DYNAMIC PREDICTION OF DRUG CONSUMPTION PATTERNS USING MACHINE LEARNING TECHNIQUES

A project report submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

by

KONDA LAKSHMI PRASANNA (21BF1A0593)
LALALU SHAIK USMAN (21BF1A05A0)
PARASU SAMBA SIVARAM (21BF1A05D2)
NAGELI BHARATH KUMAR (21BF1A05B9)

Under the guidance of

Mr. R. Venkataramana, M. Tech

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu
Accredited by NBA, New Delhi & NAAC with 'A' grade)
Karakambadi Road, TIRUPATI – 517507

2021 - 2025

SRI VENKATESWARA COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Approved by AICTE, New Delhi & PermanentlyAffiliated to JNTUA, Ananthapuramu
Accredited by NBA, New Delhi & NAACwith 'A' Grade)
Karakambadi Road, TIRUPATI – 517507.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled, **"DYNAMIC PREDICTION OF DRUG CONSUMPTION PATTERNS USING MACHINE LEARNING TECHNIQUES"** is a bonafide record of the project work done and submitted by

KONDA LAKSHMI PRASANNA	***21BF1A0593***
LALALU SHAIK USMAN	***21BF1A05A0***
PARASU SAMBA SIVARAM	***21BF1A05D2***
NAGELI BHARATH KUMAR	***21BF1A05B9***

under my guidance and supervision for the partial fulfillment of the requirements for the award of B.Tech degree in **COMPUTER SCIENCE AND ENGINEERING.** This project is the result of their own effort and that it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above

GUIDE

Mr. R. Venkataramana., M. Tech
Associate Professor,
Department of CSE,
SV College of Engineering,
Tirupati.

HEAD OF THE DEPARTMENT

Dr. A. Ganesh., B.Tech., M.Tech., Ph.D (PDF)
Professor and HOD,
Department of CSE,
SV College of Engineering,
Tirupati.

External Viva-Voce Exam held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the project entitled **"DYNAMIC PREDICTION OF DRUG CONSUMPTION PATTERNS USING MACHINE LEARNING TECHINIQUES",** done by us under the esteemed guidance of **Mr. R. Venkataramana, M. Tech.,** Assistant Professor, and is submitted in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Technology** in **Computer Science and Engineering**.

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

Date:	**KONDA LAKSHMI PRASANNA**	**21BF1A0593**
Place:	**LALALU SHAIK USMAN**	**21BF1A05A0**
	PARASU SAMBA SIVARAM	**21BF1A05D2**
	NAGELI BHARATH KUMAR	**21BF1A05B9**

ACKNOWLEDGEMENT

We are thankful to our guide **Mr. R. Venkataramana, M. Tech., Assistant Professor, Department of CSE** for his valuable guidance and encouragement. His helping attitude and suggestions have helped in the successful completion of the project report.

We would like to express our gratefulness and sincere thanks to **Dr. A. Ganesh., Ph.D (PDF)., Professor & HOD, Department of CSE,** for his kind help and encouragement during the course of study and in the successful completion of the project work.

We have great pleasure in expressing out hearty thanks to our beloved Principal **Dr. N. Sudhakar Reddy** for spending his valuable time with us to complete this project.

Successful completion of any project cannot be done without proper support and encouragement. We sincerely thank the Management for providing all the necessary facilities during course of study.

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

KONDA LAKSHMI PRASANNA	**21BF1A0593**
LALALU SHAIK USMAN	**21BF1A05A0**
PARASU SAMBA SIVARAM	**21BF1A05D2**
NAGELI BHARATH KUMAR	**21BF1A05B9**

CONTENTS

	Abstract	**i**
	List of figures	**ii**
	List of tables	**iii**
	Abbreviations	**iv**
Chapter No.	**Description**	**Page No.**
1	**INTRODUCTION**	**01**
2	**PROJECT DESCRIPTION**	**02**
	2.1 PROBLEM DEFINITION	02
	2.2 PROJECT DETAILS	03
	2.3 DATA	04
3	**COMPUTATIONAL ENVIROMENT**	**07**
	3.1 SOFTWARE REQUIREMENTS	07
	3.2 HARDWARE REQUIREMENTS	07
	3.3 FUNCTIONAL REQUIREMENTS	08
	3.4 SOFTWARE FEATURES	08
4	**FEASIBILITY STUDY**	**12**
	4.1 TECHNICAL FEASIBILITY	12
	4.2 OPERATIONAL FEASIBILITY	13
	4.3 ECONOMIC FEASIBILITY	13
5	**SYSTEM ANALYSIS**	**14**
	5.1 EXISTING SYSTEM	14
	5.1.1 Disadvantages of Existing System	14
	5.2 PROPOSED SYSTEM	15
	5.2.1 Advantages of Proposed System	16
6	**SYSTEM DESIGN**	**17**
	6.1 SYSTEM ARCHITECTURE	18
	6.2 UML DIAGRAMS	20

Chapter No.	Description	Page No.
7	**SYSTEM IMPLEMENTATION**	27
	7.1 ALGORITHM	27
	7.2 WORKING AND MODULE EXPLANATION	29
8	**SOURCE CODE**	**32**
9	**TESTING**	**51**
	9.1 UNIT TESTING	51
	9.2 INTEGRATION TESTING	52
	9.3 SYSTEM TESTING	52
	9.4 ACCEPTANCE TESTING	53
10	**SCREEN LAYOUTS**	**54**
11	**CONCLUSIONS**	**59**
12	**FUTURE ENHANCEMENTS**	**60**
13	**BIBLIOGRAPHY**	**61**

ABSTRACT

Drug consumption, whether occasional or habitual, has far-reaching implications on individuals and society, emphasizing the need for systems capable of analyzing and predicting patterns of use. This project focuses on developing an advanced system to predict whether an individual has consumed drugs and, if so, the recency of consumption, such as within an hour, a day, or a year. Leveraging demographic, socio-economic, and psychological data—including factors such as age, gender, education, and behavioral traits—the system employs machine learning algorithms to identify patterns of drug usage. Random Forest and XGBoost are utilized independently to capture complex relationships within the data and optimize prediction accuracy. By providing insights into the timing and likelihood of drug consumption, this research aims to assist healthcare professionals, policymakers, and rehabilitation efforts in crafting timely and effective responses to drug-related challenges.

Keywords: drug consumption prediction, machine learning, Random Forest, XGBoost, timing of drug use, demographic analysis, socio-economic factors, behavioral traits, prediction accuracy, early detection, public health strategies.

LIST OF FIGURES

Fig. No.	Figure Name	Page No.
6.1	System Architecure	19
6.2.1	Use Case Diagram	21
6.2.2	Class Diagram	22
6.2.3	Sequence Diagram	23
6.2.4	Activity Diagram	24
6.2.5	Component Diagram	25
6.2.6	Deployment Diagram	26
7.1.1	XGBoost Classifier Diagram	28
7.1.2	Random Forest Classifier Diagram	29
8.3	System Testing	53
10.1	Index Page	54
10.2	Input Pages	57
10.3	Output Pages	58

LIST OF TABLES

Table No.	Table Name	Page No.
8.1	Unit Testing	51
8.2	Integration Testing	52

ABBREVIATIONS

WHO	-	World Health Organization
CSV	-	Comma-Separated values
ML	-	Machine Learning
UML	-	Unified Modeling Language
RF	-	Random Forest
XGBoost	-	Extreme Gradient Boosting

Chapter 1

INTRODUCTION

Substance abuse and drug consumption are major public health concerns affecting millions of individuals globally. These behaviors not only have detrimental effects on the physical and mental health of individuals but also contribute to a wide range of societal issues including crime, unemployment, family disruption, and increased healthcare costs. Identifying patterns in drug usage and predicting potential instances of consumption can be crucial in formulating effective intervention strategies and rehabilitation plans. With the emergence of advanced data-driven technologies, there is an opportunity to leverage machine learning (ML) techniques to enhance our understanding and prediction of drug consumption behaviors.

Traditional methods of drug consumption analysis rely on clinical interviews, surveys, and psychological assessments, which are often time-consuming, subjective, and limited in scale. Machine learning, on the other hand, provides an automated and scalable solution capable of handling vast and complex datasets. By analyzing factors such as demographic details (age, gender, education), socio-economic status, psychological traits, and behavioral indicators, ML models can uncover hidden patterns and correlations that may not be evident through conventional statistical analysis.

This project proposes a machine learning-based approach to dynamically predict whether an individual has consumed drugs and determine the recency of consumption—ranging from within an hour to over a year. The system utilizes powerful algorithms such as **Random Forest** and **XGBoost** to analyze and model non-linear relationships within the data. These algorithms are chosen for their ability to manage high-dimensional data, handle class imbalances, and provide high prediction accuracy.

Unlike existing systems that are often constrained by limited prediction capabilities and are less effective in diverse real-world settings, this proposed model integrates advanced ensemble learning methods to improve reliability and adaptability.

Chapter 2

PROJECT DESCRIPTION

2.1 PROBLEM DEFINITION

Drug consumption is a pervasive global issue that has significant consequences for individuals, families, communities, and public health systems. Whether the usage is experimental, occasional, or habitual, substance abuse poses risks such as physical and mental health deterioration, criminal behavior, loss of productivity, and increased healthcare costs. Timely detection and accurate prediction of drug use patterns are crucial for deploying effective interventions, managing public health responses, and designing targeted rehabilitation programs. However, predicting drug consumption—especially its frequency, recency, and risk factors—remains a complex challenge due to the multifactorial nature of substance abuse.

Traditional approaches to drug use analysis rely on interviews, surveys, and statistical methods, which are often limited in scope, subjective in interpretation, and slow to produce actionable results. While some machine learning methods have been used to predict drug consumption, they primarily focus on binary classification (user vs. non-user) and lack the capacity to accurately determine when drug use occurred (e.g., within an hour, a day, or a year). Moreover, these models often struggle to deal with imbalanced datasets, non-linear data relationships, and noisy or incomplete inputs—issues that are common in behavioral and health-related data.

Factors influencing drug consumption are diverse and include demographic characteristics, psychological traits and socio-economic status. These variables interact in complex, non-linear ways that are not easily captured by traditional statistical models. Without an intelligent system capable of recognizing these hidden patterns, at-risk individuals may not be identified in time for preventive action.

To address these gaps, this project proposes a machine learning-based framework that uses advanced algorithms like Random Forest and XGBoost to analyze complex feature sets and predict drug consumption behavior more accurately. The system aims to determine not only whether drug use has occurred but also the approximate timing of the usage. Healthcare professionals, policymakers, and public safety organizations in deploying proactive and personalized interventions.

2.2 PROJECT DETAILS

The project titled **"Dynamic Prediction of Drug Consumption Patterns Using Machine Learning Techniques"** is designed to address the growing need for intelligent, data-driven systems that can analyze and predict patterns of drug consumption. The main objective is to develop a highly accurate predictive model that determines whether an individual has consumed drugs, and if so, the recency of consumption—whether it occurred within an hour, a day, or a year. This is achieved by integrating advanced machine learning models with rich, multi-dimensional datasets containing demographic, socio-economic, and psychological factors.

To accomplish this, the system utilizes two powerful ensemble learning algorithms: **Random Forest** and **XGBoost**. These algorithms are selected for their ability to model complex, non-linear relationships and handle noisy, high-dimensional, and imbalanced data effectively. Each model is trained independently on a labeled dataset comprising user profiles with known drug usage histories. The input features include variables such as age, gender, education level, income, psychological traits, and behavioral indicators, allowing the models to uncover nuanced patterns and trends related to substance use.

The project follows a structured pipeline starting from **data preprocessing**, which includes data cleaning, normalization, handling missing values, and feature encoding. Following this, the **feature selection** process ensures that only the most relevant variables are used, enhancing both model performance and interpretability. The next stage involves **training and evaluation** of the models using key performance metrics such as **accuracy, precision, recall, F1-score**, and **confusion matrix**. These metrics allow for comprehensive assessment and comparison of each model's ability to classify users correctly and detect recent or past drug consumption events.

A major strength of this project is its focus on interpretability and real-world applicability. Feature importance analysis is used to identify the key factors influencing consumption behavior, helping healthcare professionals and policy designers make informed decisions.

In summary, this project combines robust data handling, advanced machine learning algorithms, and targeted evaluation strategies to deliver a dynamic prediction

system. It aims to assist in early identification of at-risk individuals, inform preventive and rehabilitative measures, and ultimately contribute to reducing the social and health burdens associated with drug abuse.

2.3 DATA

The effectiveness and reliability of any machine learning-based prediction system heavily depend on the quality, diversity, and relevance of the dataset used for training, validation, and testing. In the context of predicting drug consumption patterns, data must capture a wide spectrum of individual characteristics and behavioral indicators to accurately reflect real-world scenarios. For this project, the dataset comprises a rich blend of **demographic**, **socio-economic**, and **psychological features** linked with individual drug use behavior.

The dataset used includes real-world survey data that has been pre-compiled and labeled to indicate whether the individual has consumed specific substances, along with the **recency of consumption**—categorized into classes such as "Never Used," "Used in Last Year," "Used in Last Month," "Used in Last Week," and "Used in Last Day." The target variable is derived from these labels, enabling both classification and recency-based prediction tasks.

