {
document.querySelectorAll(".init-swiper").forEach(function(swiperElement) { let config =
  JSON.parse(
  swiperElement.querySelector(".swiper-config").innerHTML.trim()
  );

if (swiperElement.classList.contains("swiper-tab"))
  { initSwiperWithCustomPagination(swiperElement, config);
  } else {
  new Swiper(swiperElement, config);
  }
  });
  }

  window.addEventListener("load", initSwiper);

  /**
  * Initiate glightbox
  */
const glightbox = GLightbox({ selector: '.glightbox'
  });

  /**
  * Init isotope layout and filters
  */
document.querySelectorAll('.isotope-layout').forEach(function(isotopeItem) { let layout =
  isotopeItem.getAttribute('data-layout') ?? 'masonry';
  let filter = isotopeItem.getAttribute('data-default-filter') ?? '*'; let sort =
  isotopeItem.getAttribute('data-sort') ?? 'original-order';

  let initIsotope;
```

```
  imagesLoaded(isotopeItem.querySelector('.isotope-container'),  function()  {
initIsotope =  new Isotope(isotopeItem.querySelector('.isotope-container'), { itemSelector: '.isotope-
  item',
  layoutMode: layout, filter: filter,
  sortBy: sort
  });
  });

isotopeItem.querySelectorAll('.isotope-filters li').forEach(function(filters)
  { filters.addEventListener('click', function() {
  isotopeItem.querySelector('.isotope-filters .filter-active').classList.remove('filter-active');
  this.classList.add('filter-active');
  initIsotope.arrange({
  filter:  this.getAttribute('data-filter')
  });
if (typeof aosInit === 'function') { aosInit();
  }
  }, false);
  });

  });

  })();
```

6.2 Implementation:

6.2.1 Front-End Implementation:

The front-end of the self-harm forecasting system provides a simple, user-friendly, and interactive interface for users and analysts. The main modules include User Login, Data Input/Upload, Trend Visualization, and Admin Monitoring Panel.

The Login module enables secure access to the system, ensuring that only authorized users can perform analysis. Input validation is applied to ensure correctness and prevent invalid data submission. The Admin Panel allows administrators to monitor system activity, view prediction history, and manage datasets and users.

The **Data Input module** allows users to provide social media text data or upload datasets for analysis. Once submitted, the input is sent to the backend through APIs for processing. The system then returns forecasting results, which are displayed in the form of graphs, charts, or statistical summaries representing predicted self-harm trends.

The interface is designed with consistent layouts, clear navigation, and structured output presentation to ensure usability and easy interpretation of forecasting results.

6.2.2 Backend Implementation (Django):

The backend is implemented using Python-based frameworks (such as Django/Flask), providing a secure and scalable environment for processing and prediction. The architecture follows a modular design:

- **Models:** Store datasets, extracted mental signals, prediction results, and user information.
- **Views/Controllers:** Handle user requests, perform validation, and trigger preprocessing and forecasting operations.
- **APIs:** Define endpoints for data submission, preprocessing, prediction, and result retrieval.

Security mechanisms such as authentication, request validation, and controlled access ensure safe handling of user data. Prediction results and processed data are stored for further analysis and monitoring.

6.2.3 Model Integration and Processing Workflow

The machine learning module is integrated as a core backend component. When input data is received, it passes through a preprocessing pipeline that includes normalization, tokenization, stop - word removal, and lemmatization.

After preprocessing, feature extraction techniques such as TF-IDF and sentiment analysis are applied to extract **mental signals** like stress, sadness, fear, and anxiety. These signals are aggregated and converted into **multivariate time-series data**.

The forecasting process is carried out using machine learning regression models, which analyze temporal patterns in the data. Time-delay embedding techniques may also be applied to capture dependencies across time. The trained model generates predictions of self-harm trends, including expected injuries and deaths.

The results are returned to the front-end in a structured format and visualized for easy understanding.

6.2.4 Deployment and Reliability

The system is deployed on a standard server environment with proper configuration for security and performance. The application supports handling large-scale social media datasets efficiently. APIs are tested to ensure consistent behavior under different usage conditions.

Unit testing and integration testing are performed to validate preprocessing steps, model predictions, and system responses. The system is designed to be scalable, allowing integration with real-time data sources and cloud-based services in the future.

