AN

INTERNSHIP REPORT

ON

DIABETES DISEASE PREDICTION USING ENSEMBLE MACHINE LEARNING

BY

M K N REVANTH KUMAR

HALL TICKET NO: 2129-24-862-081

MASTER OF COMPUTER APPLICATIONS

BY

OSMANIA UNIVERSITY, HYDERABAD-500007

AURORA'S POST GRADUATE COLLEGE (MCA),

NAMPALLY, HYDERABAD.

[2024-2026]

AURORA'S PG COLLEGE

(Affiliated To Osmania University, Hyderabad, Ts.)

NAMPALLY, HYDERABAD

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that **M K N REVANTH KUMAR**, bearing Hall Ticket No: **2129-24-862-081** isa Bonafide, student at the college in **Master of Computer Applications**. The following are the complete details of evaluation of the student's performance during the project internship, tools training and reviews during the project work.

This internship titled "**DIABETES DISEASE PREDICTION USING ENSEMBLE MACHINE LEARNING"** which is being submitted in partial fulfilment of the requirements for The award of MCA program of Department of **COMPUTER APPLICATIONS** of Osmania.

This has not been submitted to any other University or Institution for the award of any Degree Or Diploma.

Signature of the Guide:

Mrs. S. VIJAYA LAXMI.

Signature of HOD:

Mrs. S. VIJAYA SRI.

Signature of Principal:

DR. V. SREE JYOTHI.

VisionTech

ARPAD

INTERNSHIP COMPLETION CERTIFICATE

PROUDLY PRESENTED TO

M K N REVANTH KUMAR

In recognition of the successful completion of
AIML Internship at
VisionTech PVT LTD in partnered with Arpad Technologies from 23 Jun 2025 To 23 Aug 2025

Certificate ID: VTINT267
Issued Date: 30 Sep 2025

Scan to Verify

DIRECTOR

ACKNOWLEDGEMENT

The success and final output of the project required a lot of guidance and Assistance from many people, and I am extremely privileged to have got this all along the completion of my project. All that I have done, only due to such supervision and assistance, and I will not forget to thank them. I consider myself lucy enough to get such a good project. This project would Be added as an assert to my academic profile. I would like to express my thankfulness to my principal and **Head of the Department** for their constant motivation and valuable help through the project Work.

I owe my deep gratitude to my project guide **Mrs. S. VIJAYA LAXMI** who took, keep interest in my project work and guided me all along, till the completion of my project work by providing all the necessary information for developing a good system. I would like to thank my friends who helped me in improving my thesis.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staff of MCA which helped me in successfully completing our project work. Also, I would like to extend our sincere efforts to all staff in the laboratory for their timely support.
Finally, I wish to thank my parents for their love and encouragement, without whom I would never have enjoyed so many opportunities.

M K N REVANTH KUMAR
(2129-24-862-081)

DECLARATION

(ANNEXURE-I)

I hereby declare that this internship Report titled **"DIABETES DISEASE PREDICTION USING ENSEMBLE MACHINE LEARNING"** submitted by me To the Department of Computer Applications, OU, Hyderabad, is a Bonafide Work undertaken by me and it is not submitted to any other University or Institution for the award of any degree/diploma certificate or published any Time before.

Name:

M K N REVANTH KUMAR

Signature:

M K N REVANTH KUMAR

INDEX

LIST OF FIGURES

DIABETES DISEASE PREDICTION USING ENSEMBLE MACHINE LEARNING

ABSTRACT

This project is about predicting the diabetes of the patient using the data. Using the past records of the patents data using the ensemble machine learning techniques like random forest and decision tree models we will be predicting the diabetes of the patient bases on the features taken from the data. Ensemble techniques will take more than one algorithm in an algorithm to predict the output. Based on the independent features of the patient health condition the machine learning model will be predicting whether the patient has diabetes or not. Using frontend techniques, we will be sending the predicted data we will be creating the webpage, and we will be showing the statistics of the patient's diabetes predicted data.

CHAPTER - 1

INTRODUCTION

The accurate prediction of health conditions using historical data has become a cornerstone of modern medical informatics. This project focuses on predicting the presence of diabetes in patients by leveraging historical health records and advanced computational methods. By analysing specific features related to a patient’s health condition, the system aims to provide a reliable diagnostic classification.

To achieve high predictive accuracy, this project utilizes ensemble machine learning techniques. Unlike standard models, ensemble methods combine multiple algorithms to produce a single, robust output. Specifically, this study implements Random Forest and Decision Tree models to process independent health variables and determine whether a patient is diabetic.

In addition to the backend predictive engine, the project incorporates frontend techniques to bridge the gap between data science and user accessibility. A dedicated webpage has been developed to serve as the interface for the system. This platform not only delivers the final prediction but also presents the statistics of the predicted data, offering a comprehensive view of the patient's health status based on the machine learning analysis.

CHAPTER – 2
LITERATURE SURVEY

Title: A survey on diabetes risk prediction using machine learning approaches.

Author: Birjais et al**.**

Abstract:

Diabetes is one of the world's fastest-growing chronic diseases, according to the World Health Organization. This study focuses on early diagnosis by experimenting with the PIMA Indian Diabetes (PID) dataset. The research evaluates several machine learning classifiers, including Decision Trees and Random Forest, to predict whether a person is diabetic based on specific health attributes. The results demonstrate that while individual models perform well, ensemble-related approaches like Random Forest often yield higher stability and accuracy, making them suitable for clinical decision support systems.

Author: Ramesh Prasad Bhatta.

Title: Diabetes Prediction Using Random Forest and XGBoost Machine Learning Algorithm

Abstract:

Diabetes mellitus is a prevalent chronic disease where timely identification is crucial for effective management. This study investigates the application of Random Forest (RF) and XGBoost classifiers for predicting diabetes using the PIMA Indian Diabetes dataset. To enhance predictive capability, a soft voting ensemble integrating these models was developed. The research highlights the internal working of tree-based algorithms and concludes that ensemble methods significantly improve reliability in forecasting the onset of diabetes based on glucose levels, age, and BMI.

CHAPTER – 3

SYSTEM ANALYSIS

System analysis involves defining the problem and establishing how the project will address it through technical requirements. For this project, the analysis focuses on transforming medical data into a predictive tool for early diabetes detection.

EXISTING SYSTEM

The existing system primarily relies on manual diagnosis and traditional clinical method**s**. Patients must visit a healthcare facility to undergo physical tests, such as fasting blood Glucose, or oral glucose tolerance tests.

DISADVANTAGES

Manual Dependency: Diagnosis depends heavily on the physical presence of the patient and the availability of medical experts.

Time-Consuming: Laboratory results often take hours or days to be processed and communicated back to the patient.

Late Detection: Because symptoms can be subtle in the early stages, manual systems often fail to identify "prediabetic" conditions until the disease has progressed.

Costly: Frequent visits to diagnostic centers and specialized laboratory tests increase the financial burden on the patient.

PROPOSED SYSTEM

The proposed system introduces an automated, machine learning-driven approach to predict diabetes using historical data. By utilizing ensemble techniques, the system provides a more robust and accurate output compared to single-model approaches.

Ensemble Modeling: The system uses Random Forest and Decision Tree models. By combining multiple algorithms, it minimizes errors and provides a more reliable prediction of whether a patient is diabetic.

ADVANTAGES

Feature-Based Prediction: The model analyzes independent patient health features (such as BMI, glucose levels, and age) to determine the output.

Web-Based Interface: Unlike traditional methods, this system features a frontend webpage. Users can input data digitally and receive results instantly.

Data Visualization: The proposed system doesn't just give a "Yes/No" result; it displays statistics of the predicted data, allowing for a better understanding of the patient's health trends.

Efficiency: It reduces the need for constant clinical visits for preliminary risk assessment, saving both time and money for the patient and the healthcare provider.

CHAPTER – 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

MINIMUM (Required for Execution)		MY SYSTEM (Development)
System	Pentium IV 2.2 GHz	i5 Processor 5th Gen
Hard Disk	50GB	512 Gb
Ram	4 Gb	8 Gb

4.2 SOFTWARE REQUIREMENTS

Operating System	Windows 10/11
Development Software	Machine Learning.
Programming Language	Python/Java.
Database	MYSQL.
Frontend Development	HTML, CSS, and JavaScript.
Web Server	Flask.

CHAPTER – 5

SYSTEM STUDY

The System Study is a detailed examination of the current manual processes and how the new automated system will replace them.

Objective: To design a system that takes patient health features (like glucose, BMI, and age) and accurately predicts diabetic risk using ensemble learning.

Data Analysis: The study involves analyzing historical medical records to identify the independent features that most strongly correlate with diabetes.

Model Selection: We study why Random Forest and Decision Tree are chosen—specifically because they handle non-linear data well and provide "feature importance," showing which health metric influenced the prediction the most.

Workflow: The study defines the flow from user input on a webpage processing via the Machine Learning model, returning statistical results to the frontend.

5.1 Feasibility Study

The feasibility study determines if the project is "doable" across technical, economic, and social dimensions.

Feasibility is the study of three things -

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

5.2 Technical Feasibility

This evaluates whether the technology to build the project is available.

Algorithms: Random Forest and Decision Trees are well-documented; mature algorithms available in libraries like Scikit-Learn.

Frontend: Standard web technologies (HTML/CSS/JS) can easily send data to a Python-based backend (like Flask or Django) for prediction.

Hardware: The project does not require high-end supercomputers; a standard PC or a basic cloud server is sufficient to train and run these models.

Economic Feasibility

This evaluates the costs versus the benefits.

Development Costs: Since we are using open-source tools and historical datasets, the initial software cost is near zero.

Operational Benefits: For healthcare providers, this system reduces the time doctors spend on manual data assessment. For patients, it provides a low-cost "first-opinion" tool that can prevent expensive emergency treatments through early detection.

Social Feasibility

This evaluates how the system will be accepted by people and society.

User Acceptance: A webpage interface makes the technology accessible to non-technical users, allowing patients to check their status from home.

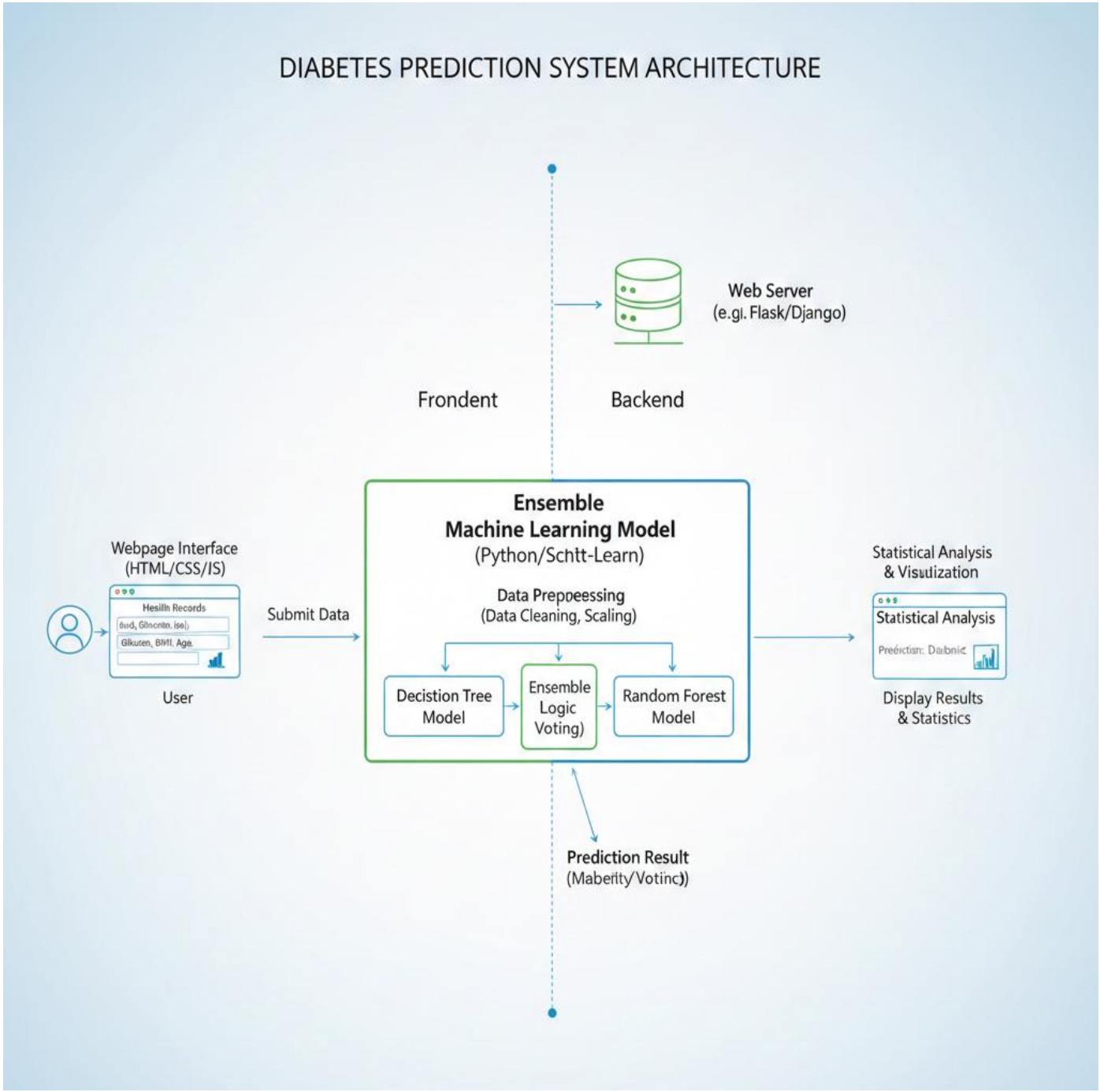
Health Impact: By providing statistics and predictions, it encourages "proactive" healthcare rather than "reactive" healthcare. It empowers users to take charge of their health before the condition becomes chronic.