Key features in the dataset include:

- **Demographic attributes**: age, gender, ethnicity, education level, employment status
- **Socio-economic indicators**: income group, housing conditions, region
- **Psychological traits**: scores from personality assessments such as the Big Five (Neuroticism, Extraversion, Openness, Agreeableness, Conscientiousness)
- **Behavioral indicators**: impulsivity, sensation-seeking, and risk-taking tendencies

The data preprocessing pipeline includes several essential steps such as **missing value imputation**, **feature encoding** for categorical variables, **normalization**, and **outlier removal** to ensure the dataset is clean and ready for model training. Feature engineering is also employed to create derived variables or indicators that may enhance predictive accuracy—such as a "stress level index" or "peer influence score."

To ensure generalization and avoid model overfitting, the dataset is split into **training (70%)**, **validation (15%)**, and **testing (15%)** sets. This division allows the models to learn patterns from historical data, fine-tune hyperparameters for optimal performance, and evaluate results on unseen data. In addition, **stratified sampling** is used to maintain class balance in the splits, especially since drug use data often exhibits imbalanced class distributions.

The overall quality and comprehensiveness of this dataset make it suitable for building a robust drug consumption prediction system. By capturing not only whether drug use has occurred but also the psychological and environmental context around it, the system gains the ability to generate deeper insights and more accurate predictions.

User Interface (UI) :

A flexible, interactive, and user-friendly User Interface (UI) has been developed as part of the drug consumption prediction system to enhance usability and accessibility for both technical users and public health professionals. The UI is built using Django, a high-level Python-based web framework that seamlessly integrates the machine learning backend with a clean and responsive front-end experience.

The interface is designed to support the primary functionality of the system: allowing users to input individual-level data—such as demographic, psychological, and behavioral attributes—and receive an immediate prediction regarding potential drug consumption status and the recency of use. This dynamic prediction output is based on real-time processing by the trained machine learning models (Random Forest and XGBoost) hosted in the backend.

The web page is designed with simplicity in mind, consisting of clear sections for data entry, prediction output, and model confidence scores. Users can fill in relevant fields such as age, gender, education level, personality traits (e.g., neuroticism, extraversion), impulsivity scores, and social behavior indicators, and then submit the form.

Upon submission, the data is sent to the backend, where preprocessing, feature extraction, and model inference are carried out. The result—classification of drug use (e.g., “Never Used”, “Used in Last Month”, “Used in Last Day”)—is then displayed alongside a confidence score indicating the model’s certainty.

To promote user understanding, the UI also includes informative tooltips, example entries, and a brief overview explaining how machine learning models identify drug usage patterns. This educational component enhances transparency and builds user trust in the system's predictions.

In summary, the user interface acts as a critical bridge between advanced machine learning technologies and end-users, delivering a responsive and informative experience. It simplifies complex backend processes and makes predictive insights accessible in a practical, real-world setting, thus empowering stakeholders to make data-informed decisions regarding drug abuse detection and intervention strategies.

Chapter 3

COMPUTATIONAL ENVIRONMENT

3.1 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating costs, planning team activities, performing tasks and tracking the team's and tracking the team's progress throughout the development activity.

Platform	**:**	Windows 10
Programming Environment (IDE)	**:**	Visual Studio Code
UML Design	**:**	Visual Paradigm
Coding Language	**:**	Python
Frontend	**:**	Streamlit (Python script)

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented. Ambiguities or omissions in these requirements can lead to significant cost overruns and delays during the project lifecycle.

Processor	**:**	Inter i3 or Higher
Speed	**:**	1.6 ghz or Higher
RAM	**:**	4 GB or Higher
Storage Disk	**:**	5 GB or Higher

3.3 FUNCTIONAL REQUIREMENTS

The Landslide Detection System incorporates several key functional requirements to ensure smooth operation and effective results. The system begins by allowing users to register and log in securely using their credentials. This authentication step ensures that only authorized users can access the prediction functionality. Once logged in, users are directed to the main interface where they can interact with the core features of the application.

A central function of the system is the ability to upload satellite images. Users can select images from their local device in commonly supported formats such as JPG and PNG. The system then validates the uploaded file to confirm it is a compatible image before proceeding. This step is essential to avoid processing errors and ensures that only appropriate data is fed into the prediction model.

Once the image is uploaded, the backend AI model processes it using advanced deep learning techniques. The system quickly analyzes the image and provides an output classifying it as either “Landslide” or “Non-Landslide.” The result is displayed clearly on a separate screen, including the image and the prediction label, enabling users to interpret and act on the information with ease.

In addition to prediction, the system allows users to re-upload a new image for further analysis, making it flexible for multiple inputs. It also includes robust error handling to manage issues like unsupported file types or upload failures. The user interface is designed to be responsive and intuitive, providing a smooth experience across devices, making this system a valuable tool for risk assessment and disaster management initiatives.

3.4 SOFTWARE FEATURES

Python:

The most abundantly used general level programming language. It is used for both a small scale and big scale systems. It can easily be interpreted. It is said to support multiple programming paradigms. It includes features of procedural, object-oriented, and functional programming together. It is already garbage-collected which makes it more efficient.

It is designed to be easy to read, write and understand, and it is popular for its simplicity, versatility and strong support for various programming paradigms such as procedural, object-oriented and functional programming. Some of the main features of python are as follows; Readability, Portability, Libraries and frameworks, Interpreted. Python is used in a wide range of applications, such as web development, data analysis, scientific computing, machine learning, and artificial intelligence. Its versatility, ease of use, and robust community support make it a popular choice for developers.

Advantages:

i. Simple and easy to learn: Python has a simple and easy-to-understand syntax that makes it easy for beginners to learn. Its code is readable and intuitive, which makes it easy to write and maintain. Large standard library: Python has a large and comprehensive standard library that provides a wide range of modules and functions, including tools for data processing, network programming, web development, and more.

ii. This saves time for developers and reduces the amount of code they need to write, which helps in efficient working of the code.

iii. Cross-platform compatibility: Python is a platform-independent language, which means that Python code can run on any platform, including Windows, macOS, Linux, and UNIX.

iv. Rapid prototyping: Python's simplicity and ease of use make it an ideal language for rapid prototyping. It allows developers to quickly write and test code without worrying about the complexities of other languages.

v. High-level language: Python is a high-level language that allows developers to write complex programs using fewer lines of code. This saves time and makes the code easier to read and maintain.

vi. Extensible: Python can be easily extended using third-party libraries and frameworks, such as NumPy, Pandas, TensorFlow, Django, and Flask. This makes it easy to add new features and functionality to the language.

NumPy:

NumPy is a Python library that provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy stands for "Numerical Python" and is widely used

for scientific computing, data analysis, and machine learning. This basically helps us deal with large datasets, matrices, and multi- dimensional arrays.

Sample NumPy code:

```
import numpy as np
# Generate a random array of integers arr = np.random.randint(1, 100, 10)
# Find the maximum value in the array max_val = np.max(arr)
# Print the array and its maximum value print("Array:", arr)
 print("Maximum value:", max_val)
```

Sklearn:

Scikit-learn, also known as sklearn, is a popular Python library for machine learning. It provides a wide range of algorithms and tools for data preprocessing, model selection, feature extraction, and more. It is a library used in machine learning in python programming language. It mainly helps us with the classification of data, regression of models, and in clustering algorithms. These algorithms include SVM, logistic regression etc.

Streamlit:

Streamlit is a Python library that allows developers to quickly and easily create web applications for machine learning and data science projects. It provides a simple and intuitive interface for building interactive web-based applications with minimal coding effort. Some of the key features include Easy to use, Interactive visualizations, Real-time updates, sharing. Streamlit is a powerful tool for building web-based applications for machine learning and data science projects, and its ease of use makes it an attractive option for developers with varying levels of experience.

Sample Streamlit Code:

```
import streamlit as st
user_input = st.text_input("Enter some text") output = st.button("Submit")
if output:
st.write("You entered:", user_input)
```

Visual Studio Code:

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. It provides an extensive set of features and extensions that can be customized to fit the needs of developers across a wide range of programming languages and frameworks. VS Code is a powerful code editor that is highly customizable and provides a range of features and tools to help developers work more efficiently. Apart from performing its core tasks, the system also includes important software features to enhance performance and usability:

- **Performance**: Capable of quickly processing large datasets and generating results in real time.
- **Scalability**: Designed to support multiple users and a growing dataset without performance issues.
- **Usability**: Simple and user-friendly interface built using Streamlit, making it easy for users to input data and view results.
- **Maintainability**: Well-structured code and documentation make future updates and debugging easier.
- **Security**: Strong data protection mechanisms are in place to ensure user privacy and data safety.

Chapter 4

FEASIBILITY STUDY

Feasibility study is the initial design stage of any project, which brings together the elements of knowledge that indicate if a project is possible or not. A feasibility study includes an estimate of the level of expertise required for a project and who can provide it, quantitative and qualitative assessments of other essential resources, identification of critical points, a general timetable, and a general cost estimate.

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. For feasibility analysis, some understanding of the major requirements for the system is essential. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization.

The feasibility of a proposed solution is evaluated in terms of its components.

4.1 TECHNICAL FEASIBILITY

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

- Analyses the technical skills and capabilities of the project development team members.
- Determines whether the relevant technology is stable and established.
- Ascertains that the technology chosen for project development has a large number of users so that they can be consulted when problems arise or improvements are required.

4.2 OPERATIONAL FEASIBILITY

Operational feasibility assesses the extent to which the required system performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (project development team) and involves visualizing whether the project will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority.
- Determines whether the solution suggested by the project development team is acceptable.
- Analyses whether users will adapt to new software.
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

4.3 ECONOMIC FEASIBILITY

Economic feasibility determines whether the required system is capable of generating financial gains for an organization. It involves the cost incurred on the project development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out project development. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on project development to produce long-term gains for an organization.
- Cost required to conduct full software investigation (such as requirements elicitation and requirement specification).

Chapter 5

SYSTEM ANALYSIS

5.1 EXISITNG SYSTEM

Existing drug consumption prediction systems primarily rely on traditional machine learning algorithms such as Logistic Regression, Support Vector Machines (SVM), Decision Trees, Naive Bayes, and k-Nearest Neighbors (k-NN). These models are typically trained on datasets containing demographic, socio-economic, and psychological factors to identify consumption patterns among individuals.

While these techniques offer moderate effectiveness for basic analysis, they often fall short when it comes to delivering more detailed or nuanced predictions. One significant limitation is their inability to accurately predict the precise timing or frequency of drug use. Additionally, these models struggle to handle challenges such as imbalanced datasets and noisy input data, which are common in real-world healthcare and behavioral datasets.

Another key limitation is the generalization capability of these algorithms. In diverse or dynamic scenarios—such as varying populations or changing behavioral trends—their performance may degrade, leading to unreliable predictions. As a result, while traditional methods are useful for foundational insights, they lack the advanced analytical capabilities necessary to deliver comprehensive and reliable interpretations of drug consumption behaviors.

5.1.1 Disadvantages of Existing System

There are some disadvantages of the existing system for drug prediction, which relies on separate models for each specific disease, include:

- **Limited Predictive Accuracy:** Traditional machine learning models like Logistic Regression, SVM, Decision Trees, and Naive Bayes provide only moderate accuracy. They struggle to make fine-grained predictions, such as the exact timing or frequency of drug use.
- **Poor Handling of Imbalanced Data:** Drug consumption datasets are often imbalanced, with fewer instances of drug users compared to non-users. This

leads to biased predictions, where models may fail to accurately detect high-risk individuals.

- **Sensitivity to Noisy Inputs:** Real-world data can be noisy or incomplete. Traditional models are not robust enough to handle missing values or inconsistent entries, resulting in degraded performance.
- **Limited Generalization:** These models may not perform well across different populations or environments, making it difficult to apply them in diverse real-world scenarios.
- **Lack of Contextual Understanding:** Traditional approaches do not consider contextual or temporal factors effectively. They often ignore sequences or patterns of behavior over time, which are critical in understanding drug usage trends.
- **Minimal Adaptability:** Existing systems lack adaptability to evolving behavioral patterns or new types of substances, limiting their usefulness in dynamic and changing environments.
- **Basic Insight Generation:** These models offer only high-level insights and are not capable of providing deep, actionable intelligence that is essential for prevention, intervention, or personalized treatment strategies.

5.2 PROPOSED SYSTEM

The proposed system leverages advanced machine learning techniques, including **XGBoost**, **Random Forest**, and **hybrid models**, to enhance the prediction of drug consumption patterns. These methods are chosen for their ability to handle complex data structures and extract meaningful insights from various types of inputs.

By analyzing a combination of **demographic**, **socio-economic**, and **psychological** factors, the system effectively identifies patterns of drug use. Unlike traditional models, it captures **non-linear relationships** and interdependencies among features, allowing for a deeper understanding of the underlying behavior.

A key advantage of this system is its **hybrid approach**, which integrates the strengths of multiple algorithms. This not only improves predictive performance but also helps to effectively manage **imbalanced datasets**, a common challenge in drug consumption studies.

The model's performance is rigorously evaluated using established metrics such as **accuracy**, **precision**, **recall**, and **F1 score** to ensure reliability and robustness.

Ultimately, the proposed system aims to provide **accurate and actionable insights** for **healthcare professionals**, **policymakers**, and **public health organizations**. It supports early identification, targeted interventions, and informed decision-making in addressing drug consumption trends within populations.