6.2.5 Conclusion

The implementation successfully integrates a user-friendly interface, a secure backend system, and a machine learning-based forecasting framework into a unified self-harm prediction platform. The system processes social media data, extracts mental signals, and generates accurate forecasting results in real time.

The modular architecture allows easy future enhancements such as real-time data integration, advanced deep learning models, and improved visualization techniques. Overall, the system provides a practical and scalable solution for forecasting self-harm trends and supporting data-driven decision-making in public health.

CHAPTER-7

SYSTEM TESTING

7. SYSTEM TESTING

System testing is a critical activity that ensures the developed self-harm forecasting system performs accurately, consistently, and reliably under real operating conditions. The primary purpose of testing is to identify defects, validate system behavior, and confirm that all functional and non- functional requirements are satisfied.

In this project, system testing focuses on validating all major modules, including front-end interfaces, backend services, preprocessing components, mental signal extraction, time-series construction, and the machine learning-based forecasting engine. Testing ensures that data input, processing workflows, model predictions, and result visualization operate correctly without errors or inconsistencies.

System testing was conducted across multiple datasets and scenarios to ensure correctness, robustness, and usability. Special emphasis was placed on forecasting accuracy, handling noisy social media data, and maintaining system stability during repeated prediction operations.

7.1 Types of System Testing

7.1.1 Unit Testing

Unit testing was performed to validate individual components of the system independently. Each module such as preprocessing, feature extraction, time-series generation, and prediction functions was tested using controlled inputs.

Key focus areas included:

- text preprocessing (tokenization, normalization, lemmatization)
- feature extraction and mental signal identification
- time-series data generation
- internal logic of forecasting models
- database operations for storing and retrieving data

Unit testing ensured that each module functioned correctly in isolation and produced expected outputs.

7.1.2 Integration Testing

Integration testing verified that different modules functioned correctly when combined together.

Modules tested in integration included:

• front-end input submission with backend processing

• preprocessing and feature extraction linked with forecasting models

• mental signal extraction combined with time-series generation

• prediction storage and retrieval from the database

This testing ensured smooth data flow and correct interaction between system components.

7.1.3 Functional Testing

Functional testing ensured that all system features worked according to specifications and user expectations.

Validation included:

• correct processing of valid input data

• handling of invalid or incomplete inputs

• accurate forecasting of self-harm trends

• proper display of results in graphs or statistical form

• correct navigation and workflow execution

Functional testing confirmed that the system is user-friendly and meets all functional requirements.

7.1.4 System Testing

System testing evaluated the entire application as a complete system. The focus was on overall performance, reliability, and accuracy under real-world conditions.

Testing verified:

• coordination between all modules

• correct handling of large-scale social media datasets

• forecasting accuracy under different scenarios

• stability during repeated prediction requests

The system demonstrated consistent and reliable performance.

7.1.5 White-Box Testing

White-box testing was applied to internal processing components such as preprocessing pipelines, feature extraction logic, and forecasting algorithms. Internal data flow, logic execution, and model computations were analyzed to ensure correctness..

7.1.6 Black-Box Testing

Black-box Black-box testing evaluated the system from the user's perspective without considering internal code. Inputs were provided through the interface, and outputs were verified to ensure correctness and usability.

This testing ensured that the system behaves correctly under various input conditions.

7.1.7 Acceptance Testing

Acceptance testing ensured that the system meets all project requirements and user expectations. The system was evaluated based on usability, prediction accuracy, and output clarity.

Test Result Summary:

All test cases were executed successfully, and the system met all expected requirements without major defects.

7.2 Testing Strategies

A structured testing strategy was followed throughout the project to ensure systematic validation from individual components to the complete system.

7.2.1 Test Strategy and Approach

Testing Testing was performed using both manual and automated methods. Dataset-based test cases were used to validate forecasting accuracy and system behavior.