Ethical Consideration: While the system provides a prediction, it is presented as a "decision-support tool" rather than a final medical diagnosis, ensuring it works alongside doctors rather than trying to replace them.

CHAPTER – 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE



6.2 UML DIAGRAMS

UML (Unified Modeling Language) is a standardized visual language used in software engineering to document and design the architecture of a system.

Think of it as a blueprint for a house. Instead of showing the actual bricks and wood (code), it shows the rooms, the wiring, and how people move through the house (system structure and logic).

A UML Class Diagram shows the relationship between different objects in your system. Since your project involves a frontend, a machine learning backend, and ensemble models, the structure is as follows:

GOALS:

Visualizing System Structure

The main goal is to provide a clear, graphical representation of the system. Instead of reading lines of code to find where the Random Forest algorithm is located, a UML diagram allows anyone to see the system components (Webpage, Backend, Ensemble Models) briefly.

Specifying Behavior and Logic

UML aims to define how the system behaves. In your project, the goal is to map out how the Ensemble Technique combines the Decision Tree and Random Forest outputs. It specifies the logic flow from "Inputting Health Features" to "Displaying Statistics."

Constructing the System

UML diagrams are not just drawings; they are blueprints used to construct the code. A goal of UML is to define the classes and methods so that the developer knows exactly what functions to write before they even open a code editor.

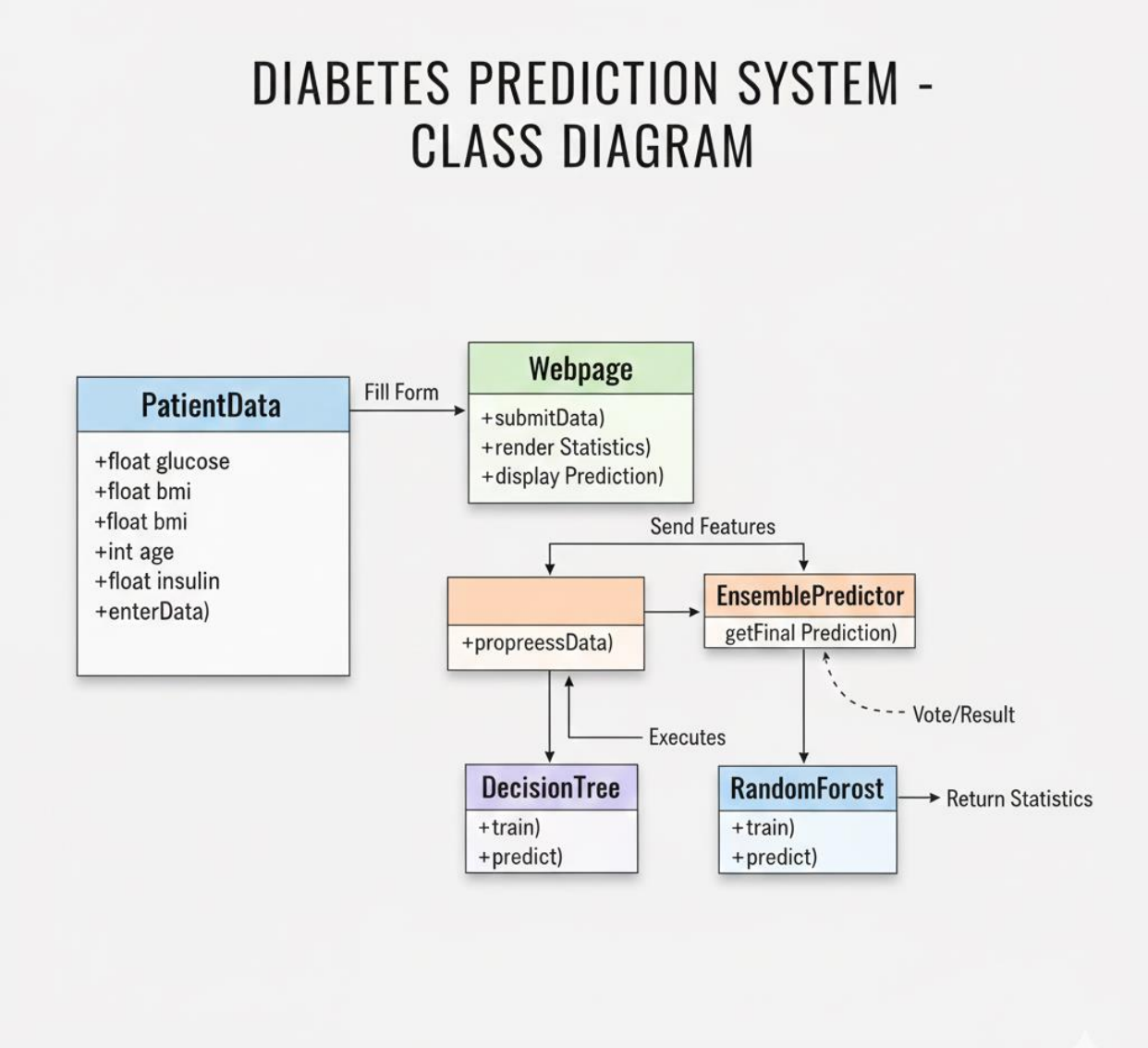
Documenting Decisions

UML serves as a permanent record of the design decisions made during the project. It documents:

Which features are being used for prediction.
How the frontend communicates with the machine learning models.
The relationship between different ensemble algorithms.

Simplifying Complexity

Machine Learning projects can be complicated. The goal of UML is to hide unnecessary details and focus on the essential parts of the system, making it easier to explain your project to others.



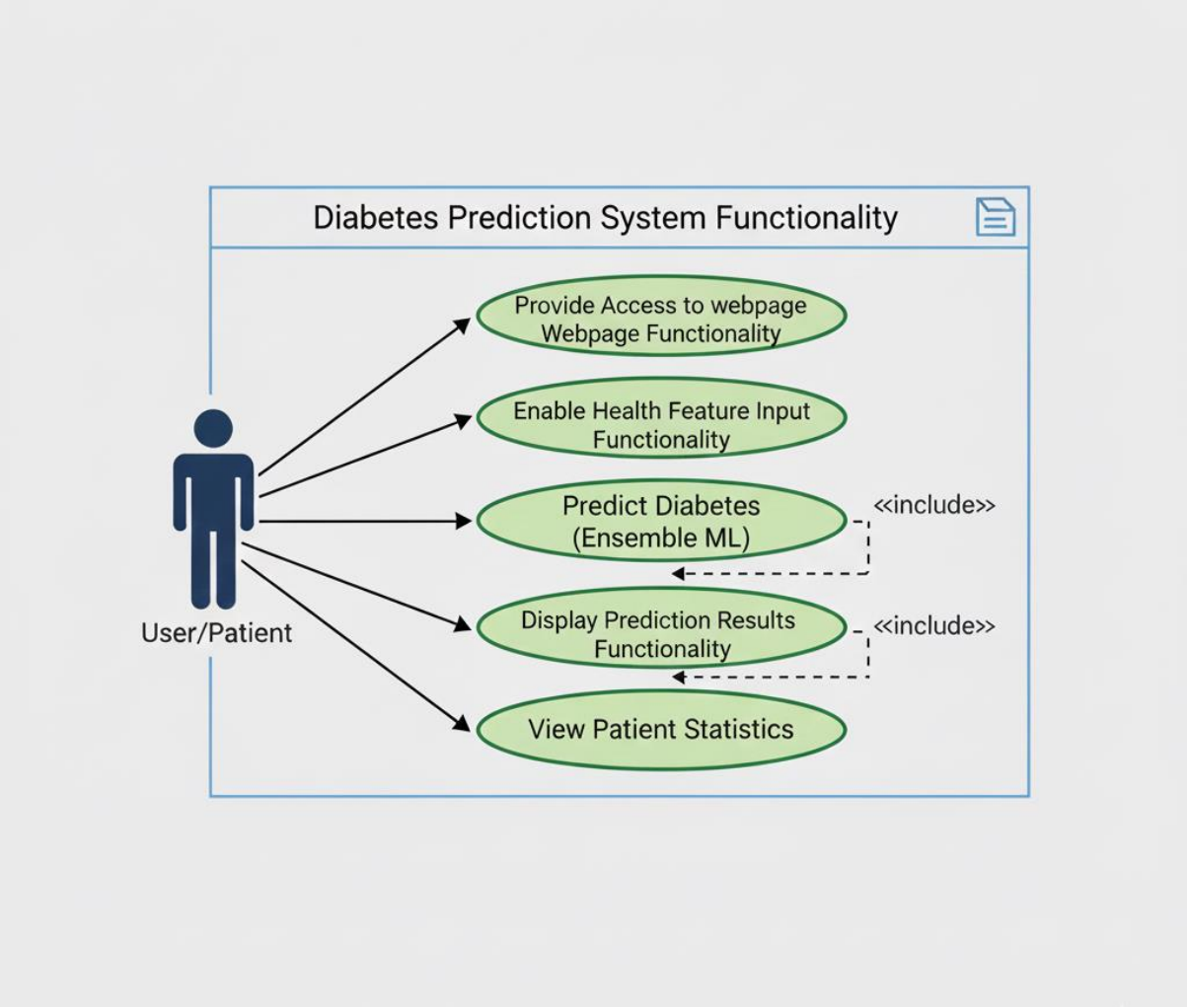
6.2.1 USE CASE DIAGRAM

For your Diabetes Prediction project, the Use Case Diagram illustrates the high-level interactions between the user and the system.

Actor: The User (Patient or Health Professional).

System Boundary: Diabetes Prediction Web Application.

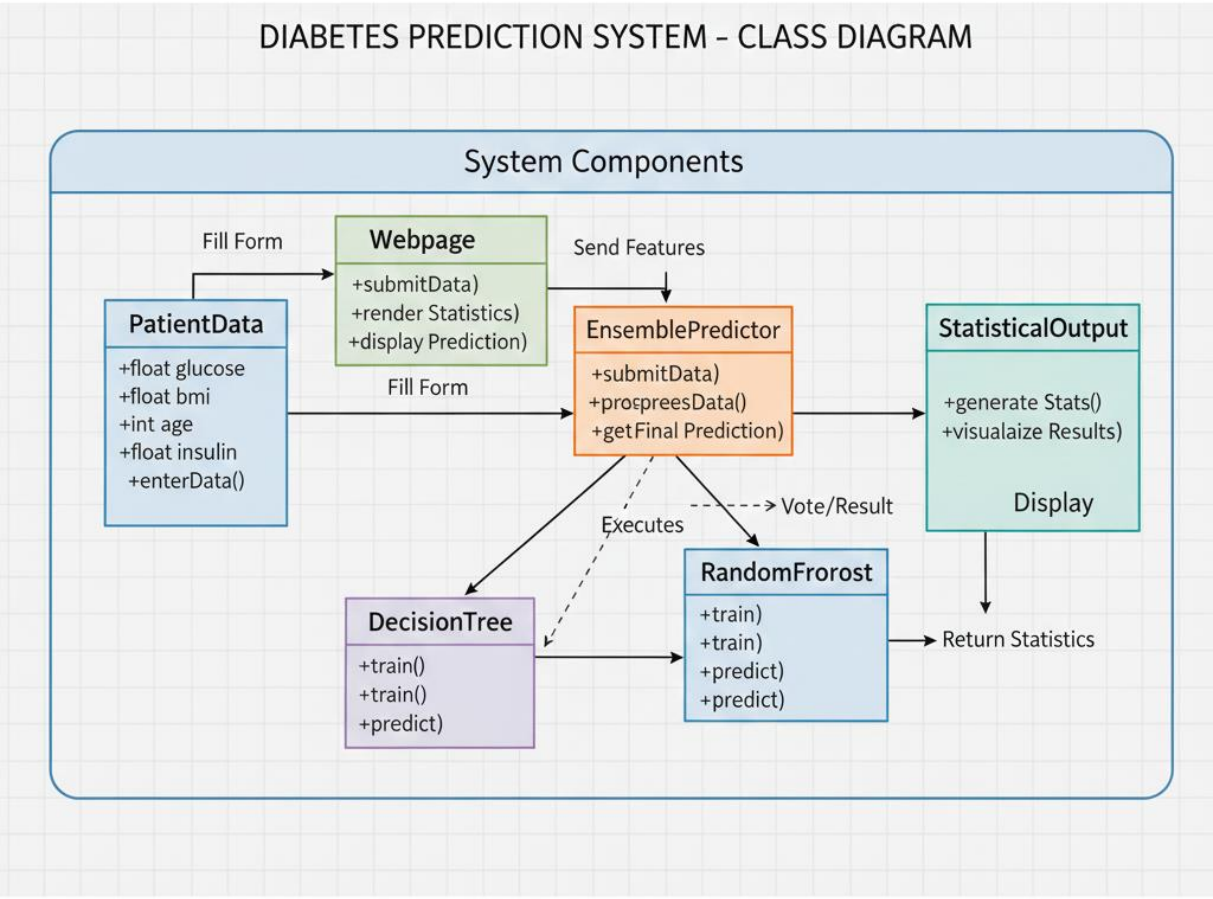
Use Cases: These represent the specific functions mentioned in your abstract, such as inputting data, running the ensemble models, and viewing statistics.