5.2.1 Advantages of Proposed System

The proposed system for the Multi Drug Prediction Model utilizing machine learning techniques offers several advantages. Here are some of the advantages:

- **Improved Predictive Accuracy:** The use of advanced algorithms like XGBoost and Random Forest allows the system to achieve higher accuracy in identifying drug consumption patterns compared to traditional models.
- **Effective Handling of Complex Data:** The system captures non-linear and intricate relationships among demographic, socio-economic, and psychological factors, leading to more meaningful predictions.
- **Robustness Against Imbalanced Data:** Through hybrid modeling and algorithmic optimization, the system effectively addresses class imbalance, ensuring better performance even with skewed datasets.
- **Enhanced Evaluation Metrics:** Model performance is measured using precision, recall, F1 score, and accuracy, providing a comprehensive assessment of effectiveness.
- **Scalability and Adaptability:** The system can be easily scaled and adapted to different datasets or evolving patterns of drug use, making it suitable for various real-world applications.
- **Support for Decision-Making:** It delivers actionable insights to healthcare professionals, policymakers, and public health organizations, enabling informed interventions and strategies.
- **Context-Aware Predictions:** The proposed system considers multiple influencing factors together, offering deeper, context-sensitive analysis rather than basic predictions.

Chapter 6

SYSTEM DESIGN

System design is the blueprint that transforms theoretical concepts into a functional and operational machine learning application. It defines the architecture, data flow, module interactions, and technology stack needed to build a robust, intelligent system for predicting drug consumption patterns. In this project, system design is vital in ensuring that each component—from data preprocessing to user interaction—is efficiently integrated and aligned with the project's objectives of accuracy, responsiveness, and usability.

The design process includes both **logical** and **physical** phases. The **logical design** outlines how user data (age, gender, personality traits, socio-economic status) flows through the system. It defines the structure of the input, the sequence of preprocessing steps (like encoding, normalization), and how these inputs feed into the machine learning models (Random Forest and XGBoost). This design phase also captures how model predictions—such as drug consumption status and recency—are processed and returned to the user.

The **physical design** focuses on the implementation specifics, including technologies like **Python**, **Django**, **Pandas**, **Scikit-learn**, and **XGBoost libraries**. The user interface is developed using Django to provide a seamless interaction layer, while backend model processing is executed in Python. The prediction engine is hosted on a secure server, ensuring real-time feedback and scalability.

The system is modularized into the following components:

- **User Input Interface**: Collects demographic, psychological, and behavioral data from the user.
- **Data Preprocessing Module**: Cleans, encodes, and prepares the data for analysis.
- **Feature Selection Engine**: Identifies the most relevant variables for accurate predictions.
- **Model Processing Unit**: Applies trained Random Forest and XGBoost models for classification.

- **Result Display Module**: Returns predictions and confidence levels to the user interface.
- Non-functional requirements such as **scalability**, **real-time performance**, **accuracy**, **data privacy**, and **ease of use** are also considered. The system is designed to accommodate growing datasets, ensure low-latency responses, and provide meaningful predictions under varying conditions.

To illustrate the system design, **UML diagrams** such as **use case diagrams** (for user interaction), **sequence diagrams** (to visualize request flow), **class diagrams** (for data structures), and **component diagrams** (for modular design) are used.

In conclusion, system design plays a central role in guiding the development of a reliable, user-centric prediction system. By breaking down complex functionality into structured modules and ensuring smooth communication between them, the design sets the stage for successful deployment and real-world application of drug consumption prediction using machine learning.

6.1 SYSTEM ARCHITECTURE

The architecture of the proposed **drug consumption prediction system** follows a modular and layered design to ensure scalability, adaptability, and high predictive accuracy. It is structured to handle diverse data inputs, perform efficient processing, and deliver real-time results through an intuitive user interface. At its core, the architecture integrates **Random Forest** and **XGBoost** models, which serve as the primary engines for classification and prediction.

The system begins with the **User Interface Layer**, where users input relevant personal data including demographic attributes (age, gender, education), psychological traits (neuroticism, extraversion, etc.), and behavioral indicators (impulsivity, risk-taking behavior). This layer is designed for simplicity and ease of use, ensuring accessibility for both technical and non-technical users.

Upon submission, the input data flows into the **Preprocessing Layer**, which performs crucial tasks such as **missing value imputation**, **feature encoding**, **data**

normalization, and **scaling**. This step ensures that the data is clean, standardized, and suitable for machine learning models.

The cleaned data is then passed into the **Model Layer**, which houses the trained **Random Forest** and **XGBoost** classifiers. These models analyze the input features to predict not only whether the user has consumed drugs, but also the **recency** of usage—categorized into "Never Used," "Used in Last Year," "Used in Last Month," and so on. The model output is a predicted class along with a **confidence score**, representing the certainty of the prediction.

Next, the **Result Display Layer** presents the output in a clear and interpretable manner. It shows the predicted drug consumption status and recency, along with the confidence level. This immediate feedback helps users or stakeholders make informed decisions.

The system also incorporates a **Data Storage Layer**, responsible for securely managing training datasets, user interaction logs (if enabled), and prediction results. This layer ensures data integrity and supports future retraining and analysis.

In summary, the layered and modular architecture ensures that the system remains flexible, scalable, and maintainable while delivering accurate and actionable insights for drug consumption prediction.

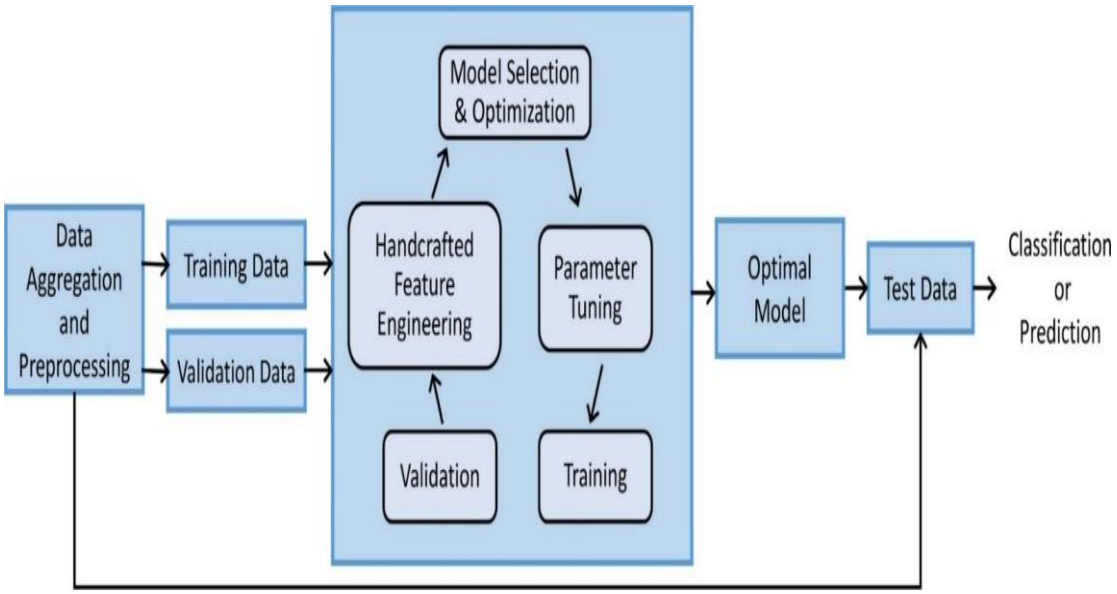


Fig. 6.1: System Architecture

6.2 UML DIAGRAMS

What is UML?

The Unified Modeling Language (UML) is a standard language for specifying, Visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process.

The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. UML is simply anther graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

Advantages:

- Provides standard for software development.
- Development time is reduced.
- The past faced issues by the developers are no longer exists.
- Has large visual elements to construct and easy to follow.
- To establish an explicit coupling between concepts and executable code.
- To creating a modeling language usable by both humans and machines UML defines several models for representing systems.

In this project six basic UML diagrams have been explained:

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram

6.2.1 Use Case Diagram

Use Case diagram is a type of behavioral diagram defined by and created from a use case analysis. A use case diagram is a type of UML (Unified Modeling Language) diagram that is used to represent the interactions between the users (or actors) and the system being designed. It is a high- level view of a software system that illustrates the various ways in which users interact with the system to achieve specific goals. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements.

The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined.

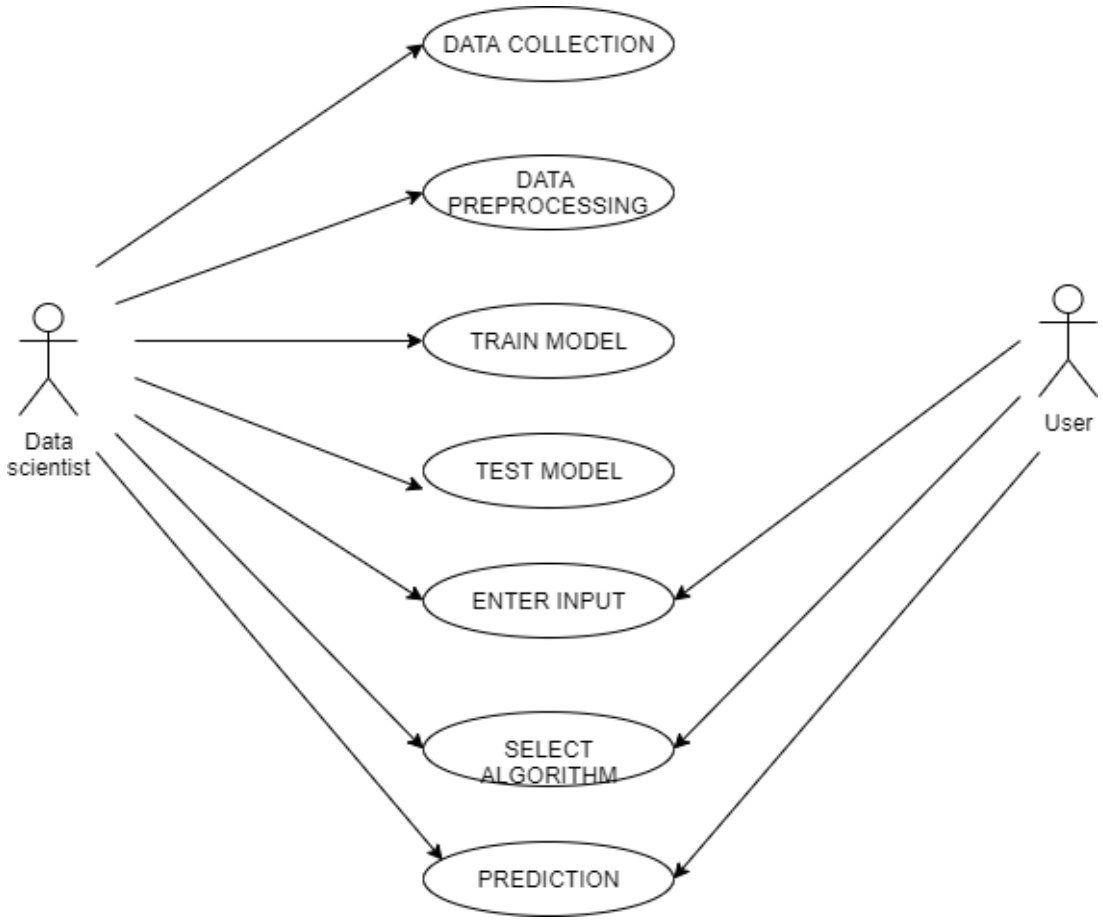


Fig. 6.2.1:Use Case Diagram

6.2.2 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints.

It is also known as a structural diagram. Class diagrams are useful for visualizing the structure of a system and for identifying the relationships between the classes. They can help to identify potential errors or omissions in the system design and can facilitate communication and collaboration between team members by providing a shared understanding of the system being designed.

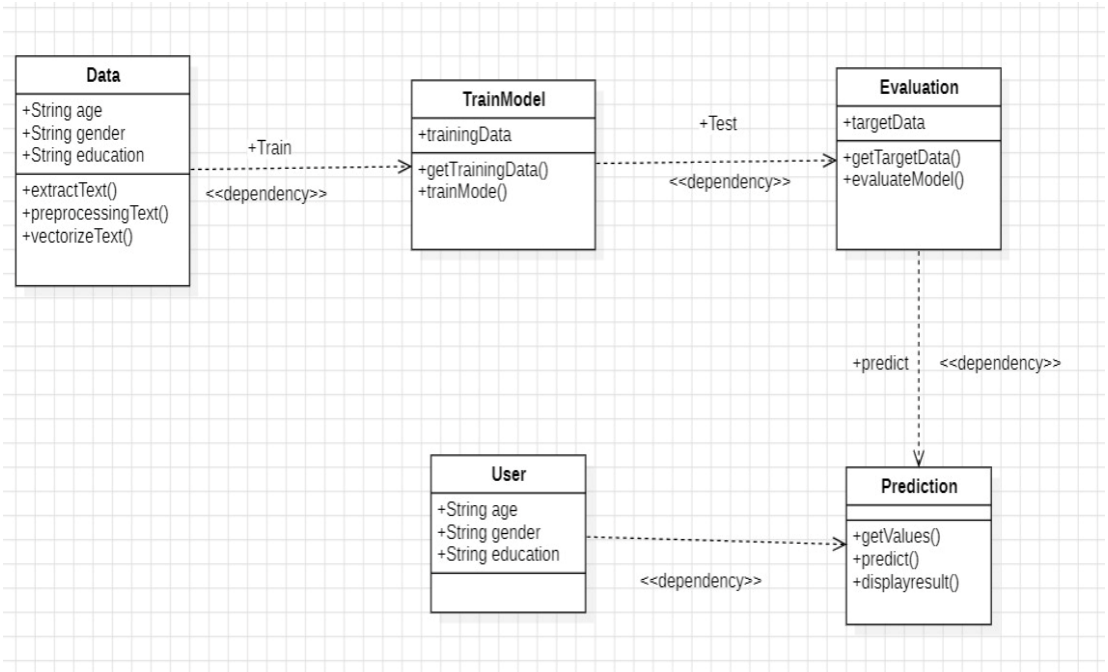


Fig. 6.2.2: Class Diagram

6.2.3 Sequence Diagram

A Sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behavior of a system. A sequence diagram is the most used interaction diagram. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.