Key objectives included:

- verifying correctness of preprocessing and feature extraction
- ensuring accurate mental signal extraction
- validating time-series modeling and forecasting
- ensuring smooth interaction between front-end and backend
- testing performance under noisy and real-world data

7.2.2 Test Objectives

The main objectives of testing were:

- all input fields and system components must function correctly
- system responses should be fast and reliable
- invalid inputs should be handled safely
- forecasting results should follow expected patterns
- system navigation should be smooth and intuitive

7.2.3 Features Tested

The major features tested include:

- data input validation and dataset handling
- preprocessing and feature extraction pipeline
- mental signal extraction accuracy
- time-series data generation
- forecasting model performance

7.2.4 Integration Testing Strategy

Integration Integration testing focused on ensuring correct interaction between modules:

• proper alignment between preprocessing and feature extraction

• correct transformation of data into time-series format

• smooth communication between backend and prediction models

• accurate flow of data between system components

7.2.5 Acceptance Criteria

The system was accepted only when the following conditions were satisfied:

- accurate forecasting of self-harm trends
- stable and efficient system performance
- error-free input handling and navigation
- meaningful and interpretable output presentation

- compliance with all functional and non-functional requirements

7.2.6 Overall Test Results

All planned test cases were successfully executed. The system demonstrated stable performance, logical correctness, and consistent forecasting accuracy. The machine learning models provided reliable predictions across different datasets and scenarios.

7.2.7 Conclusion

System testing confirmed that the developed self-harm forecasting system satisfies all functional and performance requirements. The system operates reliably under various conditions, processes large - scale social media data effectively, and produces accurate forecasting results. The modular design and rigorous testing ensure that the system is robust, scalable, and ready for real- world deployment.

7.3 Sample Test Cases

S No.	Test Case	Expected Result	Result	Remarks (if any)
01.	User Login	User is authenticated and granted access to the system dashboard	Pass	Verify incorrect credentials show appropriate error messages
02.	User Registration	New user account is created and stored in the database	Pass	Validate email format and duplicate-account handling
03.	User Account Activation	Registered user becomes active and can log in successfully	Pass	Ensure inactive users cannot access the system
04.	Text-Based Prediction	System accepts social media text or dataset input successfully	Pass	Test with valid, invalid, and empty inputs
05.	Forecasting Prediction	System predicts self-harm trends (injuries/deaths) accurately	Pass	Test with different datasets and time ranges
06.	Admin Panel	Admin can view users, predictions	Pass	Ensure only authorized admins can access this module

Table no 7.3 Test Cases

Test Case 1:

Fig 7.3.1 User Login

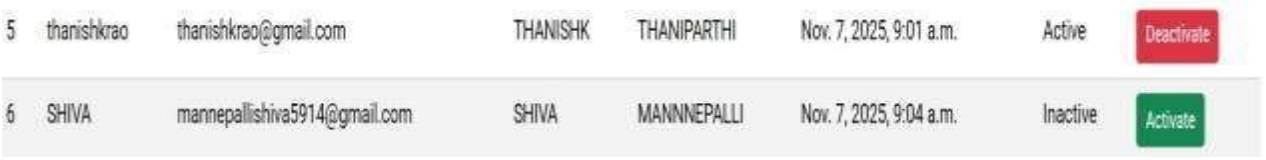
Description: The user enters valid login credentials and submits the form. The system authenticates the credentials and redirects the user to the dashboard, displaying a personalized welcome message and complete account details such as username, email, status, and last login time. Successful login confirms that authentication and user-profile retrieval are working correctly.

Test Case 2:

Fig 7.3.2 User Registration

Description: After submitting valid registration details, the system successfully creates the user account and displays a confirmation message indicating that registration is complete. The message also informs the user that their account will remain inactive until approved by the administrator. This confirms that the registration workflow and approval control mechanism are functioning correctly.

Test Case 3:



5	thanishkrao	thanishkrao@gmail.com	THANISHK	THANIPARTHI	Nov. 7, 2025, 9:01 a.m.	Active	Deactivate
6	SHIVA	mannepallishiva5914@gmail.com	SHIVA	MANNNEPALLI	Nov. 7, 2025, 9:04 a.m.	Inactive	Activate

Fig.7.3.3 User Account Activation

Description: When the administrator approves the newly registered user, the system changes the account status from Inactive to Active and displays a confirmation message indicating successful activation. The user is now allowed to log in and access the system. This verifies that the admin-controlled activation process works correctly and prevents unauthorized access before approval.