6.2.2 CLASS DIAGRAMS

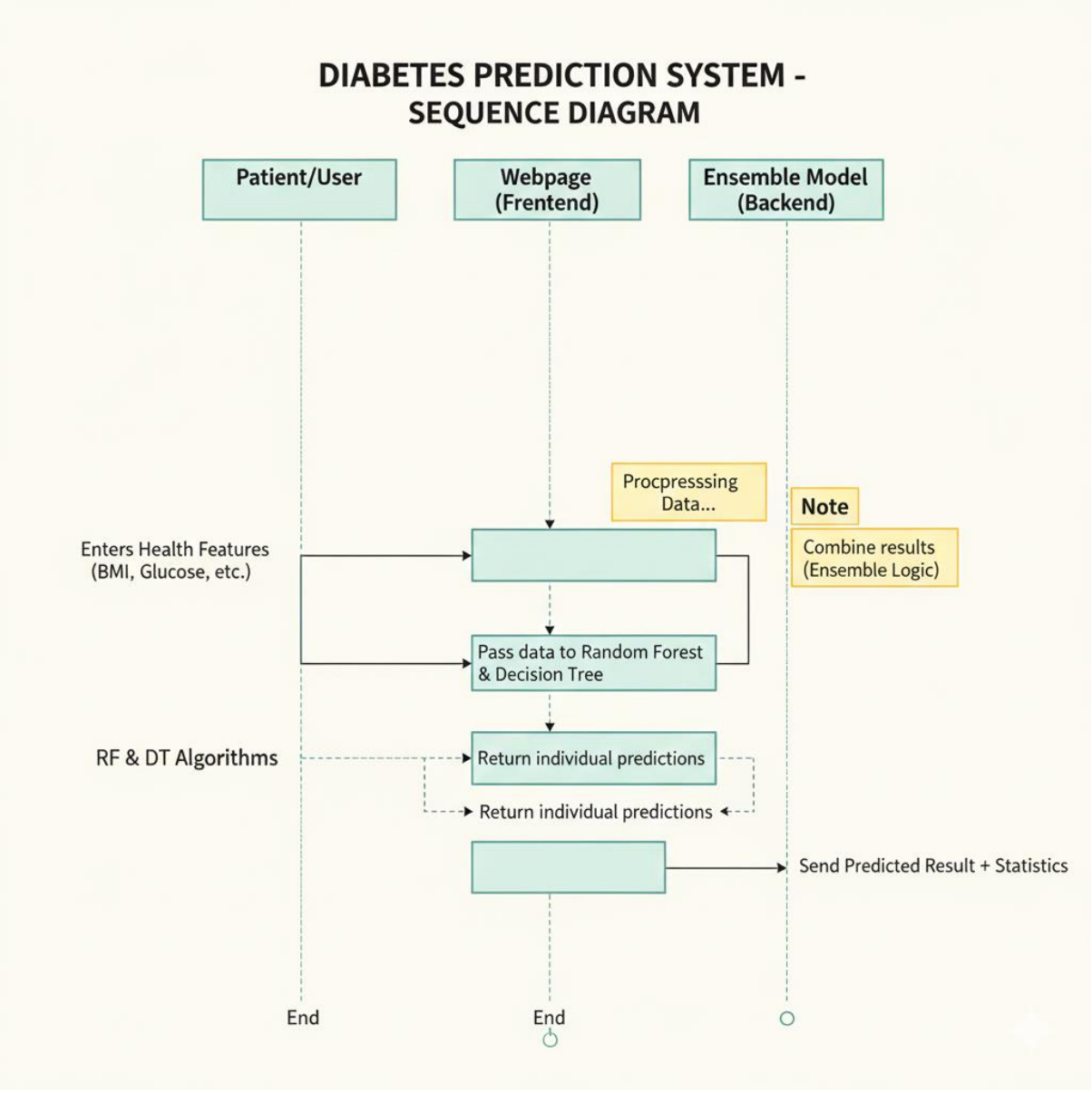
A Class Diagram is a type of static structure diagram in UML that describes the structure of a system by showing the system's classes, their attributes (data), operations (methods), and the relationships among objects.

In your project, the Class Diagram acts as the technical map for your code. It shows how the Patient Data class interacts with the Ensemble Model class, and how the Random Forest and Decision Tree algorithms are structured as part of that ensemble.



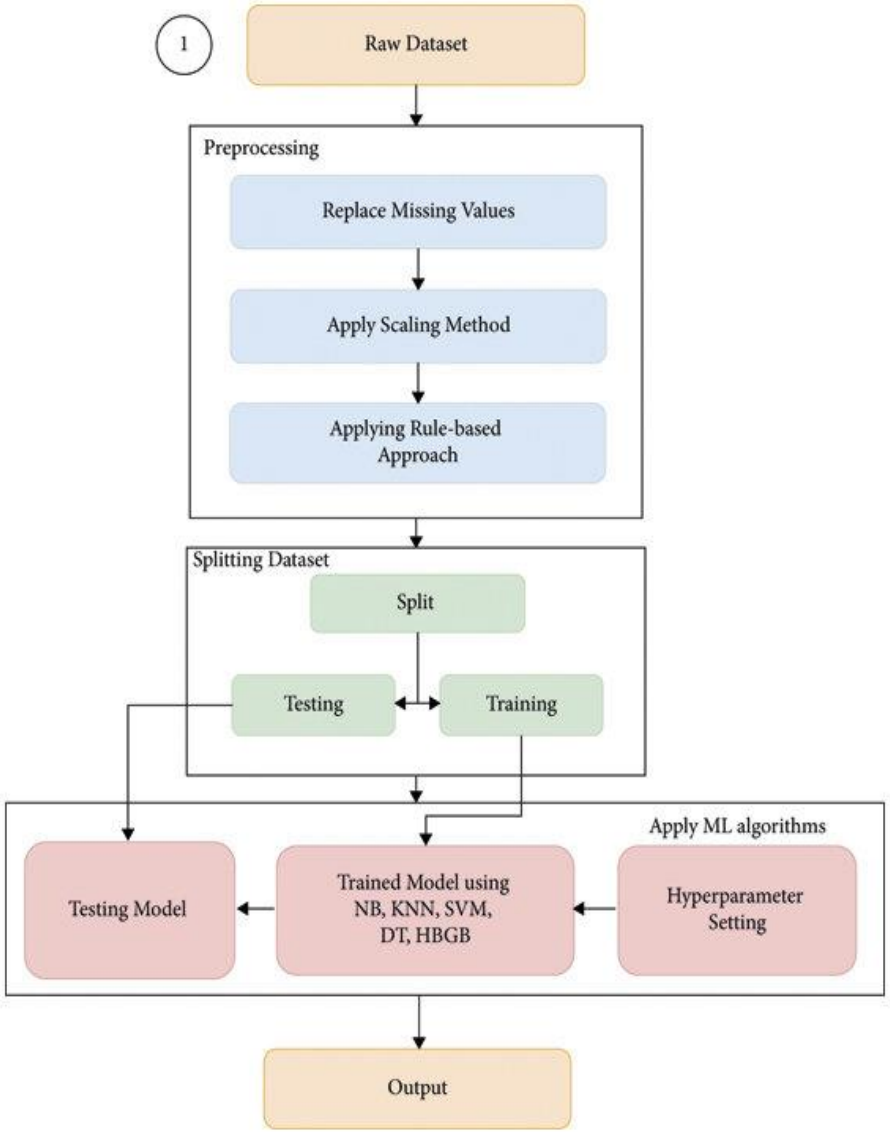
6.2.3 SEQUENCE DIAGRAM

A Sequence Diagram is a type of UML interaction diagram that shows how different objects or system components communicate with each other over time. It is essentially a timeline that reads from top to bottom, detailing exactly what messages are sent and in what order they occur to complete a specific task or use case.



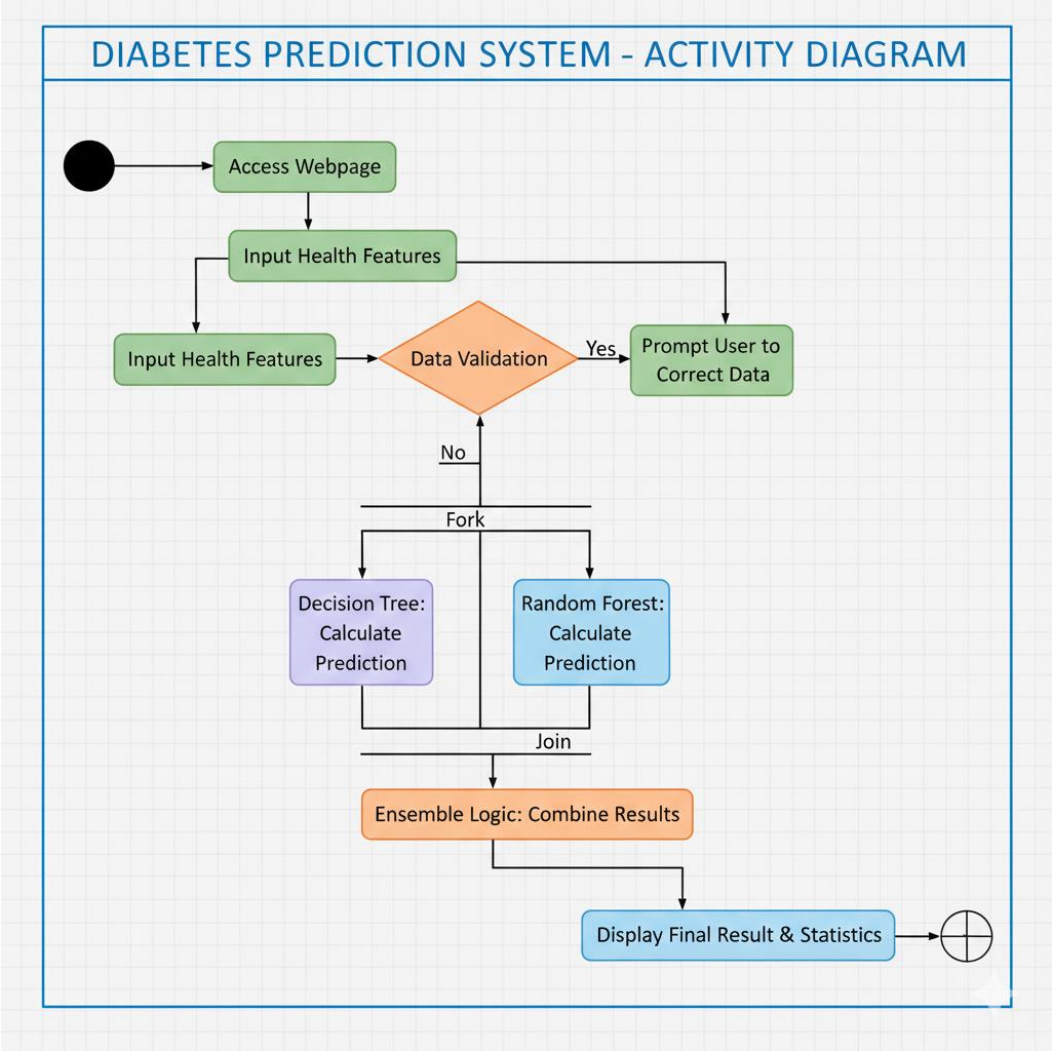
6.2.4 COLLABORATION DIAGRAM

A Collaboration Diagram is a type of interaction diagram that shows how objects or system components work together to perform a specific task.