A Sequence Diagram consists of the following components: Objects, Lifelines, Messages.

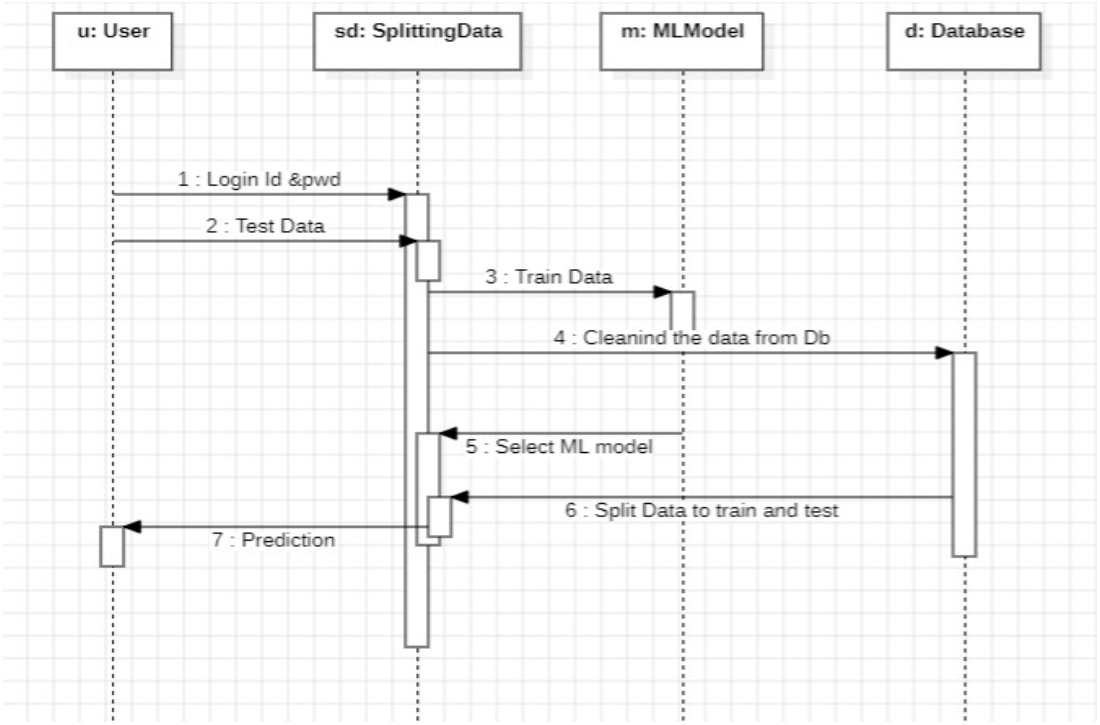


Fig. 6.2.3: Sequence Diagram

6.2.4 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. An activity diagram is a type of UML (Unified Modeling Language) diagram that models the flow of activities or tasks in a system or process. It is a high-level view of the system that illustrates the steps that are required to accomplish a specific task or objective. Activity diagrams are useful for modeling the flow of activities or tasks in a system or process and for identifying potential errors or inefficiencies in the process. They can help to ensure that all the necessary activities are included in the process and that the process is designed to handle all possible scenarios. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

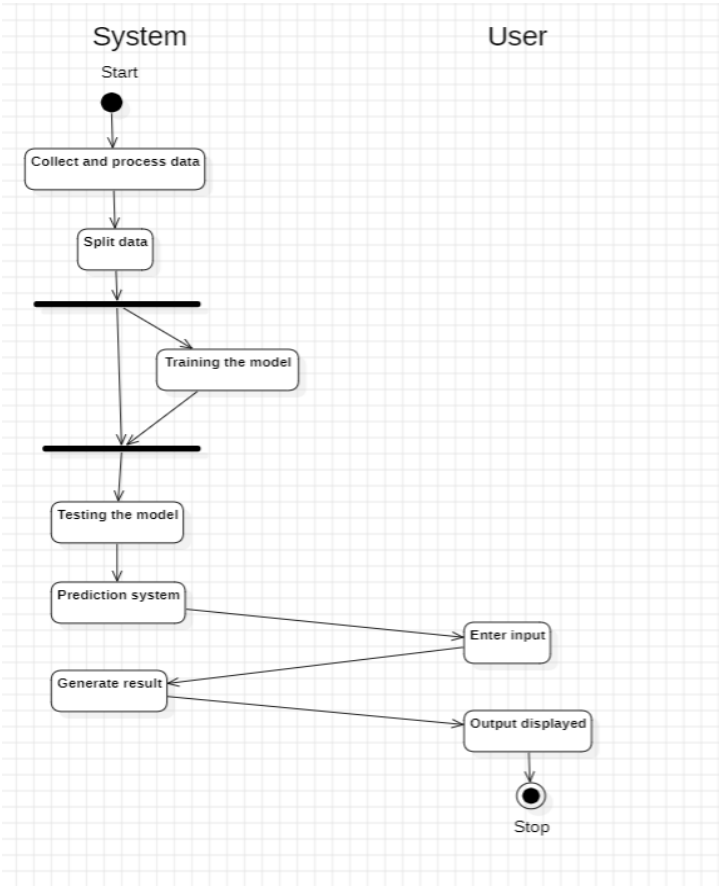


Fig. 6.2.4: Activity Diagram

6.2.5 Component Diagram

A component diagram is a type of UML (Unified Modeling Language) diagram that shows the structure of the components of a system and their interrelationships. It is a static view of the system that illustrates the physical and logical components of the system and how they work together. In a component diagram, components are represented as rectangles with the component name written inside. The interfaces of the component are shown as ports or small squares on the edges of the component.

The relationships between the components are shown using connectors, such as associations or dependencies. Component diagrams are a powerful tool for system

design, architecture, integration, and maintenance. They provide a clear and concise view of the system components and their relationships, which helps to ensure that the system is designed, built, and maintained correctly. Component diagrams are useful for visualizing the structure of a system and for identifying the relationships between the components.

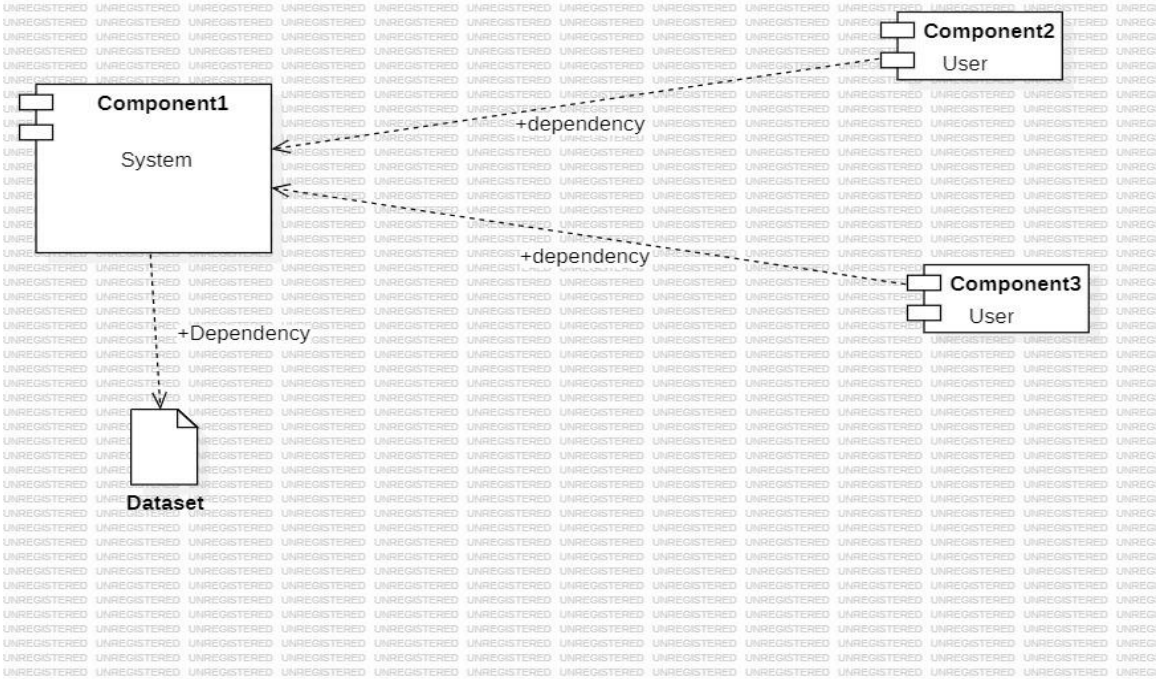


Fig. 6.2.5: Component Diagram

6.2.6 Deployment Diagram

The Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application. A deployment diagram is a type of UML (Unified Modeling Language) diagram that shows the physical deployment of software components and their relationships with hardware components in a system.

It is a static view of the system that illustrates the physical architecture of the system and how the components are deployed on hardware devices or servers. In a deployment diagram, components are represented as nodes or boxes, and the relationships between the components are shown using connectors, such as associations or dependencies. Nodes represent physical devices, such as servers, and components represent software modules or applications. Deployment diagrams are useful for visualizing the physical deployment of software components in a system and for identifying potential issues or constraints in the deployment.

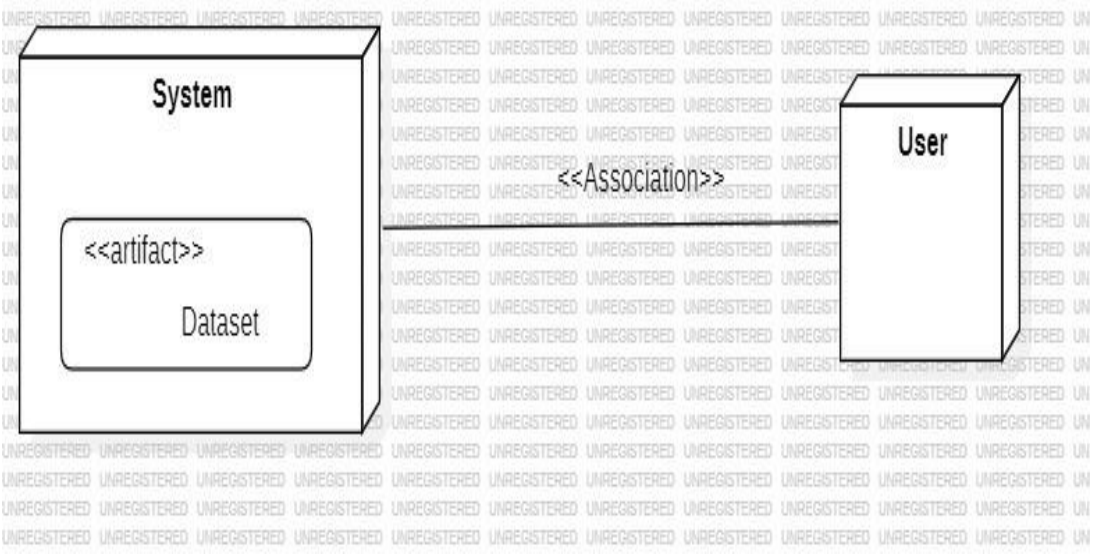


Fig. 6.2.6: Deployment Diagram

Chapter 7

SYSTEM IMPLEMENTATION

7.1 ALGORITHM

The process of disease prediction involves several steps. The first step is data collection, where historical patient data for a particular disease is collected from various sources such as hospitals and some healthcare departments. The data is then pre-processed, which involves cleaning, filtering, and normalization to ensure that it is suitable for analysis.

Next, feature selection is performed to identify the most relevant features that can be used to predict disease. These features could include Age, Gender, Glucose levels, etc. Machine learning algorithms such as regression, random forest and XGBoost are then applied to the selected features to predict the desired result. The performance of the algorithms is evaluated using various metrics such as accuracy, precision, recall.

7.1.1 XGBoost Classifier

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

XGBoost is a robust machine-learning algorithm that can help you understand your data and make better decisions. XGBoost is an implementation of gradient-boosting decision trees. It has been used by data scientists and researchers worldwide to optimize their machine-learning models. It is designed for speed, ease of use, and performance on large datasets. It does not require optimization of the parameters or tuning, which means that it can be used immediately after installation without any further configuration.

Execution speed: Execution speed is crucial because it's essential to working with large datasets. When you use XGBoost, there are no restrictions on the size of your dataset, so you can work with datasets that are larger than what would be possible with other algorithms.