Test Case 4:

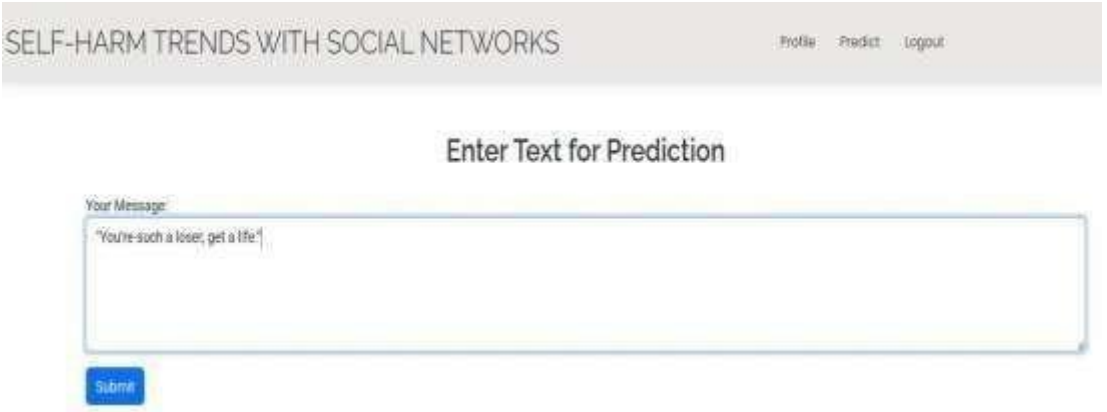


Fig. 7.3.4 Text-Based Prediction

Description: The user inputs social media text data or selects a dataset for analysis through the system interface. The system processes the data using a preprocessing pipeline that includes text cleaning, tokenization, and feature extraction. The processed data is then fed into the trained machine learning model (e.g., XGBoost-based forecasting model), which analyzes patterns related to mental health indicators and self-harm signal.

Test Case 5:

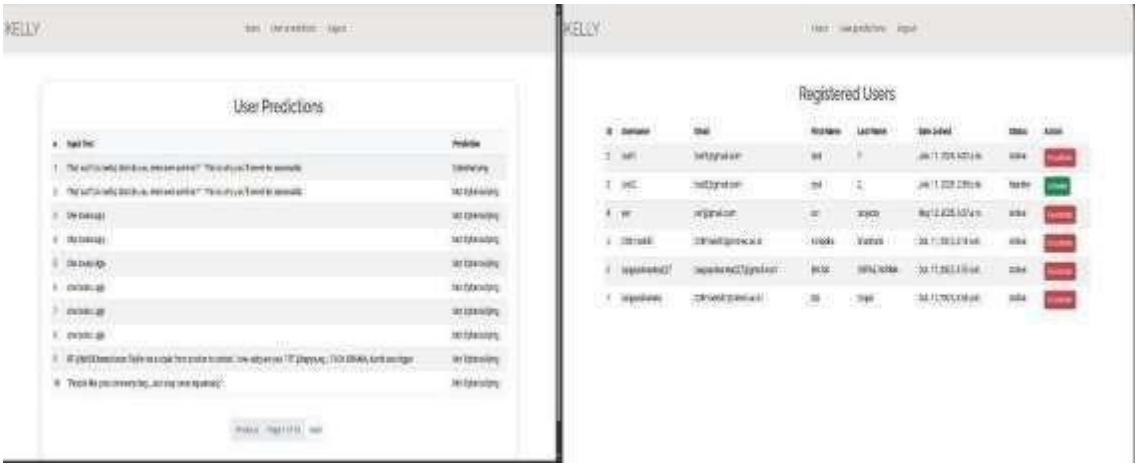


Fig. 7.3.6 Admin Panel

Description: The administrator logs into the system and navigates to the Admin Panel, where they can view both registered users and user prediction history. The system correctly displays user details, account status, and prediction logs, and allows the admin to activate or deactivate user accounts when required. This confirms that administrative monitoring and access-control functions are working correctly.

CHAPTER-8

RESULTS

8. RESULTS

Fig 8.1 Forecasting National-Level Self - Harm trends landing page

Description: The primary interface of the application, titled "Forecasting National-Level Self-Harm Trends with Social Networks," presents the system as a machine learning–based platform that analyzes social media data to detect mental health patterns and predict national-level self-harm trends. .