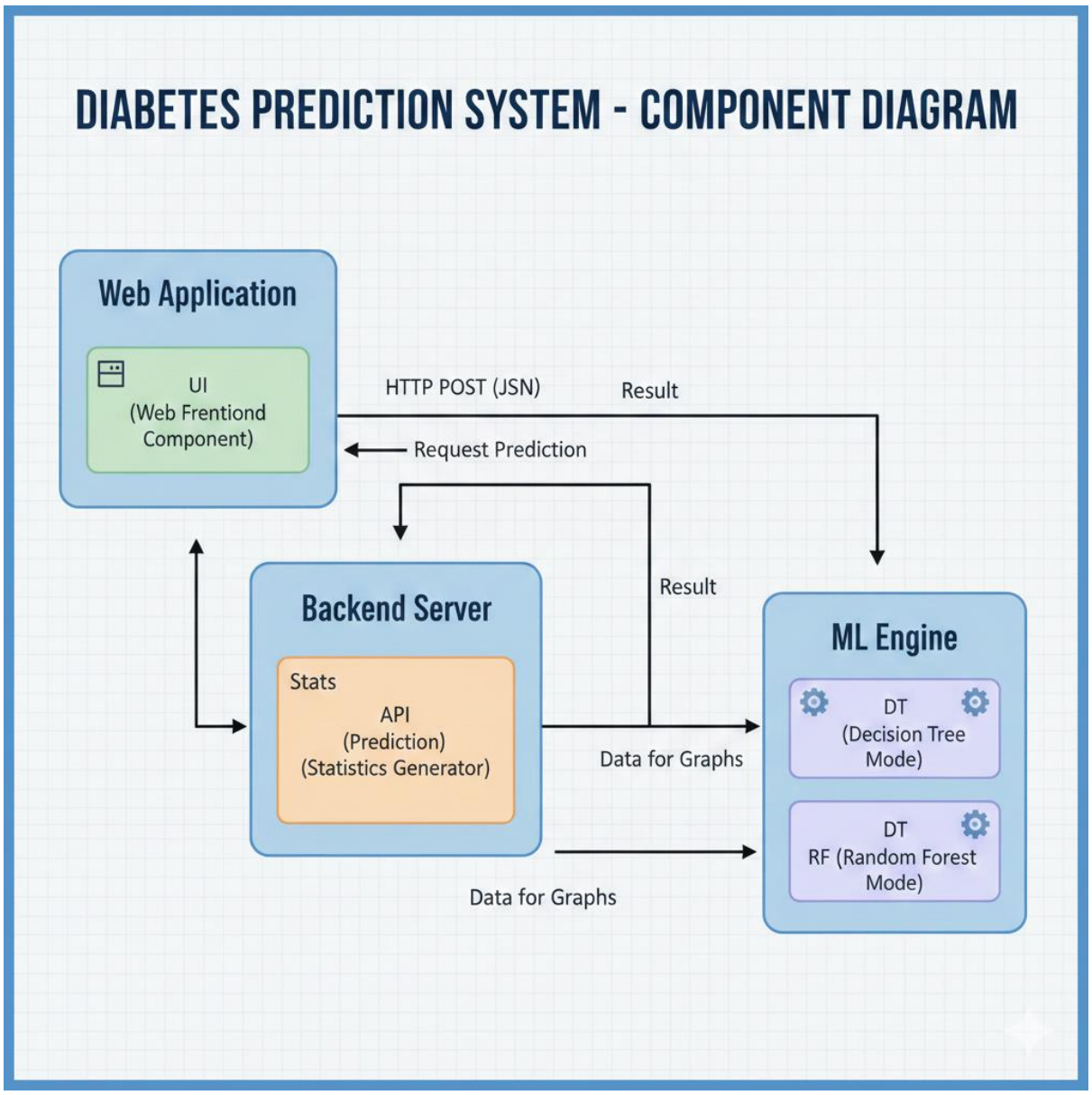
6.2.5 ACTIVITY DIAGRAM

An Activity Diagram is essentially a high-level flowchart that shows the dynamic behavior of a system. In your project, it describes the step-by-step workflow from the moment a user opens the webpage to the point where the final diabetes prediction and statistics are displayed.



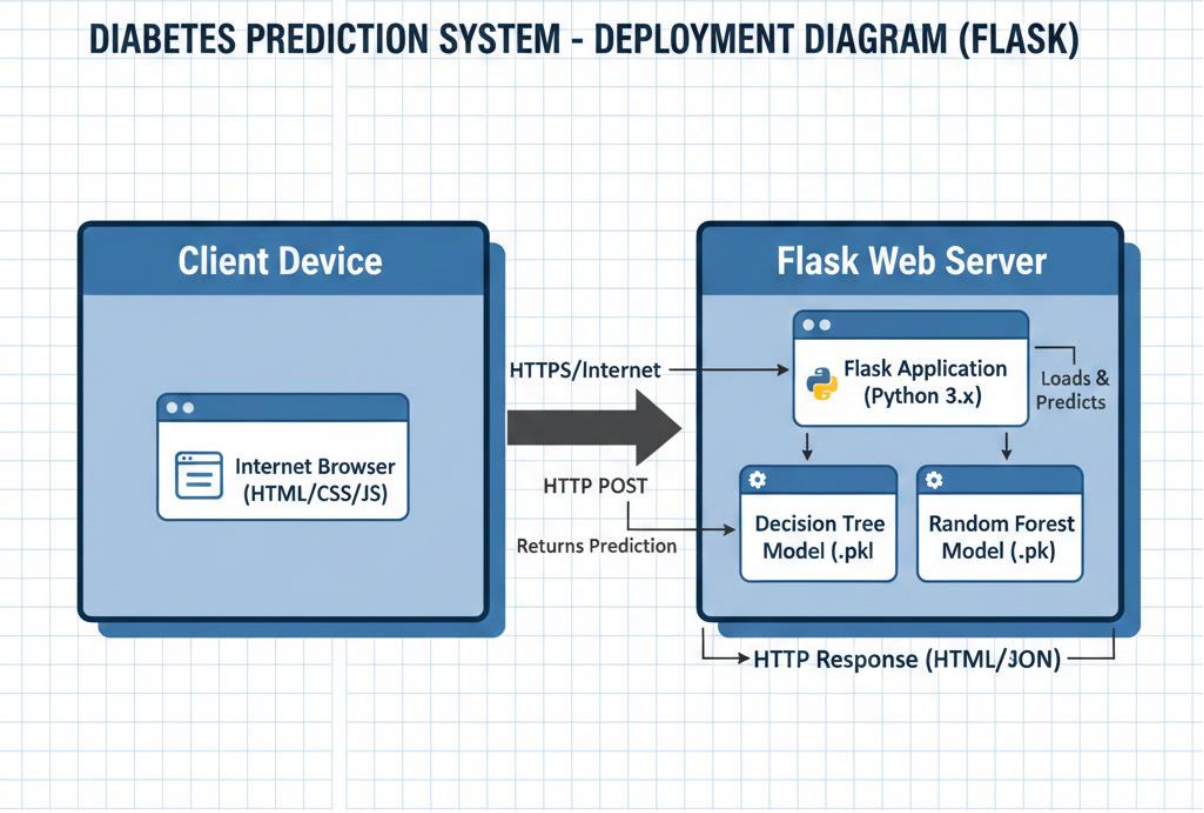
6.2.6 COMPONENT DIAGRAM

A Component Diagram is a structural UML diagram that describes how the system is divided into physical or logical components and the dependencies between them.
For your project, it shows the "modular" nature of your software—specifically how your Frontend, Backend API, and Machine Learning Models are separate pieces that work together.



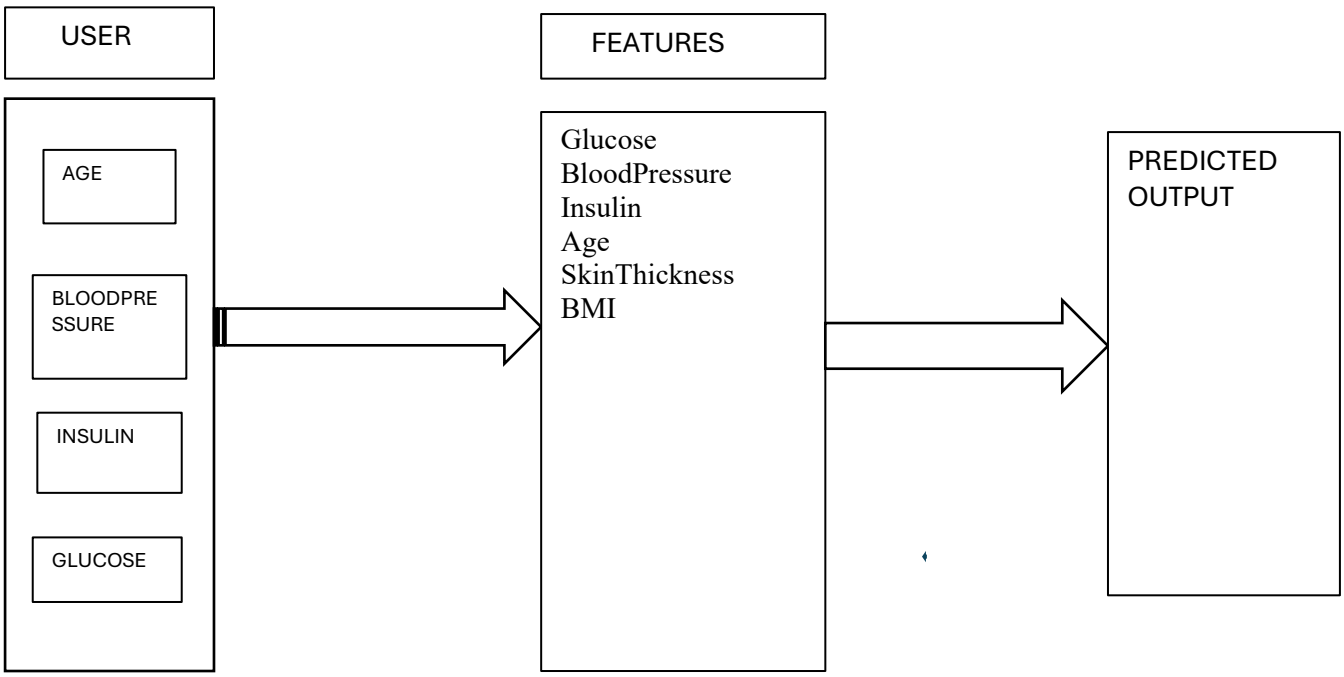
6.2.7 DEPLOYMENT DIAGRAM

It shows the physical configuration of the system. Since you are using Flask, your diagram highlights how the Client Browser communicates with the Flask Web Server, which in turn hosts your Python scripts and Trained ML Models.



6.2.8 ER DIAGRAM

An ER Diagram (Entity-Relationship Diagram) is a type of structural diagram used in database design. It acts as a blueprint that shows how different "entities" (like People, Objects, or Concepts) relate to one another within a system.



CHAPTER – 7

INPUT AND OUTPUT DESIGN

7.1 INPUT DESIGN

Input design for a project involves planning how users enter data, focusing on simplicity, accuracy, and efficiency by creating intuitive forms/screens with clear labels, consistent styles, logical grouping, and appropriate input types (text fields, dropdowns, checkboxes) to minimize errors and guide users, ensuring data quality for better system output. Key steps include identifying data needs, choosing input methods, designing user-friendly interfaces, and validating input to ensure it Is measurable and appropriate for the system's purpose.
Input Design focuses on the user-friendly collection of data via a web interface and its conversion into a machine-readable format for your Flask backend.

The primary objective of Input Design is to make data entry as easy, logical, and error-free as possible. Since your project uses specific medical features like Glucose, Blood Pressure, Insulin, and Age, the design ensures that what the user types matches what the Flask backend and ML model need.

7.1.1 OBJECTIVES

Accuracy & Data Quality: To ensure that the data entered is correct. By using specific numeric inputs, you prevent "garbage in, garbage out," which is vital for an accurate diabetes prediction.

Efficiency: To minimize the steps a user takes to provide data. This is achieved by using a simple HTML form with a clear "Predict" button.

Validation: To catch errors immediately. For example, the input design should prevent a user from entering a negative age or a blood pressure value that is physically impossible.

Consistency: To ensure the data format (e.g., integer or float) is consistent with the columns in your dataset ['Glucose', 'BloodPressure', 'Insulin', 'Age'].

User Friendliness: To provide a simple interface with labels and placeholders so that even a non-technical user can navigate the prediction tool without confusion.

7.2 OUTPUT DESIGN

Output Design is the process of determining how the information generated by your system—specifically the prediction result—will be formatted, organized, and presented to the end user. In your Diabetes Prediction project, it is the bridge between the machine learning model's numerical calculation and a human-readable medical report.

7.2.1 OBJECTIVES

Clarity: To provide a result that requires no technical knowledge to understand, such as converting a 0 or 1 into "Diabetic" or "Non-Diabetic".

Relevance: To ensure only the most important information, specifically prediction, is emphasized by the user.

Accuracy: To ensure the output correctly reflects the data processed by the Flask backend and the Ensemble model.

Timeliness: To deliver the result immediately after the user interacts with the system, such as clicking the "submit" button.