Model performance: Model performance is also essential because it allows you to create models that can perform better than other models. XGBoost has been compared to different algorithms such as random forest (RF), gradient boosting machines (GBM), and gradient boosting decision trees (GBDT). These comparisons show that XGBoost outperforms these other algorithms in execution speed and model performance.

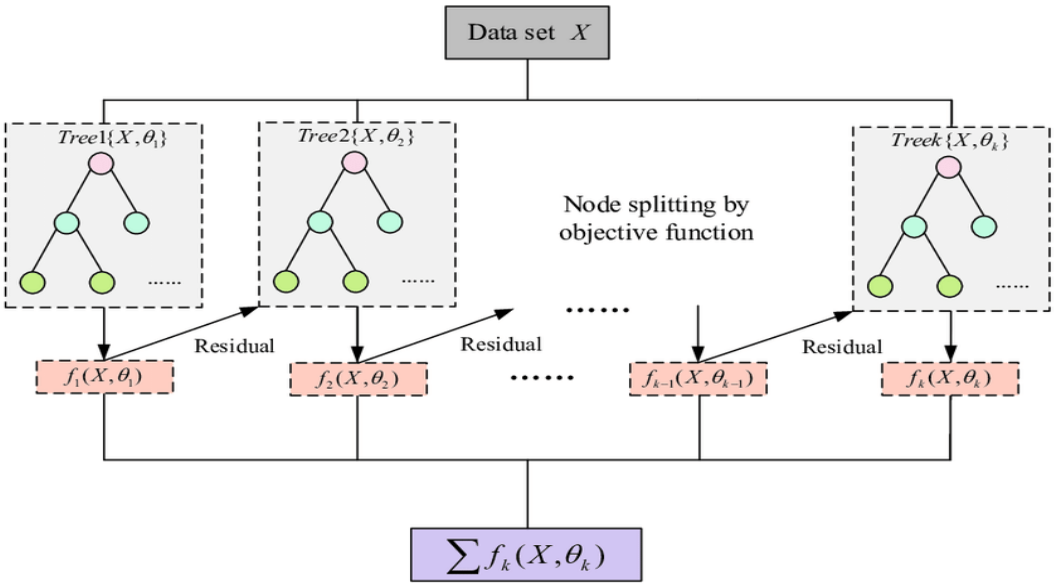


Fig. 7.1.1.: XGBoost Classifier Diagram

7.1.2 Random Forest

Random Forest Classifier is an ensemble learning algorithm that builds a multitude of decision trees and combines their outputs to provide a more accurate prediction. Each decision tree in the random forest is trained on a random subset of the training data and a random subset of the features. This helps to reduce overfitting and increase the diversity of the trees in the forest. Once the trees are built, the random forest combines their predictions through voting to produce the final prediction.

Random Forest Classifier is a powerful algorithm that is widely used in machine learning for classification tasks. It is particularly useful when dealing with high-dimensional datasets, as it can handle a large number of features without overfitting. Additionally, it is robust to noisy data and can handle missing values.

Random Forest Classifier also provides important insights into feature importance, which can help in understanding the underlying relationships between the features and the target variable. Overall, the Random Forest Classifier is a reliable and efficient algorithm for accurate prediction in many classification problems.

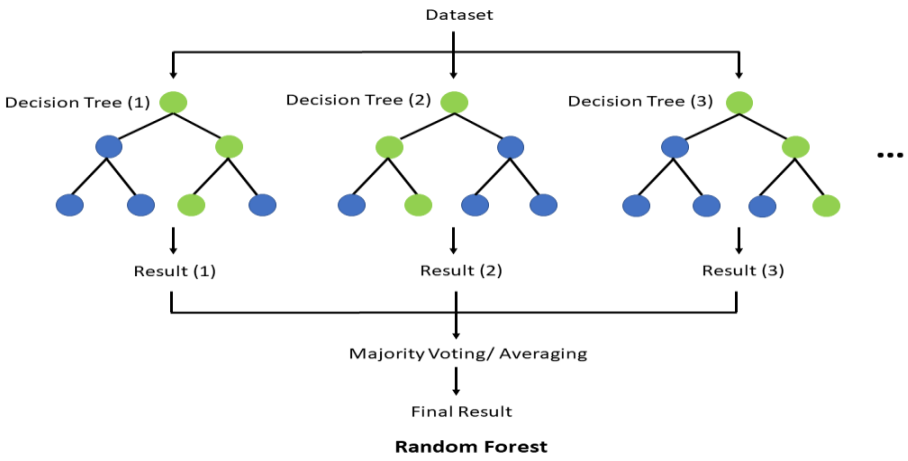


Fig. 7.1.2.: Random Forest Classifier Diagram

7.2 WORKING AND MODULE EXPLANATION

This project involves mainly of 6 modules. They are:

- Data Collection and Processing
- Splitting Data
- Training the Model
- Model Evaluation
- Building the Prediction System

7.2.1: Module Description

Data Collection and Processing

Student dropout is a major concern for educational institutions is collected from the online sources such as Kaggle and other sites in the csv format. various factors such

as academic performance, attendance, socio-economic background, and student engagement Collected data is processed to remove any missing values, noises, duplicates in the data set. Processing of data is required for improving the accuracy of the model.

Splitting Data

After the collection of the dataset, we split the dataset into Training Data and Testing Data. The Training Dataset is used for training the prediction model and Testing Dataset is used for evaluating the prediction model. For this project the Data set is split as 80% and 20%. The 80% is used for training the prediction model and 20% is used for testing the prediction model.

Training the Model

For Training the prediction model 80% of the collected dataset is used. Logistic Regression Machine Learning Algorithm is used to train the prediction model which is more accurate. It is used for predicting the categorical. Random forest predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

Model Evaluation

For Evaluating the prediction model 20% of the collected dataset is used. The metric used for evaluating the model in the project is accuracy score. Accuracy score is the percentage of correctly predicted instances in the data set. It is calculated by dividing the number of correctly predicted instances by the total number of instances in the data set.

Building the Prediction System

The Prediction System is built using the Trained and Evaluated model. Building process of the Heart Disease Prediction System includes all the above steps such as Data Collection and Processing, splitting of the data, Training the model, Model Evaluation. The Combination of all these processes forms the Prediction System. s a chance of affecting by heart attack.

User Interface

A flexible User Interface is developed for the use of Heart Disease Prediction

System. This user interface is adaptable and user friendly. Streamlit, an open-source framework for Machine learning is used to create the user interface of Student dropout is a major concern for educational institutions using Machine Learning.

The User Interface created for Prediction of Student dropout is a major concern for educational institutions using Machine Learning is a webpage which takes the parameters from the user and sends to the prediction system and the result is displayed to the user. The webpage also tends to raise awareness about the heart disease among the individuals

Chapter 8

SOURCE CODE

PYTHON CODE:

```
# Drug Prediction Using Machine Learning with Python

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
!pip install xgboost
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
df=pd.read_csv(r"C:\Users\samba\Music\DRUG
PREDICTION\DATASET\Drug_Consumption.csv")

#df.info()
#df.describe(include='all')
#df.isnull().sum()
#df["VSA"].unique()

df=df.drop(columns=[‘ID’,”Alcohol”,”Amyl”,”Benzos”,”Caff”,”Cannabis”,”Choc”,”
Coke”,”Crack”,”Ecstasy”,”Heroin”,”Legalh”,”Meth”,”Mushrooms”,”Nicotine”,”Sem
er”
def drug_encoder(x):
    if x == 'CL0':
        return 0
```

```
    elif x == 'CL1':
        return 1
    elif x == 'CL2':
        return 2
    elif x == 'CL3':
        return 3
    elif x == 'CL4':
        return 4
    elif x == 'CL5':
        return 5
    else:
        return 6

df["VSA"] = df["VSA"].apply(drug_encoder)

numeric_df = df.select_dtypes(include='number')
correlation_matrix = numeric_df.corr()

plt.figure(figsize=(20,15))
sns.heatmap(numeric_df.corr(),annot=True)
plt.title('Heatmap of Correlations',fontsize=15)
plt.show()

df["Age"].unique()
df["Education"].unique()
df["Country"].unique()
df["Ethnicity"].unique()

def age_encoder(x):
   if x == '18-24':
      return "young"
   elif x == '25-34':
      return "middle young"
```

```
    elif x == '35-44':
        return "early middle"
    elif x == '45-54':
        return "middle age"
    elif x == '55-64':
        return "adults"
    else:
        return "old adults"

df["Age"] = df["Age"].apply(age_encoder)

#df["Age"].value_counts()
#df["Gender"].value_counts()
#df["Education"].value_counts()
#df["Country"].value_counts()


#df["Ethnicity"].value_counts()

sns.countplot(x=df["Age"], hue=df['VSA'].astype(str))
plt.show()


sns.countplot(x=df["Gender"], hue=df['VSA'].astype(str))
plt.show()


#df.columns.to_list()

#df.isnull().sum()


sns.countplot(x=df["VSA"])
```

```
plt.xticks(rotation = 90)
plt.show()


from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()


df['Age']=le.fit_transform(df['Age'])


df['Gender']=le.fit_transform(df['Gender'])

df['Education']=le.fit_transform(df['Education'])


df['Country']=le.fit_transform(df['Country'])


df['Ethnicity']=le.fit_transform(df['Ethnicity'])

#df["VSA"].value_counts()

from sklearn.utils import resample
# Separate majority and minority classes
df_majority = df[df['VSA']== 0]
df_minority = df[df['VSA']== 6]
df_minority1 = df[df['VSA']== 4]
df_minority2 = df[df['VSA']== 5]
df_minority3 = df[df['VSA']== 3]
df_minority4 = df[df['VSA']== 2]
df_minority5 = df[df['VSA']== 1]
```

```
# Downsample majority class and upsample the minority class
df_minority_upsampled = resample(df_minority,
replace=True,n_samples=500,random_state=100)
df_minority1_upsampled = resample(df_minority1,
replace=True,n_samples=500,random_state=100)
df_minority2_upsampled = resample(df_minority2,
replace=True,n_samples=500,random_state=100)
df_minority3_upsampled = resample(df_minority3,
replace=True,n_samples=500,random_state=100)
df_minority4_upsampled = resample(df_minority4,
replace=True,n_samples=500,random_state=100)
df_minority5_upsampled = resample(df_minority5,
replace=True,n_samples=500,random_state=100)
df_majority_downsampled = resample(df_majority,
replace=False,n_samples=500,random_state=100)

# Combine minority class with downsampled majority class
df_balanced =
pd.concat([df_minority_upsampled,df_minority1_upsampled,df_minority2_upsample
d,df_minority3_upsampled,df_minority4_upsampled,df_minority5_upsampled,df_ma
jority_downsampled])

# Display new class counts
df_balanced['VSA'].value_counts()

sns.set(style="whitegrid")
plt.figure(figsize=(10, 5))
ax = sns.countplot(x="VSA", data=df_balanced,
palette=sns.color_palette("cubehelix", 4))
plt.xticks(rotation=90)
plt.title("Class Label Counts", {"fontname":"fantasy", "fontweight":"bold",
"fontsize":"medium"})
```

```
plt.ylabel("count", {"fontname": "serif", "fontweight":"bold"})
plt.xlabel("Class", {"fontname": "serif", "fontweight":"bold"})
df=df_balanced

x = df.drop(columns=['VSA'])
y = df['VSA']

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier

dept = [1, 5, 10, 50, 100, 500, 1000]
n_estimators =  [20, 40, 60, 80, 100, 120]
param_grid={'n_estimators':n_estimators , 'max_depth':dept}
clf = RandomForestClassifier()
model = GridSearchCV(clf,param_grid,scoring='accuracy',n_jobs=-1,cv=5)
model.fit(x_train,y_train)
print("optimal n_estimators",model.best_estimator_.n_estimators)
print("optimal max_depth",model.best_estimator_.max_depth)
optimal_max_depth = model.best_estimator_.max_depth
optimal_n_estimators = model.best_estimator_.n_estimators
from sklearn.metrics import accuracy_score
#training our model for max_depth=500,n_estimators = 60
clf = RandomForestClassifier(max_depth = optimal_max_depth,n_estimators =
optimal_n_estimators)
```

```
clf.fit(x_train,y_train)

import pickle
filename = r"C:\Users\samba\Music\DRUG PREDICTION\FRONTEND\rf_drug.pkl"
pickle.dump(clf, open(filename, 'wb'))

pred_test3 =clf.predict(x_test)
test_accuracy3 = accuracy_score(y_test, pred_test3)
pred_train = clf.predict(x_train)
train_accuracy3 =accuracy_score(y_train,pred_train)

print("AUC on Test data is " +str(accuracy_score(y_test,pred_test3)))
print("AUC on Train data is " +str(accuracy_score(y_train,pred_train)))

print("----------------------------")

# Code for drawing seaborn heatmaps
class_names =['Never Used','Used over a Decade Ago','Used in Last Decade','Used in
Last Year','Used in Last Month','Used in Last Week','Used in Last Day']
df_heatmap = pd.DataFrame(confusion_matrix(y_test, pred_test3.round()),
index=class_names, columns=class_names )
fig = plt.figure( )
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")
```