Fig 8.2 User Authentication & System Access

Description: The entry point for the forensics platform, featuring Registration and Login modules. This interface ensures secure access to the ensemble-based detection tools for social media monitoring.

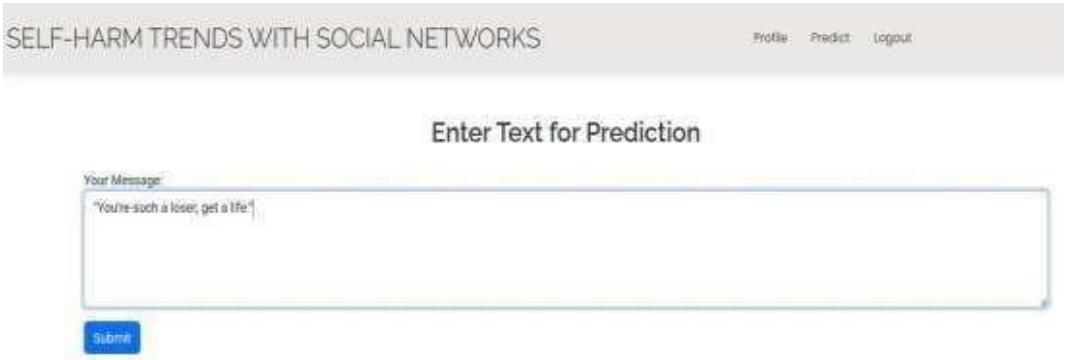


Fig 8.3 Ensemble-Based Text Classification

Description: This screen shows the trend forecasting module in action. Social media data is processed through a machine learning model (such as XGBoost), which analyzes patterns related to mental health indicators. The system then predicts national-level self-harm trends and displays the results as risk levels or trend insights.

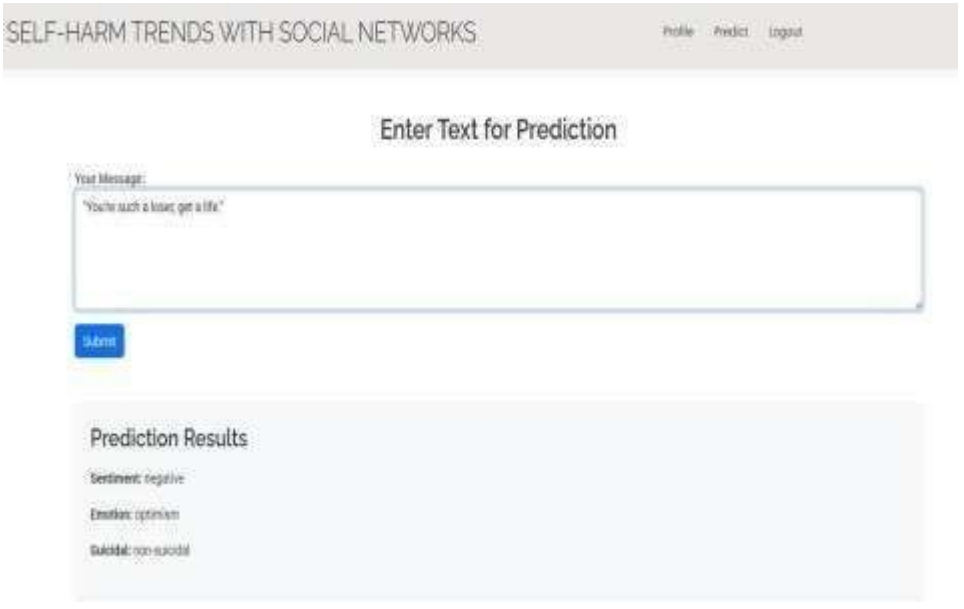


Fig 8.4 Prediction Results

Description: This This interface shows the system generating prediction results for self-harm trends. By leveraging advanced machine learning techniques (such as XGBoost with optimized feature analysis), the model effectively captures subtle patterns in social media data that indicate changes in mental health signals

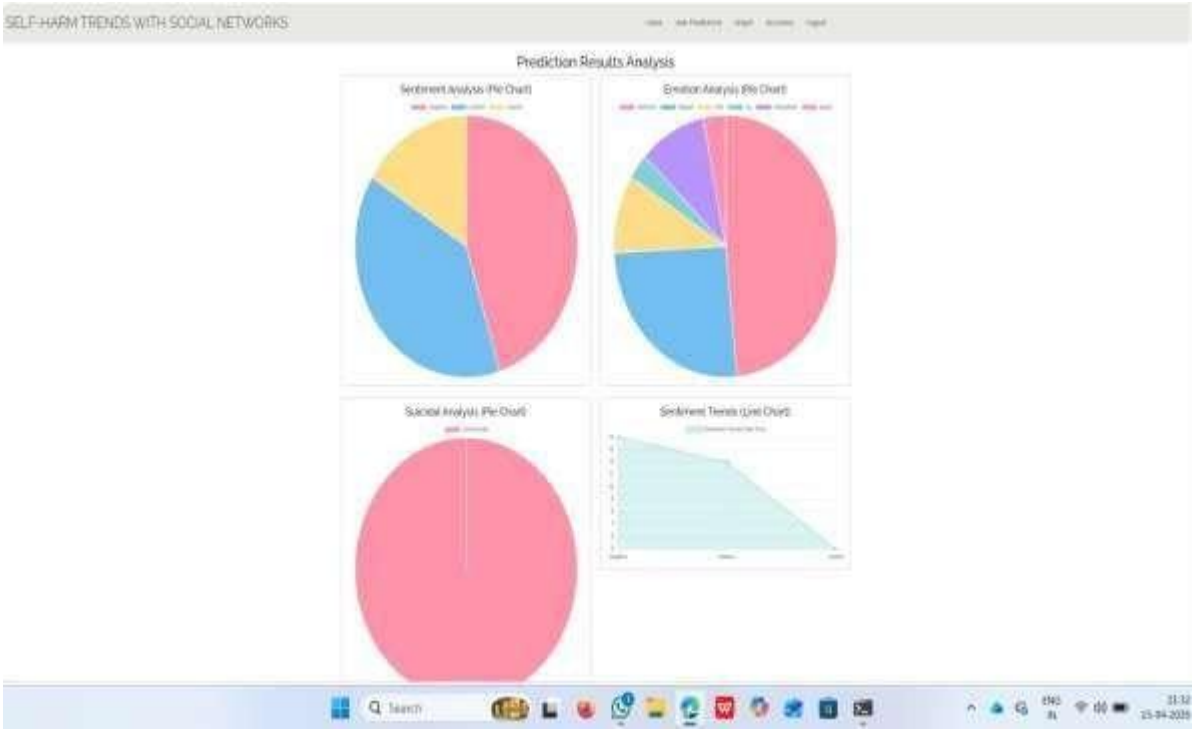


Fig 8.5 Sentiment, Suicidal & Emotion Analysis

Description: A dashboard displaying Graph Comparisons across two different datasets. It represents the count of each class across two different datasets.

CHAPTER-9

CONCLUSION

9. CONCLUSION

The present project introduces a comprehensive and intelligent framework for forecasting national-level self-harm trends using social network data. The system effectively leverages machine learning, natural language processing (NLP), and time-series analysis to provide accurate, reliable, and scalable predictions of self-harm patterns. The overall framework follows a structured pipeline that includes data collection, preprocessing, feature extraction, mental signal identification, time-series modeling, and forecasting, ensuring a systematic and end-to-end solution.

During the implementation phase, large-scale social media textual data was collected and processed to extract meaningful insights about user behavior and emotional states. The raw data underwent multiple preprocessing steps, including text cleaning, normalization, tokenization, and stop-word removal, to eliminate noise and improve data quality. These preprocessing techniques ensured that unstructured and informal social media text could be effectively analyzed.

The processed text was then transformed into numerical features using techniques such as TF- IDF and sentiment analysis. This step enabled the extraction of important **mental signals**, including stress, anxiety, sadness, and suicidal tendencies, which serve as key indicators for forecasting self-harm trends. These signals were further aggregated over time to form **multivariate time-series data**, allowing the system to capture temporal patterns and behavioral trends.

The core strength of the system lies in its machine learning-based forecasting approach. By applying regression models and time-series techniques, the system is capable of predicting future self-harm trends, including potential injuries and deaths. The model supports both nowcasting and future forecasting, enabling timely insights and proactive decision-making. The results demonstrate that integrating social media-based mental signals with time-series modeling significantly improves prediction accuracy compared to traditional approaches.