Actionability: To provide a result that is immediately understandable, often using visual cues like color-coding (Red for high risk, Green for low risk).

Data Verification: To display the user's original inputs (Age, Glucose, BP, Insulin) alongside the result so they can verify the data used for the prediction.

Confidence Transparency: To show probability or confidence scores, which is particularly useful when using Ensemble methods like Random Forest.

CHAPTER – 8

IMPLEMENTATION

8.1 MODULES

- MANAGEMENT
- USER

8.2 DESCRIPTION

1. DIABETES DISEASE PREDICTION MODULE

Description:

This project focuses on the development of a predictive system that identifies the likelihood of diabetes in patients by leveraging Ensemble Machine Learning techniques. By combining multiple algorithms, the system achieves higher accuracy and robustness compared to using a single model.

The system uses a Flask-based web application where users input key health metrics—specifically Age, Glucose, Blood Pressure, and Insulin. These inputs are processed by an ensemble of models, such as Random Forest and Decision Trees, to provide a reliable "Diabetic" or "Non-Diabetic" prediction.

Core Functionalities

Ensemble Modeling Power: Combines multiple algorithms to create a "strong learner" that is more accurate and less prone to errors than a single model.

Targeted Medical Inputs: Specifically designed to process the four most critical health markers: Age, Glucose, Blood Pressure, and Insulin.

Web-Based Interface: Built with the Flask framework, allowing users to access the diagnostic tool through any web browser without installing complex software.

Real-Time Processing: Delivers instant prediction results immediately after the user submits the HTML form.

Key Objectives

High Accuracy: Using ensemble methods to minimize the risk of false negatives, which is critical in medical diagnostics.

User-Centric Design: Providing a simple HTML interface for patients or doctors to get instant results without needing technical expertise.

Real-time Prediction: Ensuring the Flask backend processes inputs and delivers the "Output Design" in seconds

CHAPTER – 9

SOFTWARE ENVIRONMENT

9.1 WHAT IS PYTHON

Python is a high-level, interpreted programming language known for its clear syntax and readability. Created by Guido van Rossum and released in 1991, it has become one of the most popular languages in the world because it allows you to express complex concepts in fewer lines of code compared to languages like C++ or Java.

Key Features

Easy to Read: Python uses English-like keywords and relies on indentation to define code blocks, making it very "clean" to look at.

Interpreted: You don't need to compile your code before running it. The Python interpreter executes the code line-by-line, which is great for quick testing.

Dynamically Typed: You don't have to declare whether a variable is a number or text; Python figures it out automatically at runtime.

Extensive Libraries: It has a massive "standard library" and thousands of third-party packages (via PyPI) that provide pre-written code for almost any task.

Applications Of Python

Data Science: Analyzing large datasets, creating visualizations, and performing complex math.

AI & Machine Learning: Building intelligent systems and neural networks (using libraries like TensorFlow or PyTorch).

Web Development: Managing the "back-end" server-side) of websites using frameworks like Django or Flask.

Automation: Writing simple scripts to automate repetitive tasks, like renaming 1,000 files at once.

Software Testing: Automating the testing process for new software applications.

Advantages Of Python

Simplicity and Readability

Python was designed to be easy to read. It uses a clean, uncluttered syntax that mimics the English language.

Low Learning Curve: Beginners can pick up the basics in a few days compared to weeks for languages like C++.
Reduced Maintenance: Because the code is easy to read, it is much easier for teams to collaborate on and maintain long-term projects.

Vast Library Support

You rarely must start from scratch in Python. It follows a "batteries included" philosophy, meaning it comes with a massive standard library.

Pre-built modules: Whether you need to manipulate images, perform complex calculations, or scrape data from the web, there is likely already a library (like NumPy, Pandas, or Requests) available to do the heavy lifting for you.

High Productivity

Because Python is so concise, developers can achieve more with fewer lines of code. This leads to:
Faster Prototyping: You can turn an idea into a working script much faster than in other languages.
Speed to Market: Companies can develop and deploy applications in record time.

Portability and Extensibility

Write Once, Run Anywhere: Python is cross-platform. Code written on a Mac will run on Windows or Linux without needing modification.
Integration: Python can easily call components written in C or C++, allowing you to use Python for the logic while using "faster" languages for performance-heavy tasks.

Thriving Community and Ecosystem

Python is one of the largest and most active developer communities in the world.

Endless Resources: From Stack Overflow to YouTube, there are millions of tutorials and forums to help you troubleshoot.
Enterprise Support: Major corporations contribute to Python's growth, ensuring the language stays modern and secure.

Advantages Of Python Over Other Languages

Development Speed vs. Execution Speed

The Advantage: Python allows for **2–10 times faster development** than C++ or Java. In 2026, "time-to-market" is often more valuable than raw processing speed.
The Trade-off: Python is an **interpreted** language, meaning it runs slower than **compiled** languages (C++, Rust, Go).
The Solution: Most modern Python libraries (like NumPy or TensorFlow) are written in C++ under the hood. You get the easy Python syntax with the "hidden" speed of C++.

Readability & Syntax (Python vs. Java/C#)

Less "Boilerplate": To print a line in Java, you need a class, a main method, and specific syntax. In Python, you just use print().
Indentation vs. Brackets: Python uses whitespace (indentation) to structure code, whereas JavaScript, C++, and Java use curly braces {}. This forces Python code to be clean and organized by default, making it much easier for teams to read each other's work.

Ecosystem Dominance (The "AI Advantage")

Standard for AI/ML: In 2026, Python is the undisputed king of Generative AI and Machine Learning. While you *can* do AI in JavaScript or Java, almost all new research, tools (like LangChain), and pre-trained models are released for Python first.
Library Wealth: Python has over 500,000 packages. Whether you are doing satellite data analysis (NASA), high-frequency trading, or simple web scraping, there is already a "kit" ready for you.

Disadvantage Of Python

Execution Speed (Performance)

Python is an interpreted language, meaning the code is executed line-by-line at runtime rather than being compiled into machine code beforehand.

Slower than C++/Java: For CPU-intensive tasks like high-end 3D gaming or complex physics simulations, Python is significantly slower.

Multi-threading Issues: Python has something called the Global Interpreter Lock (GIL). This makes it difficult for a single Python process to run multiple threads truly simultaneously on multi-core processors.

2. Memory Consumption

Python is not memory efficient. It uses a lot of RAMS because it needs to support high-level data types and automatic garbage collection.

Object Overhead: Every variable in Python is an "object," which takes up much more space than a simple variable in a language like C.

Not for RAM-constrained systems: This makes it a poor choice for tiny embedded systems or microcontrollers where memory is very limited.

Weak in Mobile Development

While Python dominates the desktop, server, and AI worlds, it is almost non-existent in the mobile app market.

No Native Support: Neither Android nor iOS supports Python as a native language.

Heavy Apps: While frameworks like Kivy exist, the resulting apps are often bulky and don't feel "native" to the user compared to apps written in Swift or Kotlin.

Runtime Errors

Because Python is **dynamically typing** (you don't declare if a variable is a string or a number), many errors that would be caught during "compilation" in other languages only appear when the program is running.

Harder to Debug Large Apps: In a massive codebase, a simple typo in a variable name might not cause a crash until that specific line of code is triggered hours later.

History Of Python

Python's journey from a humble holiday project to the world's most popular programming language is a fascinating tale of simplicity and community-driven innovation. It all began in December 1989 at the Centrum Wiskunde & Informatica (CWI) in the Netherlands, where Dutch programmer Guido van Rossum sought a successor to the ABC language that could handle exceptions and interface with the Amoeba operating system. During his Christmas break, van Rossum started writing the interpreter as a personal "hobby" project to keep himself occupied. The name was inspired not by the reptile, but by his love for the BBC comedy series *Monty Python's Flying Circus*, reflecting a desire for the language to be fun and accessible. By February 1991, the first public code (version 0.9.0) was released, already featuring core concepts like classes with inheritance, functions, and exception handling. Python 1.0 arrived in 1994, introducing functional programming tools like lambda, map, and filter, and shortly after, Guido moved to the USA to continue his work at CNRI. The year 2000 marked a massive milestone with the release of Python 2.0, which brought modern features like list comprehensions and a cycle-detecting garbage collector, cementing its place in professional development. However, the language's most controversial moment came in 2008 with the release of Python 3.0, a "breaking" version designed to clean up fundamental design flaws by sacrificing backward compatibility. This led to a decade-long "civil war" as developers slowly migrated from version 2 to 3, a transition that finally concluded when Python 2 was officially retired on January 1, 2020. During this time, Python's ecosystem exploded as it became the primary language for the Data Science and Artificial Intelligence revolutions, fueled by libraries like NumPy, Pandas, and TensorFlow. Guido van Rossum, who served as the "Benevolent Dictator for Life" (BDFL) for nearly 30 years, stepped down in 2018, leaving the language's future to a democratic Steering Council. Today, in 2026, Python stands at the absolute pinnacle of the tech industry, serving as the foundational language for Generative AI and Large Language Models, used by everyone from middle-school students to rocket scientists at NASA. Its history proves that a focus on human readability and open community collaboration can outlast and outshine even the most complex technical competitors.

What is Java

Java is a robust, object-oriented programming language launched by Sun Microsystems in 1995 with the powerful promise of "Write Once, Run Anywhere" (WORA). Unlike languages that compile code for a specific operating system, Java compiles code into a universal "bytecode" that runs on any device equipped with a Java Virtual Machine (JVM), making it a cornerstone for everything from Android mobile apps to massive enterprise banking systems. In 2026, it remains a dominant force in the industry due to its strict security features, automatic memory management (garbage collection), and its ability to handle high-performance tasks like big data processing and cloud-native microservices. While its syntax is more "verbose" or wordy compared to Python, this structure provides a level of stability and predictability that top-tier companies like Google and Netflix rely on for their most critical infrastructure.

JDBC In Java

JDBC (Java Database Connectivity) is a standard Java API that acts as a bridge or "translator" between a Java application and a relational database (like MySQL, PostgreSQL, or Oracle). It allows you to write Java code to perform database tasks like connecting to a server, sending SQL queries, and retrieving data without needing to know the low-level details of how each specific database communicates.

How JDBC Works (The Workflow)

The JDBC architecture follows a layered approach where your application talks to the API, and the API uses a "Driver" to talk to the database.

The process typically involves these **5 key steps**:

Register the Driver: You first load the database-specific driver class (e.g., com.mysql.cj.jdbc.Driver). This tells Java which "translator" to use for your specific database.

Establish a Connection: You use the DriverManager to open a connection session using a **JDBC URL** (like jdbc:mysql://localhost:3306/db), username, and password. This creates a Connection object.

Create a Statement: Through the connection, you create a Statement or PreparedStatement object. This acts like a vehicle that carries your SQL command to the database.

Execute the Query: You send your SQL command (e.g., SELECT or INSERT) using methods like executeQuery() (to get data) or executeUpdate() (to change data).

Process Results & Close: If you requested data, it arrives in a ResultSet (a virtual table). Once finished, you must close the result set, statement, and connection to free up computer memory.

Summary of JDBC Drivers

There are 4 types of drivers but Type 4 (Thin Driver) is the most common today because it is written entirely in Java and communicates directly with the database protocol, making it the fastest and most portable option.

What is Machine Learning

Before we look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and

understand aspects of newly observed data. I'll leave the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is like the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Challenges in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert

resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machine Learning

Healthcare & Personalized Medicine

Machine learning has shifted healthcare from "one-size-fits-all" to high-precision care.

Medical Imaging: Deep learning models analyze X-rays and MRIs to detect diseases like cancer or eye disorders with higher accuracy than human specialists.

Early Detection: Wearable devices monitor vital signs (heart rate, blood sugar, sleep) to predict emergencies like heart attacks or strokes before they happen.

Drug Discovery: ML simulates how new drugs interact with human cells, shortening the time to develop life-saving medications from years to months.

Finance & Security

The financial sector uses ML to manage risk and move money at lightning speed.

Fraud Detection: Banks use "anomaly detection" models to analyze millions of transactions in real-time, instantly blocking cards if spending behavior doesn't match your unique profile.

Algorithmic Trading: Over 70% of stock market trades are now executed by ML models that analyze global news and price trends to make split-second decisions.

Credit Scoring: Instead of just looking at debt, ML evaluates alternative data to help people without traditional bank accounts get approved for loans.

Transportation & Smart Logistics

ML is the "brain" behind self-driving vehicles and global supply chains.

Autonomous Vehicles: Companies like Tesla and Waymo use computer vision to help cars "see" and make split-second decisions on the road.

Route Optimization: Apps like Google Maps use ML to predict traffic congestion 30 minutes into the future, saving millions of gallons of fuel annually.

Predictive Maintenance: Airlines and trucking companies use sensors to predict when an engine part will fail, allowing them to fix it *before* a breakdown occurs.

Daily Life & Entertainment

Most of your digital experiences are curated by "Recommendation Engines."

Content Curation: Netflix, Spotify, and YouTube use ML to learn about your tastes, ensuring that the "Up Next" video is something you want to watch.