XgBoost

```
import xgboost as xgb
from sklearn.metrics import accuracy_score

model = xgb.XGBClassifier(n_estimators=1000, learning_rate=0.04,random_state=1)
model.fit(x_train, y_train)
```

```
import pickle
filename = r'C:\Users\samba\Music\DRUG
PREDICTION\FRONTENDX_gb_drug.pkl'
pickle.dump(model, open(filename, 'wb'))

pred_test4 =model.predict(x_test)
test_accuracy4 = accuracy_score(y_test, pred_test4)
pred_train = model.predict(x_train)
train_accuracy4 =accuracy_score(y_train,pred_train)

print("AUC on Test data is " +str(accuracy_score(y_test,pred_test4)))
print("AUC on Train data is " +str(accuracy_score(y_train,pred_train)))

print("----------------------------")
# Code for drawing seaborn heatmaps
class_names =['Never Used','Used over a Decade Ago','Used in Last Decade','Used in
Last Year','Used in Last Month','Used in Last Week','Used in Last Day']
df_heatmap = pd.DataFrame(confusion_matrix(y_test, pred_test4.round()),
index=class_names, columns=class_names )
fig = plt.figure( )
heatmap = sns.heatmap(df_heatmap, annot=True, fmt=”d”)
```

INDEX PAGE *(Index.html)*：

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Drug Consumption Prediction</title>
   <link href="{% static 'layout/styles/layout.css' %}" rel="stylesheet" type="text/css"
media="all">
   <style>
      body, html {
         margin: 0;
         padding: 0;
         height: 100%;
      }

      .bgded {
         background-image: url('{% static "images/3.jpg" %}');
         background-size: cover;
         background-position: center;
         background-repeat: no-repeat;
         height: 100%;
         display: flex;
         flex-direction: column;
         justify-content: center;
         align-items: center;
      }
      .heading {
         font-size: 32px;
```

```
            color: white;
            margin-bottom: 20px;
            text-align: center;
        }

        #pageintro {
            background: rgba(255, 255, 255, 0.8);
            padding: 20px;
            border-radius: 10px;
            text-align: center;
        }

        .form-group {
            margin-bottom: 20px;
        }

        .input100 {
            font-size: 18px;
            padding: 10px;
            width: 100%;
            box-sizing: border-box;
        }

        .btn {
            font-size: 20px;
            padding: 10px 20px;
            cursor: pointer;
        }
    </style>
</head>
<body id="top">
    <div class="bgded">
```

```
    <h3 class="heading">Dynamic Prediction of Drug Consumption Patterns Using Machine Learning Techniques</h3>
    <div id="pageintro" class="hoc clear">
      <article>
        <form action="input" method="POST" enctype="multipart/form-data" class="login100-form validate-form">
          {% csrf_token %}
          <div class="form-group">
            <label class="label-input100" for="name">Name</label>
            <input class="input100" type="text" id="name" name="name" style="color: black;" required>
          </div>
          <div class="form-group">
            <label class="label-input100" for="password">Password</label>
            <input class="input100" type="password" id="password" name="password" style="color: black;" required>
          </div>
          <div class="form-group">
            <button type="submit" class="btn">LOGIN</button>
          </div>
        </form>
      </article>
    </div>
  </div>


  <!-- JAVASCRIPTS -->
  <script src="{% static 'layout/scripts/jquery.min.js' %}"></script>
  <script src="{% static 'layout/scripts/jquery.backtotop.js' %}"></script>
  <script src="{% static 'layout/scripts/jquery.mobilemenu.js' %}"></script>
</body>
</html>
```

INPUT PAGE (*Input.html)* :

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

<head>
   <title>Drug Consumption Prediction</title>

   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">

   <!-- Stylesheets -->
   <link rel="stylesheet" type="text/css" href="{% static
'vendor/bootstrap/css/bootstrap.min.css' %}">
   <link rel="stylesheet" type="text/css" href="{% static 'css/main.css' %}">

   <style>
      body {
         background: url("{% static 'images/02.jpeg' %}") no-repeat center center
fixed;
         background-size: cover;
         height: 100vh;
         display: flex;
         align-items: center;
         justify-content: center;
         margin: 0;
      }

      .container-login100 {
         width: 100%;
         max-width: 550px;
         background-color: rgba(0, 0, 0, 0.85);
```

```
    border-radius: 15px;
    padding: 0px;  /* Reduced top space */
    box-shadow: 0 4px 0px rgba(0, 0, 0, 0.5);
    max-height: 80vh;
    overflow-y: auto;
}

.login100-form-title {
    text-align: center;
    color: #000;  /* Black Title */
    margin-bottom: 15px;
    font-weight: bold;
}

.section-title {
    color: #000;  /* Black Subheading */
    text-align: center;
    font-weight: bold;
    margin-bottom: 15px;
    text-transform: uppercase;
    font-size: 18px;
}

.input-field {
    margin-bottom: 15px;
}

.input-field label {
    color: #000;
    font-weight: bold;
    display: block;
    margin-bottom: 5px;
}
```

```
        .input100 {
            background-color: #f0f0f0;
            color: #333;
            border: 1px solid #00ff00;
            border-radius: 5px;
            padding: 10px;
            width: 100%;
        }
        .login100-form-btn {
            background-color: #00ff00;
            color: #000;
            width: 100%;
            border: none;
            border-radius: 25px;
            padding: 12px;
            font-weight: bold;
        }

        .login100-form-btn:hover {
            background-color: #00cc00;
        }
    </style>
</head>
<body>
    <div class="container-login100">
        <div class="wrap-login100">
            <form action='output' method="POST" enctype="multipart/form-data"
class="login100-form validate-form">
                {% csrf_token %}
                <span class="login100-form-title">
                    Drug Consumption Prediction
                </span>
```

```
            <div class="section-title">Provide Details</div>

            <!-- Input Fields with Black Labels -->
            <div class="input-field">
               <label for="AGE">AGE</label>
               <input placeholder="Enter your age" name="AGE" id="AGE"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="Gender">Gender</label>
               <input placeholder="Enter your gender" name="Gender" id="Gender"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="Education">Education</label>
               <input placeholder="Education level" name="Education" id="Education"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="Country">Country</label>
               <input placeholder="Your country" name="Country" id="Country"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="Ethnicity">Ethnicity</label>
               <input placeholder="Your ethnicity" name="Ethnicity" id="Ethnicity"
type="text" class="validate input100">
            </div>
```

```
            <div class="input-field">
               <label for="Nscore">Nscore</label>
               <input placeholder="Enter Nscore" name="Nscore" id="Nscore"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="Escore">Escore</label>
               <input placeholder="Enter Escore" name="Escore" id="Escore"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="Oscore">Oscore</label>
               <input placeholder="Enter Oscore" name="Oscore" id="Oscore"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="AScore">AScore</label>
               <input placeholder="Enter AScore" name="AScore" id="AScore"
type="text" class="validate input100">
            </div>

            <div class="input-field">
               <label for="Cscore">Cscore</label>
               <input placeholder="Enter Cscore" name="Cscore" id="Cscore"
type="text" class="validate input100">
            </div>
            <div class="input-field">
               <label for="Impulsive">Impulsive</label>
```

```
            <input placeholder="Enter Impulsive score" name="Impulsive"
id="Impulsive" type="text" class="validate input100">
          </div>
          <div class="input-field">
            <label for="SS">SS</label>
            <input placeholder="Enter SS score" name="SS" id="SS" type="text"
class="validate input100">
          </div>

          <!-- Algorithm Selection -->
          <div class="input-field">
            <label for="algo">Select Algorithm</label>
            <select class="input100" name="algo">
              <option value='svm'>RF Classifier</option>
              <option value='xgb'>XGBoost Classifier</option>
            </select>
          </div>

          <!-- Predict Button -->
          <div class="container-login100-form-btn">
            <button class="login100-form-btn">Predict</button>
          </div>
        </form>
      </div>
    </div>

    <!-- Scripts -->
    <script src="{% static 'vendor/jquery/jquery-3.2.1.min.js' %}"></script>
    <script src="{% static 'vendor/bootstrap/js/bootstrap.min.js' %}"></script>
    <script src="{% static 'js/main.js' %}"></script>
</body>
</html>
```

OUTPUT PAGE *(output.html)*：

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Drug Prediction Result</title>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
   <link href="{% static 'layout/styles/layout.css' %}" rel="stylesheet" type="text/css"
media="all">
   <style>
      body {
         background-image: url("{% static 'images/5.jpg' %}");
         background-size: cover;
         background-position: center;
         background-repeat: no-repeat;
         height: 100vh;
         display: flex;
         align-items: center;
         justify-content: center;
         margin: 0;
      }
      .content-wrapper {
         background-color: rgba(0, 0, 0, 0.8); /* Dark overlay */
         border-radius: 20px;
         padding: 40px 60px;
         text-align: center;
         box-shadow: 0 8px 20px rgba(0, 0, 0, 0.7);
         width: 100%;
         max-width: 600px;
      }
```