From a system design perspective, the project follows a well-defined software engineering methodology, including requirement analysis, system architecture design, UML modeling, and testing. The modular and scalable architecture ensures flexibility, allowing future enhancements such as integration of advanced deep learning models or real-time data streams. The system also ensures efficient data handling and reliable performance across different scenarios.

Beyond technical contributions, this project addresses a critical public health issue by providing a data-

driven approach to understanding and predicting self-harm trends. The ability to analyze real-time social media data and generate early warnings can assist policymakers, healthcare professionals, and researchers in implementing preventive measures and improving mental health support systems. The use of open-source tools and standard computational resources further enhances the practicality and accessibility of the solution.

Furthermore, the system demonstrates the importance of integrating interdisciplinary approaches by combining concepts from data science, machine learning, psychology, and public health. By analyzing behavioral patterns reflected in social media, the project highlights how digital data can be utilized to gain meaningful insights into societal mental health conditions. This interdisciplinary approach not only enhances prediction capability but also opens new avenues for research in mental health analytics and early risk detection.

Another significant contribution of this project is its ability to handle large-scale and unstructured data efficiently. Social media data is inherently noisy, dynamic, and diverse, yet the proposed system successfully processes and transforms it into structured and meaningful information. The implementation proves that with proper preprocessing, feature extraction, and modeling techniques, even complex and unstructured data sources can be effectively utilized for accurate forecasting and analysis.

Finally, the system establishes a strong foundation for future advancements in intelligent healthcare monitoring systems. With further improvements such as real-time data integration, advanced deep learning models, and enhanced visualization tools, the framework can evolve into a comprehensive decision-support system. Such systems can play a crucial role in reducing self-harm risks, improving mental health awareness, and supporting timely interventions, thereby contributing to overall societal well-being.

In conclusion, the project successfully demonstrates that combining NLP, machine learning, and time-series analysis provides an effective approach for forecasting self-harm trends at a national level. The developed system achieves its intended objectives by delivering accurate predictions, handling large-scale data efficiently, and offering a scalable framework for future research and real-world deployment.

CHAPTER-10

FUTURE ENHANCEMENTS

10. FUTURE ENHANCEMENTS

Although the developed system provides a strong and effective foundation for forecasting national-level self-harm trends using social networks, several enhancements can be implemented to further improve its accuracy, scalability, adaptability, and real-world applicability. As mental health patterns evolve over time and across different populations, continuous improvements in both modeling techniques and system architecture are essential to ensure reliable and meaningful predictions.

One of the primary directions for future enhancement is the integration of advanced deep learning models such as BERT (Bidirectional Encoder Representations from Transformers), RoBERTa, and LSTM-based architectures. These models are capable of capturing deeper contextual and semantic relationships in textual data, enabling more accurate extraction of mental signals such as stress, depression, and suicidal tendencies. Transformer-based models, in particular, can better understand subtle expressions, sarcasm, and context-dependent language in social media posts. A hybrid approach that combines deep learning models with existing machine learning and time-series forecasting techniques can significantly improve prediction performance.

Another important enhancement involves expanding the dataset to include multilingual and cross-domain data. Current systems are often limited to English-language datasets, which restricts their applicability in diverse regions. Extending the system to support multiple languages using multilingual embeddings or transformer models would make it more inclusive and globally applicable. Additionally, incorporating multimodal data such as images, videos, and audio along with textual data can enhance the system's ability to detect mental health signals in complex real-world scenarios.

The system can also be extended to support real-time monitoring and streaming analysis. By integrating with live social media APIs or data streams, the framework can continuously analyze incoming data and provide instant forecasting of self-harm trends. This would enable early warning systems and proactive intervention by healthcare organizations and policymakers, making the system highly valuable for real -time public health monitoring.

Furthermore, the backend system can be enhanced to improve scalability and performance for large -scale deployment. This includes optimizing APIs, implementing caching mechanisms, enabling efficient handling of large datasets, and using cloud-based infrastructure. Incorporating

distributed computing and parallel processing techniques can further improve system performance and reduce processing time for large volumes of social media data.