Virtual Assistants: Siri and Alexa use Natural Language Processing (NLP) to understand human speech, accents, and context.

Hyper-Personalized Shopping: Retailers like Amazon use ML to predict what you'll need next, often managing inventory levels before you even place an order.

Cybersecurity

In 2026, cyber threats move too fast for humans. ML acts as the frontline of defense.

Threat Hunting: ML identifies new, unknown malware (Zero-day exploits) based on suspicious behavior rather than a database of old viruses.

Phishing Prevention: Modern email filters can detect 99% of phishing attempts by analyzing subtle patterns in the sender's language and IP history.

How to Start Learning Machine Learning

Arthur Samuel coined the term "Machine Learning" in 1959 and defined it as a "Field of study that gives computers the capability to learn without being explicitly programmed". And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Learning machine learning (ML) in 2026 is a journey from understanding the "why" (math) to the "how" (coding) and finally the "what" (building). Because the field is so vast, having a structured roadmap is essential to avoid burnout.

Here are the 6 essential steps to mastering machine learning as of 2026.

Step 1: Build the Mathematical Foundation

You don't need a PhD, but you must understand the "engine" under the hood. Most ML algorithms are just fancy math applied to data.

Linear Algebra: Learn about vectors, matrices, and eigenvectors (crucial for how machines "see" data).

Calculus: Focus on **Derivatives** and **Gradients** (used to "train" models by finding the lowest error).

Statistics & Probability: Understand mean, median, standard deviation, and **Bayesian logic** to handle uncertainty.

Step 2: Master Python & Core Libraries

Python remains the undisputed language for AI. You need to go beyond basic syntax and master the "Data Science Stack":

NumPy: For high-speed mathematical operations.

Pandas: For cleaning and "wrangling" messy data tables.

Matplotlib/Seaborn: For visualizing data trends (if you can't see the data, you can't model it).

Step 3: Learn "Classic" Machine Learning

Before jumping into deep learning (AI like ChatGPT), you must master the core algorithms that run 90% of business applications:

Supervised Learning: Linear Regression (predicting numbers) and Logistic Regression/Decision Trees (predicting categories).

Unsupervised Learning: K-Means Clustering (finding hidden patterns in data without labels).

Scikit-Learn: This is the most important library for this stage—it contains almost every classic algorithm ready to use.

Step 4: Dive into Deep Learning & Neural Networks

This is where the "AI" magic happens. Learn how layers of artificial neurons mimic the human brain to process images and text.

Neural Network Basics: Learn about weights, biases, and activation functions (like ReLU or Sigmoid).

Frameworks: Pick one and stick with it—PyTorch (favorite for researchers) or TensorFlow (favorite for industry).

Transformers: In 2026, you *must* understand the Transformer architecture, as it is the foundation of modern Large Language Models (LLMs).

Step 5: Master MLOps (Deployment)

In 2026, companies don't just want models; they want models that work in the real world.

Model Deployment: Learn how to turn a model into an API using FastAPI or Flask.

Cloud Platforms: Get familiar with AWS SageMaker, Google Vertex AI, or Azure ML.

Monitoring: Learn how to track if your model's accuracy is "drifting" over time as new data comes in.

Step 6: Build a "Problem-Solving" Portfolio

Certificates are nice, but projects get you hired. Don't just copy tutorials; solve a unique problem.

Kaggle: Join competitions to practice on real-world datasets.

GitHub: Document your code and write "README" files explaining *why* you chose a specific model.

Disadvantages Of Machine Learning

Data Dependency and "Garbage In, Garbage Out"

The quality of an ML model depends entirely on the data used to train it.

Data Quality: If the training data is noisy, incomplete, or incorrect, the model will produce inaccurate results.

Data Quantity: Many advanced models (like Deep Learning) require millions of data points to function correctly, which is expensive and time-consuming to collect.

Algorithmic Bias and Fairness

ML models learn from historical data, which often contains human prejudices.

Social Bias: If a hiring tool is trained on data from a company that historically only hired men, the AI will learn to discriminate against women.

Echo Chambers: Recommendation engines can create "filter bubbles," showing users only what they already believe, which can radicalize opinions.

The "Black Box" Problem (Lack of Interpretability)

Complex models, especially Deep Neural Networks, are often "Black Boxes."

Non-Transparency: We can see the input and the output, but it is often impossible to explain *exactly why* the model made a specific decision.

High Risk: In fields like medicine or law, a "just trust me" answer from a machine is often legally and ethically unacceptable.

High Computational and Environmental Costs

Building and maintaining cutting-edge ML models requires massive resources.

Hardware Costs: Training a Large Language Model (LLM) requires thousands of high-end GPUs (like NVIDIA H100s), costing millions of dollars.

Energy Consumption: The electricity required to cool data centers and power model training has a significant carbon footprint.

Security Vulnerabilities

Machine learning introduces new types of "cyber" attacks that traditional software doesn't face.

Adversarial Attacks: Hackers can make tiny, invisible changes to an image (like a stop sign) that cause an autonomous car to "see" a speed limit sign instead.

Data Poisoning: If an attacker can slip "bad data" into the training set, they can create a "backdoor" into the AI system.

Python Development Steps

Planning and Environment Setup

Before writing code, modern developers ensure their workspace is isolated to avoid "dependency hell" (where different projects require conflicting versions of the same library).

Define Goals: Identify the problem, user requirements, and core features.

Install Python: Ensure you have the latest stable version (currently Python 3.13+).

Virtual Environments: Use tools like venv or pyenv to create a dedicated space for your project's libraries.

Version Control: Initialize a Git repository (git init) to track changes and collaborate on platforms like GitHub.

Design and Tech Stack Selection

Python's versatility means you must choose the right "tools for the job" based on your project type:

Web Development: Choose a framework like FastAPI (for speed/APIs) or Django (for large, feature-rich sites).

Data Science/AI: Select libraries like Pandas, NumPy, or PyTorch.

Database: Decide on a storage system (e.g., PostgreSQL for relational data or **MongoDB** for flexible documents).

Iterative Development (Coding)

This is where the actual logic is written. Python's interpreted nature allows for rapid "write-and-run" cycles.

Modular Coding: Break your project into smaller, reusable functions and classes.

API Integration: In 2026, many apps use "AI wrappers" to connect to LLMs via APIs (like OpenAI or Anthropic).

Type Hinting: Use optional type hints (e.g., name: str) to make your code more readable and catch bugs early.

Testing and Debugging

Python makes debugging easier by stopping execution the moment an error is found.

Unit Testing: Use frameworks like pytest to test individual pieces of logic automatically.

Debugging: Use the built-in debugger in VS Code or PyCharm to step through code line-by-line.

Linting: Use tools like Ruff or Flake8 to automatically check for style errors and "un-Pythonic" code.

Deployment and Maintenance

Once the code works, it needs to move from your computer to a server.

Containerization: Use **Docker** to package your app so it runs the same way on every computer.

CI/CD: Set up "Continuous Integration", so your code is automatically tested every time you push it to GitHub.

Hosting: Deploy to cloud platforms like AWS, Heroku, or Railway.

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something

Main Areas of Use

Artificial Intelligence & Machine Learning: It is the industry standard for building neural networks and AI agents.

Data Analysis: Python has replaced tools like Excel for processing massive datasets (millions of rows) efficiently.

Web Development (Back-end): Powering the logic behind sites like Instagram, Pinterest, and Spotify using frameworks like Django.

Automation & Scripting: Writing small programs to automate repetitive tasks like renaming files, scraping website data, or sending bulk emails.

Software Testing: Used heavily for automated bug testing in large software suites.

Modules Used in Project

Pandas (The Data Wrangler)

Pandas is the most important library for Data Manipulation. It introduces a structure called a DataFrame, which is essentially a super-powered Excel spreadsheet inside your code.

Purpose: Loading data from CSV/Excel files, cleaning missing values, and filtering/sorting data.

Key Feature: It allows you to perform complex operations (like calculating the average salary per department) in a single line of code.

Matplotlib (The Foundation of Visuals)

Matplotlib is the "grandfather" of Python plotting libraries. It provides low-level control over every element of a graph.

Purpose: Creating basic 2D graphs like line charts, bar charts, and histograms.

Key Feature: Extreme customization. You can control the exact color, font, and thickness of every line and axis in your graph.

Seaborn (The Statistical Artist)

Seaborn is built on top of Matplotlib. It makes it much easier to create beautiful, complex statistical visualizations.

Purpose: Making data look professional and uncovering patterns (like correlations) between variables.

Key Feature: High-level "themes" and complex plots like Heatmaps and Violin plots that would take 20 lines of code in Matplotlib but only 1 line in Seaborn.

Scikit-Learn / sklearn (The ML Brain)

While the others focus on data and charts, sklearn is for Machine Learning. It contains all the classic algorithms used to make predictions.

Purpose: Building models for classification (e.g., "Is this email spam?"), regression (e.g., "What will the house price be?"), and clustering.

Key Feature: It provides a consistent "fit" and "predict" workflow, making it very easy to switch between different types of AI models.

Installation Of Python Windows and MAC

Installation on Windows (10 & 11)

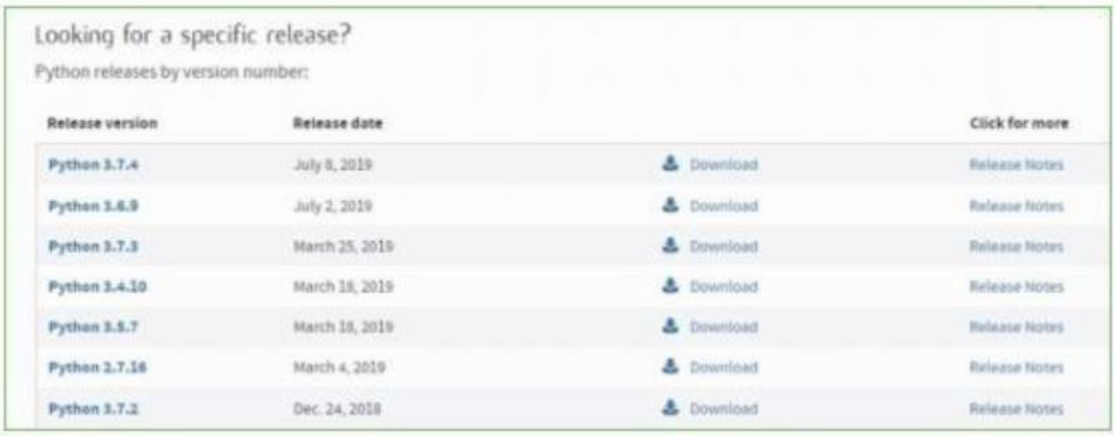
The most common way is using the official installer, but the Microsoft Store is a great alternative for beginners.

Method A: Official Installer (Recommended)

Download: Go to python.org/downloads and click the **"Download Python 3.x.x"** button.

Run Installer: Double-click the downloaded .exe file.

CRITICAL STEP: Before clicking "Install Now," check the box that says **"Add python.exe to PATH"**. If you miss this, you won't be able to run Python from the command prompt easily.