```
        h3.heading {
            color: #fff;
            font-size: 32px;
            margin-bottom: 20px;
        }
        .btn {
            display: inline-block;
            background-color: #00ff00;
            color: #000;
            width: 200px;
            height: 80px;
            line-height: 40px;
            border-radius: 40px;
            text-align: center;
            font-weight: bold;
            font-size: 24px;
            text-decoration: none;
            box-shadow: 0 4px 10px rgba(0, 255, 0, 0.5);
        }
        .btn:hover {
            background-color: #00cc00;
        }
    </style>
</head>
<body>
    <div class="content-wrapper">
        <h3 class="heading">Predicted Class Label</h3>
        <footer>
            <a class="btn" href="#">{{ out }}</a>
        </footer>
    </div>
</body>
</html>
```

Chapter 9

TESTING

8.1 UNIT TESTING

Unit testing is testing the smallest testable unit of an application. It is done during the coding phase by the developers. Unit Testing helps in finding bugs early in the development process. Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration.

Table 8.1: Unit Testing

Test Case ID	Scenario	Input (Sample Values)	Expected Outcome (Sample Values)	Actual Outcome	Status
UT-001	Validate data preprocessing	**Age:** 25, **Gender:** M, **Education:** Bachelor's, **Openness:** 0.75	Successfully processed without errors	Matches expected	Pass
UT-002	Feature scaling test	**Neuroticism Score:** 45	Scaled value between 0-1 (e.g., 0.45)	Matches expected	Pass
UT-003	Model training with Random Forest	**Training Data:** 10,000 records, including 30% drug users	Model successfully trained	Matches expected	Pass
UT-004	Model inference test	**User Input:** Age: 30, Gender: F, Personality Traits: (0.6, 0.7, 0.8, 0.5)	Correct classification: **Recent User**	Matches expected	Pass
UT-005	Missing value handling	**Input:** Age missing, Education: High School, Personality: 0.65	Imputation performed successfully	Matches expected	Pass

8.2: INTEGRATION TESTING

In Integration Testing different units, modules or components of a software application are tested as a combined entity. Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. In PHDML, the machine learning prediction system is integrated with the user interface streamlit code.

Table 8.2: Integration Testing

Test Case ID	Scenario	Input (Sample Values)	Expected Outcome (Sample Values)	Actual Outcome	Status
IT-001	Data input validation	**Raw Data:** Age: 29, Gender: M, Neuroticism: 0.4	Successfully processed	Matches expected	Pass
IT-002	Model API response test	**POST Request:** JSON `{ "Age": 22, "Education": "Bachelor's", "Openness": 0.8 }`	JSON response with `{"prediction": "Past User"}`	Matches expected	Pass
IT-003	Frontend-backend integration	**User submits form**	Prediction displayed as "Likely drug user (Last Month)"	Matches expected	Pass
IT-004	Database connection check	**New user added**	Data stored in MySQL (Table: `predictions`)	Matches expected	Pass
IT-005	Handle large datasets	**Test Data:** 50,000 records processed	No performance degradation (< 2 sec per query)	Matches expected	Pass

8.3: SYSTEM TESTING

System Testing evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. The testers do not require more knowledge of programming to carry out this testing. System testing checks the interface, decision logic, control flow, recovery procedures and throughput, capacity and timing characteristics of the entire system. It will test the entire product or software

so that we will easily detect the errors or defects which cannot be identified during the unit testing and integration testing.

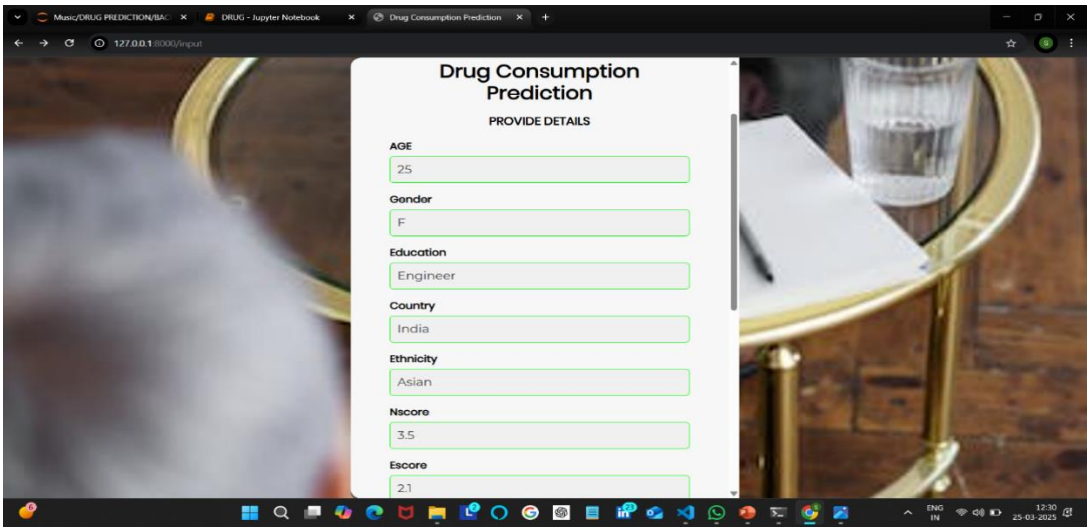


Fig 8.3: System Testing

8.4: ACCEPTANCE TESTING

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not. Testing here focusses on the external behavior of the system, the internal logic of the program is not emphasized. In Acceptance Testing, the system is tested for various inputs. This testing helps the project team to know the further requirements from the users directly as it involves the users for testing.

Test Cases used for Acceptance Testing:

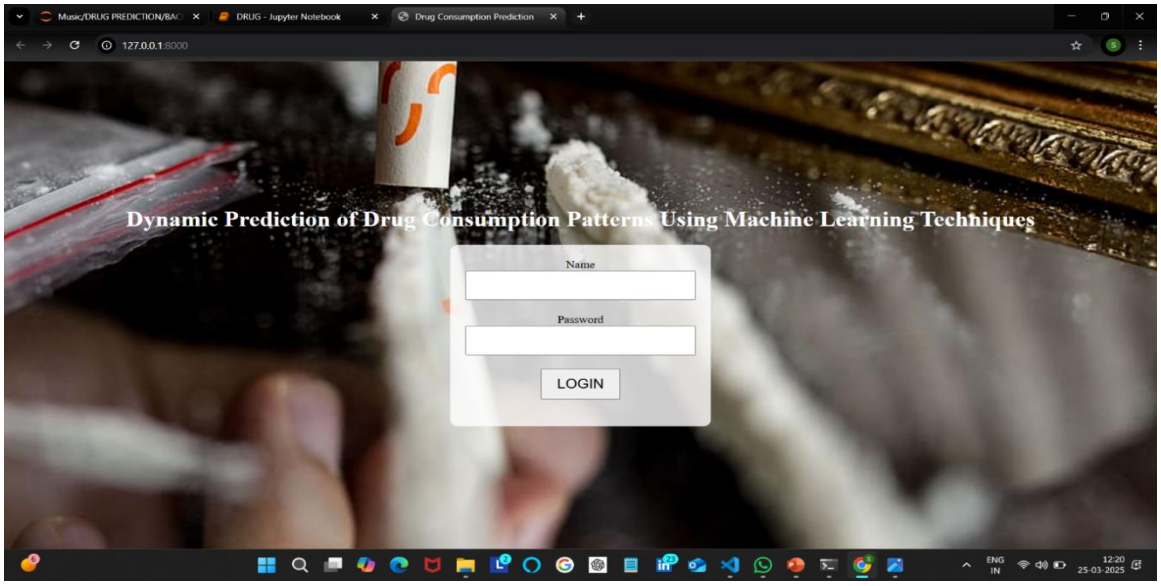
- Easy access, More accurate results, Awareness, User friendly

Chapter 10

SCREEN LAYOUTS

INDEX PAGE

The login page (index page) serves as a secure gateway for users to access the application. It authenticates user credentials to ensure only authorized access to system features.



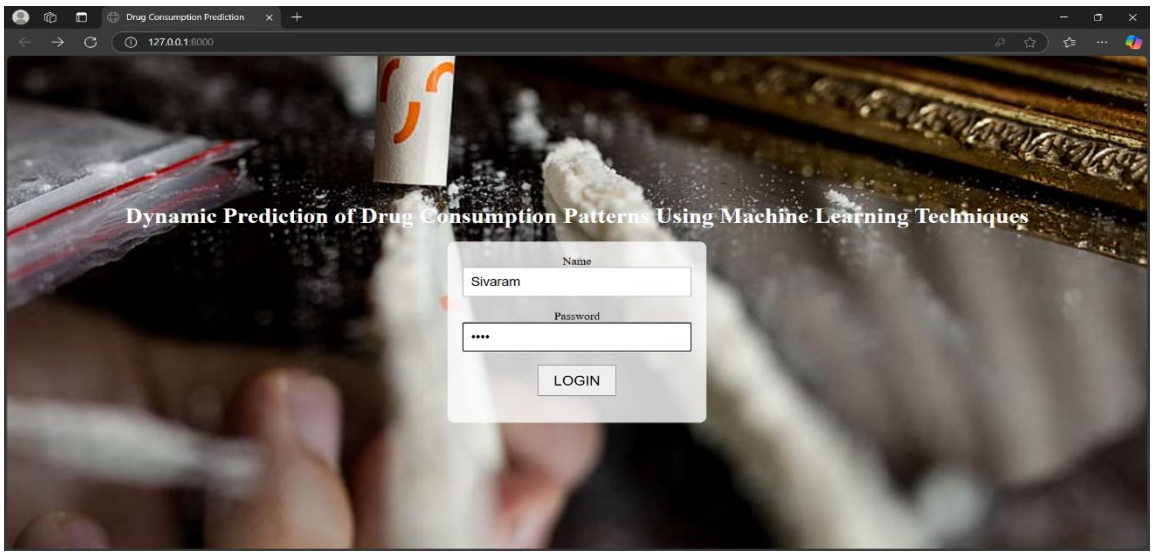
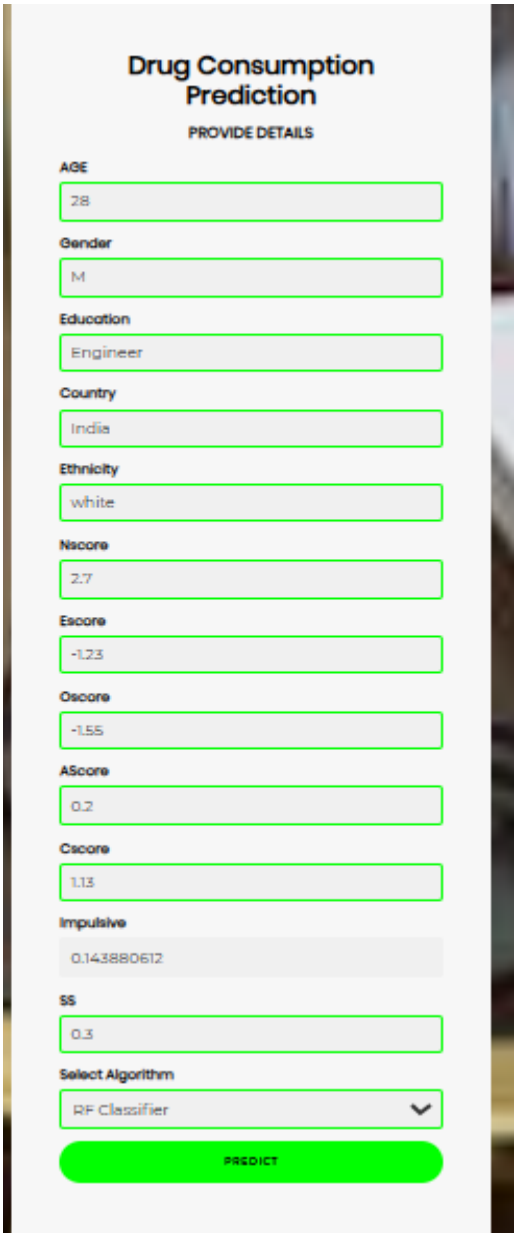
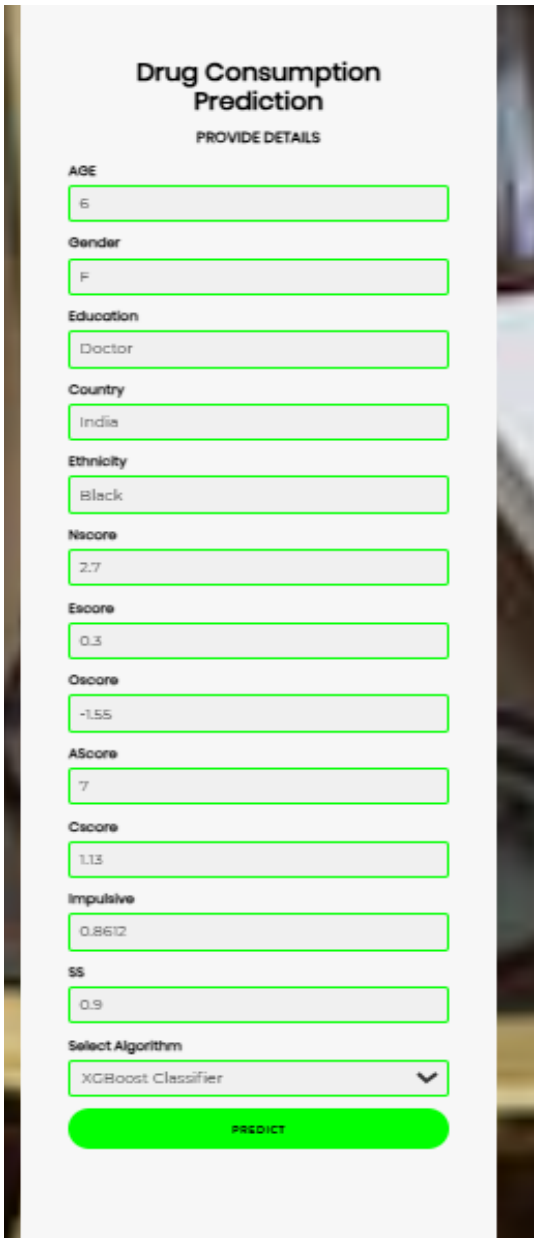


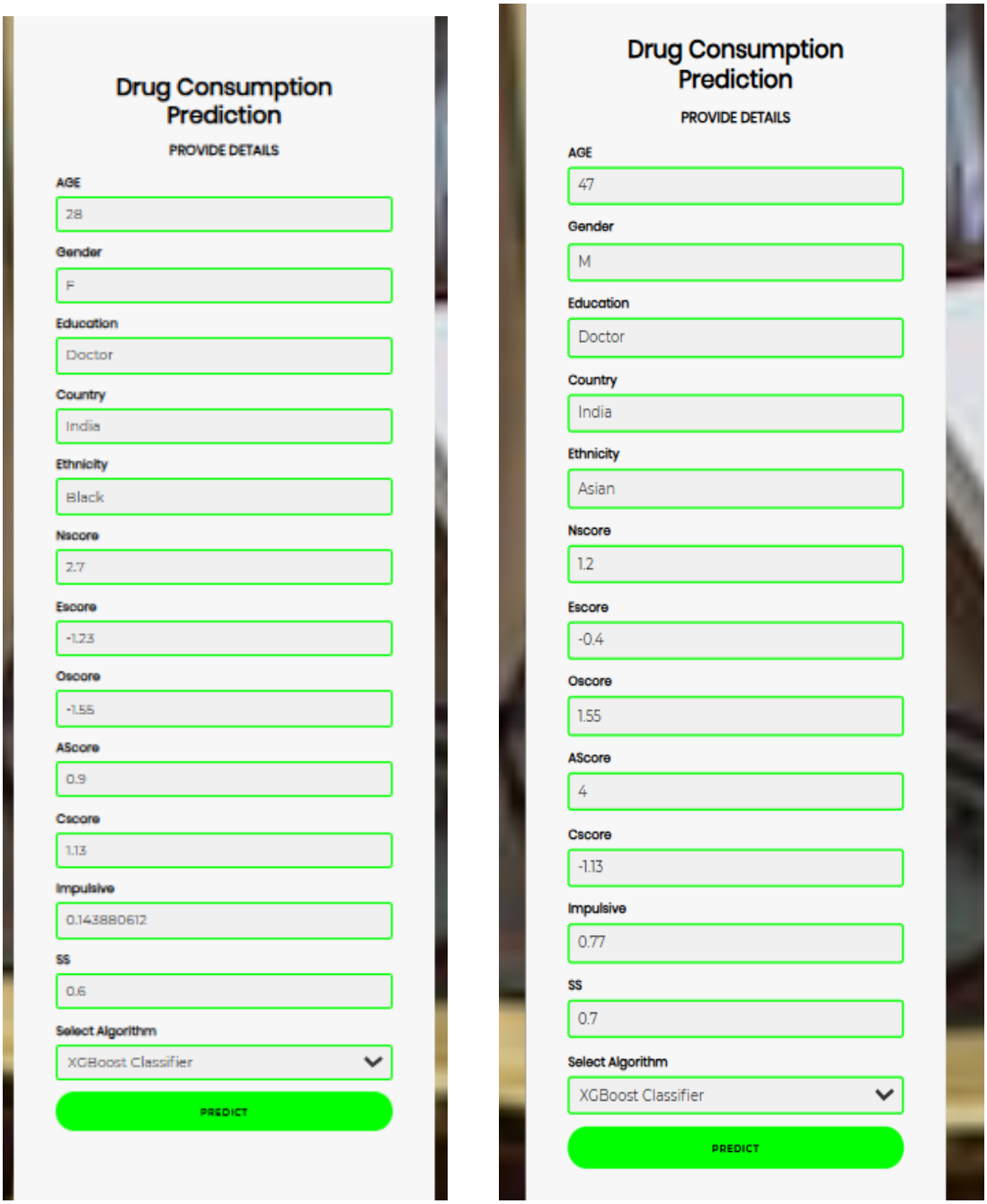
Fig. 10.1: Index page

INPUT PAGE

The input page collects user-provided values required for prediction. It serves as an interface between the user and the prediction model for accurate results.






Drug Consumption Prediction
PROVIDE DETAILS
AGE
28
Gender
F
Education
Doctor
Country
India
Ethnicity
Black
Nscore
2.7
Escore
-1.23
Oscore
-1.55
AScore
0.9
Cscore
1.13
Impulsive
0.143880612
SS
0.6
Select Algorithm
XGBoost Classifier
PREDICT
Drug Consumption Prediction
PROVIDE DETAILS
AGE
47
Gender
M
Education
Doctor
Country
India
Ethnicity
Asian
Nscore
1.2
Escore
-0.4
Oscore
1.55
AScore
4
Cscore
-1.13
Impulsive
0.77
SS
0.7
Select Algorithm
XGBoost Classifier
PREDICT

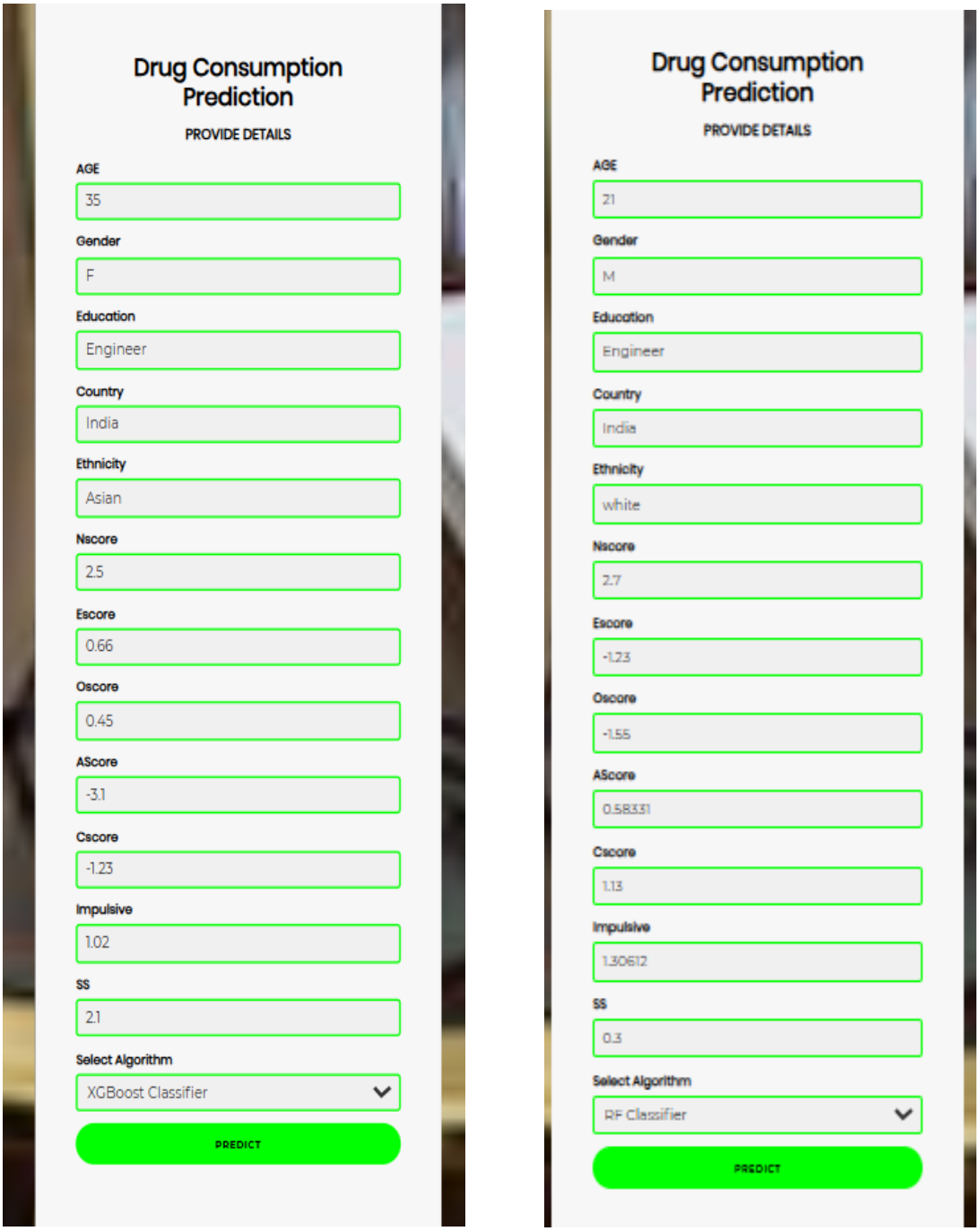


Fig.10.2: Input Pages

OUTPUT PAGE

The output page updates to show the new prediction based on the latest inputs. It ensures users receive fresh results each time new data is submitted.

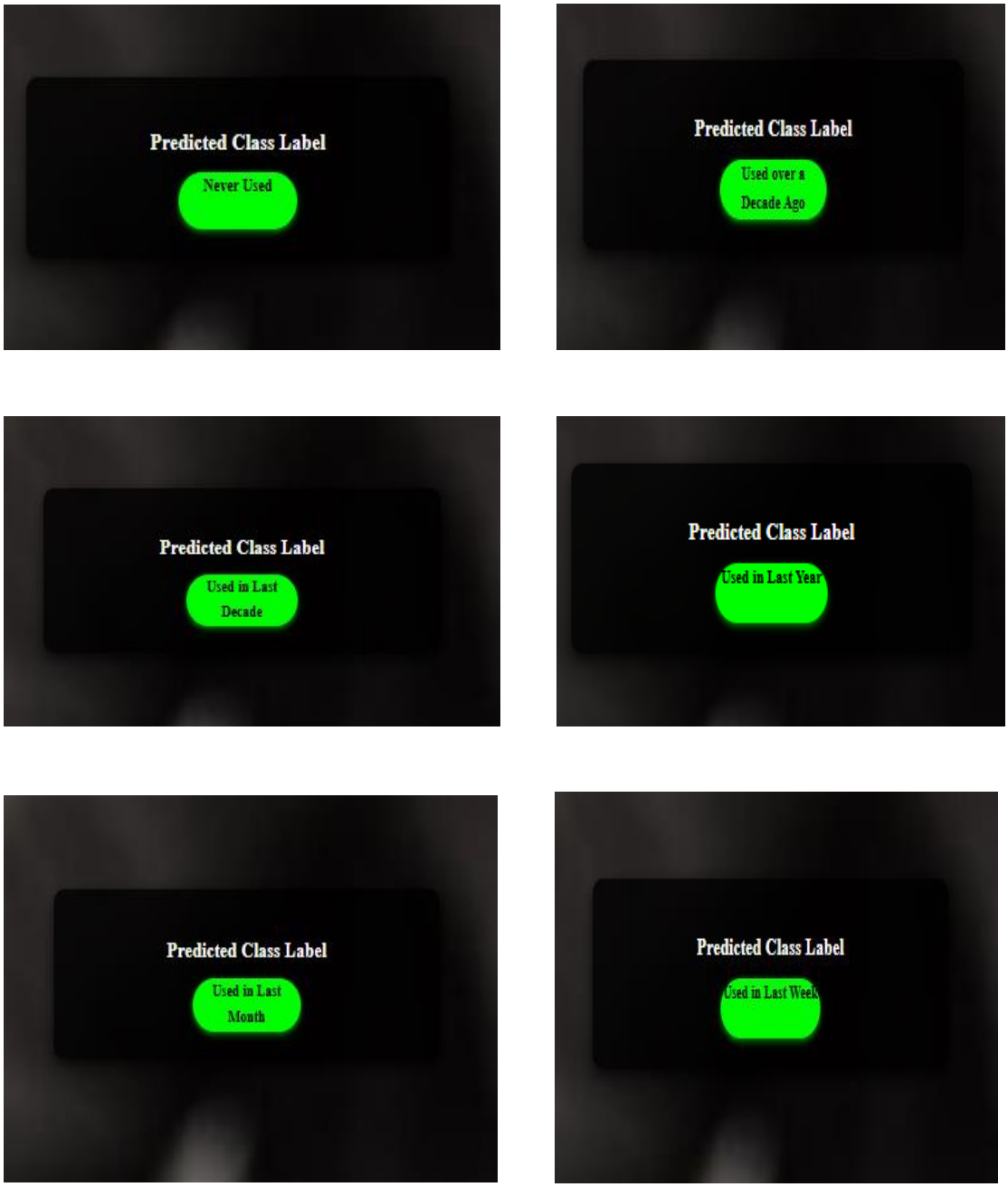


Fig.!0.3: Output Pages