In addition, adaptive learning mechanisms can be introduced to improve the system over time. By incorporating feedback loops where experts validate predictions, the system can continuously retrain and refine its models. This human-in-the-loop approach helps improve accuracy, reduce errors, and adapt to changing patterns in mental health trends. Periodic model updates using new data can ensure long-term effectiveness.

From a usability and analytical perspective, integrating interactive dashboards and visualization tools can enhance the system's practical utility. These dashboards can display insights such as temporal trends, regional variations, and severity levels of self-harm risks. Such visualizations can assist researchers, healthcare professionals, and policymakers in making informed decisions and implementing preventive measures.

Finally, strengthening privacy, security, and ethical considerations is essential for real-world deployment. Future enhancements may include data anonymization, secure data storage, and compliance with data protection standards. Incorporating explainable AI (XAI) techniques can provide transparency in predictions, while fairness-aware models can reduce bias and ensure ethical use of the system.

Collectively, these enhancements will transform the current system into a more intelligent, scalable, and reliable platform for forecasting self-harm trends. By integrating advanced technologies, expanding data diversity, enabling real-time analysis, and ensuring ethical deployment, the system can significantly contribute to improving mental health awareness and preventive strategies at a national level.

REFERENCES

REFERENCES

1. A. Sharma, R. Verma, and S. Gupta, “Recent Advances in Social Media-Based Mental Health Analysis,” 2026.
2. P. Mehta and K. Reddy, “Ensemble Learning Techniques for Mental Health Prediction,” 2025.
3. S. Kumar, N. Patel, and V. Singh, “Time-Series Forecasting of Health Data Using Machine Learning,” 2025.
4. R. Iyer and M. Das, “Role of Emotional Signals in Predicting Human Behavior,” 2025.
5. L. Chen and Y. Wang, “Social Media as a Proxy for Public Health Monitoring,” 2024.
6. M. Khan and A. Ali, “Machine Learning Models for Suicide and Self-Harm Prediction,” 2024.
7. T. Nguyen and H. Tran, “Multivariate Time-Series Analysis in Healthcare Forecasting,” 2024.
8. J. Brown and E. Wilson, “Natural Language Processing for Mental Signal Extraction,” 2024.
9. X. Zhang, Y. Liu, and H. Chen, “Deep Learning Approaches for Mental Health Prediction Using Social Media Data,” IEEE Transactions on Affective Computing, 2024.
10. M. Patel and R. Singh, “Early Detection of Suicidal Ideation Using Social Network Analysis,” International Journal of Data Science and Analytics, 2024.
11. S. Tuarob, K. Chatrinan, T. Noraset, T. Tawichsri, and T. Thaipisutikul, “Forecasting National-Level Self-Harm Trends With Social Networks,” IEEE Access, 2023.
12. P. Shah and A. Shah, “Comparative Evaluation of Voting, Bagging, and Boosting Techniques in Prediction Systems,” Procedia Computer Science, 2023.
13. M. Jain and S. Gupta, “An Ensemble-Driven Analytical Framework for Social Data Processing,” IEEE International Conference on Data Science and Advanced Analytics, 2023.
14. R. Biswas, M. Paul, and A. Das, “Deep-Hybrid Ensemble Models for Pattern Detection in Social Data,” arXiv Preprint, 2023.
15. S. W. Azumah, P. Oppong, and A. Mensah, “Adversarial-Aware Ensemble Learning Approaches for Social Data Analysis,” arXiv Preprint, 2023.
16. A. Toktarova, S. Khusainova, and M. Fayzullina, “Comparative Study of Ensemble and Single Models for Social Network Hate Speech,” International Journal of Advanced Computer

Science and Applications, 2023.

17. M. S. Jahan and M. Oussalah, “A Systematic Review of Hate Speech Detection with Ensemble NLP Models,” Neurocomputing, 2023.
18. T. Mahmud, R. Hasan, and S. Rahman, “Ensemble-Based Prediction for Low-Resource Social Data,” Information Processing & Management, 2023..
19. A. Ahmed and F. Khan, “Machine Learning and Ensemble Techniques for Social Data Prediction,” CEUR Workshop Proceedings, 2023.
20. J . Sathya and F. M. H. Fernandez, “Ontology-Assisted Models for Social Media Text Analysis,” International Conference on Communication and Electronics Systems (ICCES), 2023.