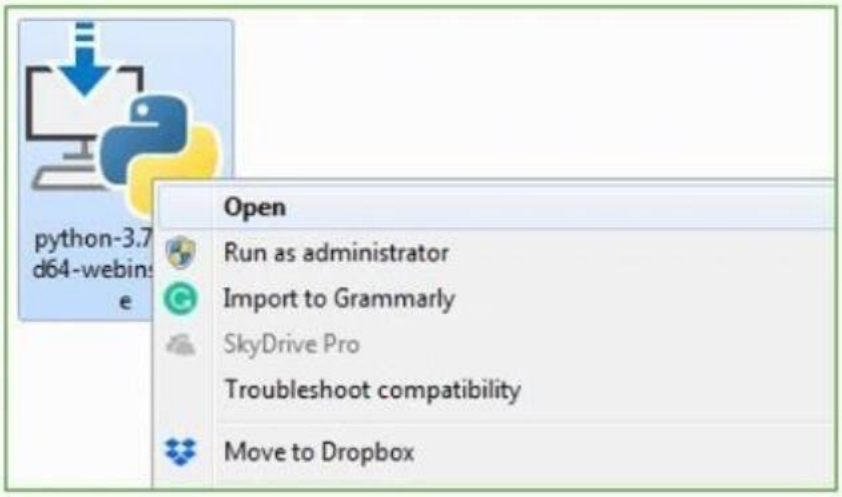
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

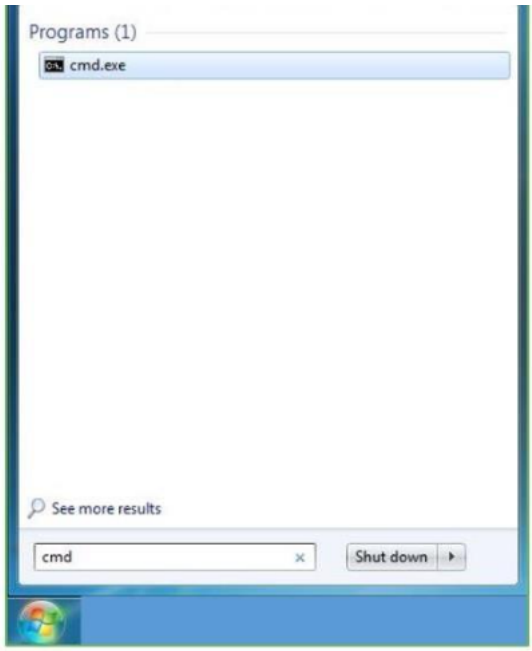
Files

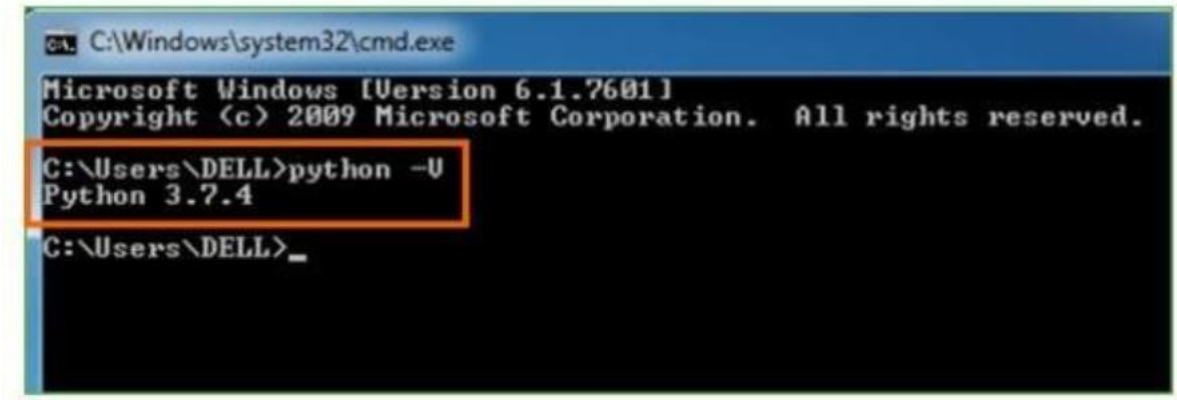
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e52d9e4ef7b9ab101f0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	642fb4fa7583daff1a442cbabcee69e6	34698416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c3821fa457738f5e4a9382c43f	28982645	SIG
Windows help file	Windows		d6399957 3a2c503b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	5b09c3cf6d9ec0b9a8e83184a43f29a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702be6b0adf76debdb30c43a583e5634d00	26880368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	2bc31c6088bbd71ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3bd1f8841679fda9413357413fd8	6741626	SIG
Windows x86 executable installer	Windows		33cc802f942a54446a3db4514763947f9	25663848	SIG
Windows x86 web-based installer	Windows		1b670c5a9d317df82c30983ea371d87c	1324606	SIG

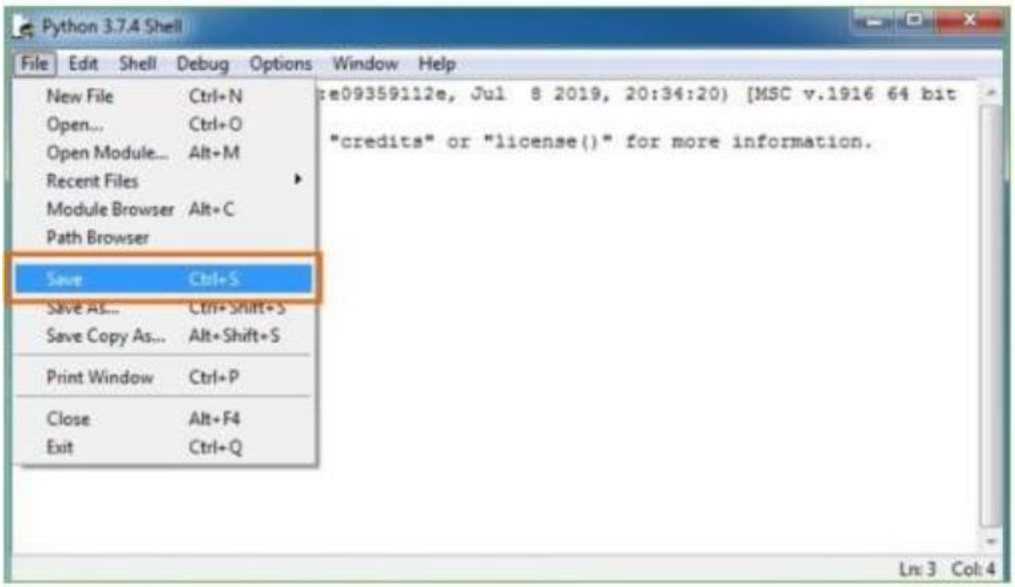


Install: Click **"Install Now"**.

Verify: Open the **Command Prompt** (cmd) and type python --version. You should see the version number.







Method B: Microsoft Store

Open the **Microsoft Store** app.
Search for **"Python"** and select the latest version (e.g., 3.13).
Click **"Get"** or **"Install"**. This method automatically handles the "PATH" for you.

Installation on macOS

Macs often come with an older version of Python pre-installed by the system. You should install a fresh version for your own work.

Method A: Official Installer

Download: Visit python.org/downloads and download the **macOS 64-bit universal2 installer**.

Install: Open the .pkg file and follow the wizard. You may need to enter your Mac password.

Verify: Open the **Terminal** and type python3 --version. (Note: on Mac, you usually need to use the command python3 instead of just python).

Method B: Homebrew (For Developers)

If you use the Homebrew package manager, it is much easier to manage updates.

Open **Terminal**.

Type brew install python and press Enter.

This will install the latest version and the package manager pip.

9.2 SOURCE CODE

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from flask import Flask, render_template, request
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.neighbors import KNeighborsClassifier

df=pd.read_csv('diabetes.csv')
df.columns

df.isna().sum()
df.head(10)
df.info()
df.describe()

X=df.iloc[:,:-1]
y=df.iloc[:,-1]

print(X)
print(y)



X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=
42)
print(X_train)
print(y_train)

st=StandardScaler()
X_train=st.fit_transform(X_train)
X_test=st.transform(X_test)
```

```
dt=DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)
dt.fit(X_train,y_train)
ypred=dt.predict(X_test)
ypred

print('accuracy score for decision tree
classifier',accuracy_score(y_test,ypred))
print(confusion_matrix(y_test,ypred))

cm=confusion_matrix(y_test,ypred)

plt.figure(figsize=(10,10))
sns.heatmap(cm, annot=True, fmt='d',cmap='Reds',
xticklabels=['predicted healty','predicted diabetes'],
            yticklabels=['actual healthy','actual diabetes'])
plt.title('Diabetes Prediction')
plt.xlabel('predicted health')
plt.ylabel('actual health')
plt.show()

plt.figure(figsize=(25,25))
plot_tree(dt, feature_names=X.columns,filled=True, rounded=True, fontsize=12,
class_names=['healthy','diabetes'])
plt.show()

gb=GradientBoostingClassifier(n_estimators=100,
learning_rate=0.2,max_depth=5,random_state=42)

gb.fit(X_train,y_train)
ypred=gb.predict(X_test)
print(ypred)

print('accuracy score for gradient boosting
classifier',accuracy_score(y_test,ypred))

choosen_tree=gb.estimators_[0,0]

plt.figure(figsize=(50,50))
plot_tree(choosen_tree, feature_names=X.columns, filled=True, rounded=True,
class_names=['healthy','diabetes'],
          fontsize=12)
plt.show()

ad=AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=1),n_estimato
rs=50,learning_rate=0.1,random_state=42)
ad.fit(X_train,y_train)
```

```
ypred=ad.predict(X_test)
print(ypred)

print('accuracy score for adaboost classifier',accuracy_score(y_test,ypred))

choosen_tree=ad.estimators_[0]
plt.figure(figsize=(50,50))
plot_tree(choosen_tree,  feature_names=X.columns,  filled=True,  rounded=True,
class_names=['healty','diabetes'],
          fontsize=12)
plt.show()

rfc=RandomForestClassifier(n_estimators=50,criterion='entropy',random_state=42
)
yf=rfc.fit(X_train,y_train)
y_pred=yf.predict(X_test)
y_pred

print('accuracy score for random forest',accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))

cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(10,10))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues',
            xticklabels=['predicted healthy','predicted diabetes'],
            yticklabels=['actual healty','predicted diabetes'])
plt.title('Diabetes Prediction')
plt.xlabel('Predicted Health')
plt.ylabel('Actual Health')
plt.show()

chosen_tree=rfc.estimators_[0]
plt.figure(figsize=(50,50))
plot_tree(chosen_tree,feature_names=X.columns,filled=True
,rounded=True,class_names=['healthy','diabetes'],
            fontsize=10)
plt.show()



choosen_tree=rfc.estimators_[1]
plt.figure(figsize=(50,50))
plot_tree(choosen_tree,                    feature_names=X.columns,filled=True,
rounded=True,class_names=['healthy','diabetes'],
          fontsize=12)
plt.show()
```

```
choosen_tree=rfc.estimators_[2]
plt.figure(figsize=(75,75))
plot_tree(choosen_tree,feature_names=X.columns,filled=True,rounded=True,class_
names=['healthy','diabetes'],
          fontsize=12)
plt.show()


base_estimator=DecisionTreeClassifier()
bc=BaggingClassifier(estimator=base_estimator,n_estimators=10,max_samples=0.8,
random_state=42)

bc.fit(X_train,y_train)
ypred=bc.predict(X_test)
print(ypred)

print('accuracy score for bagging classifier',accuracy_score(y_test,ypred))
choosen_tree=bc.estimators_[0]
plt.figure(figsize=(50,50))
plot_tree(choosen_tree,  feature_names=X.columns,  filled=True,  rounded=True,
class_names=['healty','diabetes'],
          fontsize=12)
plt.show()

df.columns
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
plt.show()

df1=df[['BloodPressure','Glucose','Insulin','Age']]
df1.head(10)
df1.isna().sum()

df2=df1['outcome']=df1.apply(lambda    x:    1    if    x['Glucose']>=140    or
x['Insulin']>=25 else 0, axis=1)
print(df2)

df1.head(30)

X=df1.iloc[:,:-1].values
y=df1.iloc[:,-1].values
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=
42)

df1.describe()
sd=StandardScaler()
X_train=sd.fit_transform(X_train)
```

```
X_test=sd.transform(X_test)
dt=DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)

dt.fit(X_train,y_train)
ypred=dt.predict(X_test)
print(ypred)
print(accuracy_score(y_test,ypred))
print(confusion_matrix(y_test,ypred))

X=df[['BloodPressure','Glucose','Insulin','Age']]
plt.figure(figsize=(10,10))
plot_tree(dt,feature_names=X.columns,        filled=True,        rounded=True,
class_names=['healthy','diabetes'],fontsize=12 )
plt.show()

X=df[['Glucose','BloodPressure','Insulin','Age']].values
y=df.iloc[:,-1].values
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=
42)

scaler=StandardScaler()
scaler_data=scaler.fit_transform(X_train)
scaler_data1=scaler.transform(X_test)

pca=PCA(n_components=2)
pca_data=pca.fit_transform(scaler_data)
pca_data1=pca.transform(scaler_data1)

print(pca_data)
rtf=RandomForestClassifier(n_estimators=150,criterion='entropy',random_state=4
2)
rtf.fit(pca_data,y_train)
ypred=rtf.predict(pca_data1)
print(ypred)
print(accuracy_score(y_test,ypred))
print(confusion_matrix(y_test,ypred))

X=df[['Glucose','BloodPressure','Insulin','Age']]
chosen_tree=rtf.estimators_[0]
plt.figure(figsize=(55,55))


plot_tree(chosen_tree,feature_names=X.columns,    filled=True,    rounded=True,
class_names=['healthy', 'diabetes'],
          fontsize=12)
plt.show()
```

```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=
42)
sd=StandardScaler()
X_train=sd.fit_transform(X_train)
X_test=sd.transform(X_test)

lr=LogisticRegression(random_state=42)
lr.fit(X_train,y_train)
ypred=lr.predict(X_test)
svc=SVC(random_state=42)

svc.fit(X_train,y_train)
ypred1=svc.predict(X_test)
#bar plot
plt.figure(figsize=(10,8))
ax=sns.countplot(x=ypred,palette=['blue','red'])
plt.title("Diabetes disease prediction using logistic regression", fontsize=14)
plt.xlabel('Predicted Class', fontsize=12)
plt.ylabel('Number of Predictions', fontsize=12)
plt.xticks([0,1],['Class 0', 'Class 1'])
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.show()

print('accuracy score for values predicted by logistic
regression',accuracy_score(y_test,ypred))
print('accuracy score for values predicted by support vector
classifier',accuracy_score(y_test,ypred))

print('confusion matrix for values predicted by logistic regression')
print(confusion_matrix(y_test,ypred))
print('confusion matrix for values predicted by support vector machine')
print(confusion_matrix(y_test,ypred))
voting_clf=VotingClassifier(estimators=[('lr',lr),('svm',svc)],voting='hard')
voting_clf.fit(X_train,y_train)


ypred=voting_clf.predict(X_test)
print(ypred)
print('voting classifier hard voting prediction',accuracy_score(y_test,ypred))
results={'model':[],'accuracy':[]}
for name,clf in [('lr',lr),('svm',svc)]:
    ypred=voting_clf.predict(X_test)
```