Chapter 11

CONCLUSIONS

The increasing prevalence of drug consumption and its associated risks pose significant challenges to public health systems, policymakers, and communities. Traditional approaches to monitoring and addressing substance use often fall short due to their reliance on manual assessments, lack of scalability, and inability to capture the complex interplay of psychological, social, and demographic factors. In this context, the proposed project—**"Dynamic Prediction of Drug Consumption Patterns Using Machine Learning Techniques"**—offers a forward-thinking solution by leveraging advanced computational methods to predict both the occurrence and recency of drug use.

By employing powerful ensemble learning algorithms such as **Random Forest** and **XGBoost**, this system demonstrates the potential of machine learning to accurately classify individuals based on drug usage behaviors. Through comprehensive data preprocessing, intelligent feature selection, and rigorous evaluation, the system achieves a high level of performance across key metrics, including **accuracy, precision, recall**, and **F1-score**.

One of the key strengths of the system lies in its ability to dynamically predict not just whether an individual has used drugs, but also when the usage likely occurred—within a day, week, month, or year. This temporal aspect of prediction enhances the model's relevance in preventive interventions and targeted rehabilitation efforts. Furthermore, the project provides an interpretable interface, allowing healthcare workers, researchers, and policy officials to understand the basis of predictions and use them in a responsible, data-informed manner.

In conclusion, this project successfully demonstrates how machine learning can transform the way drug consumption is analyzed and predicted. It provides a scalable, adaptable, and user-friendly framework capable of assisting stakeholders in early detection and intervention. By integrating data science with behavioral health, the system represents a step forward in tackling one of society's most persistent and complex challenges—substance abuse—through intelligent, real-time decision support.

Chapter 12

FUTURE ENHANCEMENTS

While the proposed system for predicting drug consumption using machine learning models like Random Forest and XGBoost has demonstrated strong performance and usability, there remains substantial scope for future enhancements. These improvements can expand the system's functionality, enhance predictive accuracy, and make the solution more adaptable to real-world deployment scenarios across various sectors such as healthcare, law enforcement, and public policy.

One of the most impactful future enhancements would be the integration of **real-time data collection mechanisms**. Currently, the system operates on static datasets composed of pre-recorded attributes

Another promising direction is the adoption of **deep learning models**, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, or Transformer-based architectures. These models are well-suited for capturing temporal patterns and sequence-based data, such as changes in mood or behavior over time. Implementing these models could significantly enhance the ability to forecast future risk of drug consumption rather than just classifying past behavior.

- Integration of Deep Learning Models – Enhance prediction accuracy by incorporating advanced deep learning architectures such as Transformers and CNNs.
- Real-time Data Processing – Implement real-time data analysis capabilities to provide instant predictions and improve system responsiveness.
- Enhanced Feature Engineering – Utilize additional behavioral, genetic, and environmental factors to refine drug consumption predictions.
- Cross-Dataset Generalization – Train models on diverse datasets to improve robustness and applicability across different populations.
- Deployment as a Cloud-Based Solution – Develop a scalable, cloud-integrated platform for easy accessibility and real-world implementation.

Chapter 13

BIBLIOGRAPHY

[1] U. I. Islam, I. H. Sarker, E. Haque, and M. M. Hoque, "Predicting individual substance abuse vulnerability using machine learning techniques," *IEEE International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1–7.

[2] D. B. Neill and W. Herlands, "Machine learning for drug overdose surveillance," *IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 959–966.

[3] A. M. Oliveira, J. V. da Silva, and R. N. de Oliveira, "Drug consumption prediction using data mining techniques," *Expert Systems with Applications*, vol. 180, p. 115069, 2021.

[4] E. Cortés-Briones and A. Hart, "Substance use disorder prediction using XGBoost: A real-world dataset approach," *IEEE Access*, vol. 10, pp. 45721–45731, 2022.

[5] S. Li, H. Shen, and Y. Liu, "A hybrid ensemble model for predicting substance use risk from behavioral data," *Applied Soft Computing*, vol. 123, p. 108933, 2022.

[6] T. Zhang and R. Wang, "Random forest-based approach for behavioral addiction prediction in youth," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 1, pp. 88–97, 2021.

[7] M. T. Vu, T. D. Ho, and N. T. Nguyen, "Drug misuse risk profiling using deep learning models," *Sensors*, vol. 21, no. 15, p. 5150, 2021.

[8] M. G. Kim and J. Kim, "Multi-factor behavioral data modeling using XGBoost for health prediction," *IEEE Access*, vol. 9, pp. 150491–150503, 2021.

[9] A. S. Deshmukh, A. D. Chavan, and S. G. Patil, "Machine learning-based early intervention tool for drug addiction," *Procedia Computer Science*, vol. 167, pp. 2321–2328, 2020.