```
    results['model'].append(name)
    results['accuracy'].append(accuracy_score(y_test,ypred))
#model names
print(results['model'])
#model accuracy score
print(results['accuracy'])
print()
df_results=pd.DataFrame(results)
print(df_results)

plt.figure(figsize=(10,10))
ax=sns.countplot(x=ypred, palette=['blue','red'])
plt.title('Diabetes Disease Prediction using VotingClassifier(hard)')
plt.xlabel('predicted class')
plt.ylabel('number of predictions')
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.show()

svc=SVC(probability=True, random_state=23)
voting_clf=VotingClassifier(estimators=[('lr',lr),('svm',svc)], voting='soft')
voting_clf.fit(X_train,y_train)

ypred=voting_clf.predict(X_test)
print(ypred)
print('voting classifier soft voting prediction',accuracy_score(y_test,ypred))

results={'model':[],'accuracy':[]}
ypred=[]
for m,a in [('lr',lr),('svm',svc)]:
    ypred=clf.predict(X_test)
    results['model'].append(m)
    results['accuracy'].append(accuracy_score(y_test,ypred))

print(results['model'])
print(results['accuracy'])
print()



#probability distribution of VotingClassifier (Soft voting)
sns.kdeplot(ypred[y_test==0],label='class 0')
sns.kdeplot(ypred[y_test==1],label='class 1')
plt.legend()
plt.title('probability distribution for (soft voting)')
plt.show()
```

```
base_models=[('lr',lr),('svm',svc)]
meta_model=LogisticRegression(random_state=42)

sv=StackingClassifier(estimators=base_models,final_estimator=meta_model,cv=5)
sv.fit(X_train,y_train)
ypred=sv.predict(X_test)
print(ypred)

print('stacking classifier prediction',accuracy_score(y_test,ypred))
knn=KNeighborsClassifier(n_neighbors=5)
base_models=[('lr',lr),('svm',svc),('knn',knn)]
meta_model=LogisticRegression(random_state=42)

stacking_clf=StackingClassifier(estimators=base_models,final_estimator=meta_mo
del,cv=5)
stacking_clf.fit(X_train,y_train)
ypred=stacking_clf.predict(X_test)
print(ypred)
print('stacking classifier prediction',accuracy_score(y_test,ypred))
print(confusion_matrix(y_test,ypred))
stack_clf=StackingClassifier(estimators=base_models,
final_estimator=meta_model, cv=10)
stack_clf.fit(X_train,y_train)
ypred=stack_clf.predict(X_test)
print(ypred)
print('stacking classifier prediction',accuracy_score(y_test,ypred))
sns.kdeplot(ypred[y_test == 0], label='Actual Class 0', shade=True)
sns.kdeplot(ypred[y_test == 1], label='Actual Class 1', shade=True)
plt.xlabel('Predicted Probability of Class and 0 and 1')
plt.title('Probability Distribution by Class')
plt.legend()
plt.show()

app=Flask(__name__,template_folder='diabetes-check-app')
@app.route('/',methods=['POST','GET'])
def home():
  try:
    prediction=""
    prediction1=""
    age1=None
    blood1=None
    glucose1=None
    insulin1=None
    if request.method=='POST':
      age=int(request.form.get('age'))
      if age<=20:
        raise ValueError('age should be greater then 20')
```

```
        bloodpressure=int(request.form.get('bloodpressure'))
        glucose= int(request.form.get('glucose'))
        insulin=int(request.form.get('insulin'))
        age1=f'Age: {age}'
        blood1=f'BloodPressure: {bloodpressure}'
        glucose1=f'Glucose: {glucose}'
        insulin1=f'Insulin: {insulin}'
        print('age: ',age)
        print('bloodpressure: ',bloodpressure)
        print('glucose: ',glucose)
        print('insulin: ',insulin)


model=RandomForestClassifier(n_estimators=150,criterion='gini',max_depth=3,ran
dom_state=42)
        model.fit(X_train,y_train)
        ypred=model.predict([[bloodpressure,glucose,insulin,age]])
        print()
        print()
        if ypred==1:
          prediction='Diabetes'
          prediction1=f'result: {prediction}'
        else:
          prediction='Healthy'
          prediction1=f'Result: {prediction}'
    except ValueError as e:
      if request.method=='POST':
        age=int(request.form.get('age'))
        bloodpressure=int(request.form.get('bloodpressure'))
        glucose=int(request.form.get('glucose'))
        insulin=int(request.form.get('insulin'))
        prediction=""


        print('age: ',age)
        print('bloodpressure: ',bloodpressure)
        print('glucose: ',glucose)


model=RandomForestClassifier(n_estimators=150,criterion='entropy',random_state
=42)
        model.fit(X_train,y_train)
        ypred=model.predict([[glucose,bloodpressure,insulin,age]])

        print()
        print()
        if ypred==1:
```

```
        print(ypred)
        prediction='Diabetes'
        prediction1=f'Result: {prediction}'
      else:
        print(ypred)
        prediction='Healthy'
        prediction1=f'Result: {prediction}'
  return
render_template('data.html',age=age1,blood=blood1,glucose=glucose1,insulin=ins
ulin1,result=prediction,result1=prediction1)


if __name__ == '__main__':
  app.run()
```

JDBC Source Code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class App{
    public static void main(String[] args) {
        try {
            Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/my_web_data","ro
ot","Firewar@@2022");
            if(conn!=null){
                System.out.println("Connection Successful...");
            }
            Statement st=conn.createStatement();

             //decisiontree prediction confusionmatrix
            st.executeUpdate("insert           into           decision_tree_cn
values(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\decisio
n_tree\\tree\\cn.png');");
            //decisionTreeClassifier,ababoosting, gradientboosting
            st.executeUpdate("insert             into             decision_tree
values(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\decisio
n_tree\\tree\\dt.png','C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_predic
tion\\decision_tree\\ada\\adt.png','C:\\Users\\tonyr\\OneDrive\\py\\diabetes_d
isease_prediction\\decision_tree\\gradient\\gt.png');");
            //pca
            st.executeUpdate("insert                  into                  pca
values(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\pca\\n"
+ //

"_feature_pred\\psdt.png','pca/pca_an/pcadt.png');");
            // random_forest_cn
            st.executeUpdate("insert           into           random_forest_cn
values(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\random_
forest\\forestclassifier\\cn.png');");

           //random_forest_classifier
           st.executeUpdate("insert             into             random_forest
values(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\random_
```

```
forest\\forestclassifier\\rfc1.png'),(2,'C:\\Users\\tonyr\\OneDrive\\py\\diabe
tes_disease_prediction\\random_forest\\forestclassifier\\rfc2.png'),(3,'C:\\Us
ers\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\random_forest\\forestcl
assifier\\rfc3.png');");

          // BaggingClassifer
            st.executeUpdate("insert                into                bagging
values(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\random_
forest\\baggingclassifier\\bcdt.png');");
            //Stacking logisticRegression, StackingClassifier
            st.executeUpdate("insert                into                stacking
values(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\stackin
g\\logistic_regression\\lrcount.png','C:\\Users\\tonyr\\OneDrive\\py\\diabetes
_disease_prediction\\stacking\\stackingclassifier\\stackclass-pdt.png'); ");
            //stacking votingClassifier
            st.executeUpdate("insert         into         voting         values
(1,'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\stacking\\vot
ingclassifier\\voting(hard)-
count.png','C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\stack
ing\\votingclassifier\\voting(soft)-pbd.png');");

           System.out.println("data saved successfully");
            conn.close();

        } catch (SQLException e) {
        }

    }
}
```

Deployment Source Code

```
from flask import Flask, render_template, send_file
import os
app=Flask(__name__,template_folder='.')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/decisiontree')
def decisionfunction():
    return render_template('/boosting/decisiontree.html',image_name='dt.png')
@app.route('/dt')
def decision():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\d
ecision_tree\\tree\\dt.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')

@app.route('/gradientboosting')
def gradientfunction():
    return
render_template('/boosting/gradientboosting.html',image_name='gt.png')
@app.route('/gb')
def gradient():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\d
ecision_tree\\gradient\\gt.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')

@app.route('/adaboosting')
def adafunction():
    return render_template('/boosting/adaboosting.html',image_name='adt.png')
@app.route('/ab')
def ada():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\d
ecision_tree\\ada\\adt.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')
```

```
@app.route('/decisiontreeCn')
def decisiontreecnfunction():
    return render_template('/confusion
matrix/decisiontreecn.html',image_name='dtcn.png')
@app.route('/dtcn')
def dtcnfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\d
ecision_tree\\tree\\cn.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')


@app.route('/randomforestCn')
def randomforestcnfunction():
    return render_template('/confusion
matrix/randomforestcn.html',image_name='rfcn.png')
@app.route('/rfcn')
def rfcnfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\r
andom_forest\\forestclassifier\\cn.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')

@app.route('/randomforest')
def randomforestfunction():
    return render_template('/bagging/randomforest.html',image_name='rfc.png')


@app.route('/rfc11')
def rfc1():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\r
andom_forest\\forestclassifier\\rfc1.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')


@app.route('/rfc12')
def rfc2():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\r
andom_forest\\forestclassifier\\rfc2.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')
```

```
@app.route('/rfc13')
def rfc3():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\r
andom_forest\\forestclassifier\\rfc3.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')


@app.route('/baggingclassifier')
def baggingclassifierfunction():
    return
render_template('/bagging/baggingclassifier.html',image_name='bgc.png')


@app.route('/bgc')
def bgcfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\r
andom_forest\\baggingclassifier\\bcdt.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')



@app.route('/logisticregression')
def logisticregressionfunction():
    return
render_template('/stacking/logisticregression.html',image_name='lrcount.png')


@app.route('/lrcount')
def lrfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\s
tacking\\logistic_regression\\lrcount.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')




@app.route('/stackingclassifier')
def stackingclassifierfunction():
    return
```

```
render_template('/stacking/stackingclassifier.html',image_name='stcl.png')



@app.route('/stcl')
def stclfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\s
tacking\\stackingclassifier\\stackclass-pdt.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')



@app.route('/votingclassifier')
def votingclassifierfunction():
    return
render_template('/stacking/votingclassifier.html',image_name='vtcl.png')


@app.route('/vtclhard')
def vtclhardfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\s
tacking\\votingclassifier\\voting(hard)-count.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')







@app.route('/vtclsoft')
def vtclsoftfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\s
tacking\\votingclassifier\\voting(soft)-pbd.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')


@app.route('/pcanfeatures')
```

```
def pcanfeaturefunction():
    return render_template('/PCA/pcaNfeatures.html',image_name='pcancl.png')


@app.route('/pcanfeat')
def pcanfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\p
ca\\n_feature_pred\\psdt.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')


@app.route('/pcafeaturesAn')
def pcaanfunction():
    return render_template('/PCA/pcaan.html',image_name='pca_an.png')


@app.route('/pcanfeatan')
def pcacnfun():

absolute_path=r'C:\\Users\\tonyr\\OneDrive\\py\\diabetes_disease_prediction\\p
ca\\pca_an\\pcadt.png'
    if os.path.exists(absolute_path):
        return send_file(absolute_path,mimetype='jpeg/png')

if __name__=='__main__':
    app.run(debug=True)
```

CHAPTER – 10

RESULT/DISCUSSION

10.1 SYSTEM TEST

System testing is the level of software testing where a complete and integrated software product is tested to verify that it meets all specified requirements. It is a "black box" testing process, meaning the testers focus on the inputs and outputs without looking at the internal code structure.

In the software testing hierarchy, system testing happens after integration testing (where modules are combined) and before user acceptance testing (where the end-user tries the app).

Types Of Testing

Software testing is often categorized by how much "vision" the tester has into the source code. These methods are commonly referred to as Black-Box, White-Box, and Gray-Box testing. Unit Testing is a specific level of testing that usually employs one of these methods.

Black-Box Testing

In Black-Box testing, the tester has no knowledge of the internal code, structure, or implementation of the software. They interact with the application just like an end-user would.

Focus: Inputs and Outputs. You provide an input and check if the output matches the expected result.

Who does it? Usually independent Quality Assurance (QA) teams or end-users.

Techniques: Boundary Value Analysis (testing limits like 0 or 100), Equivalence Partitioning, and State Transition testing.

Example: Testing a login page by entering a username and password to see if you get redirected to the dashboard, without knowing how the database or authentication logic works.

White-Box Testing

Also known as "Glass-Box" or "Clear-Box" testing, this method gives the tester full access to the internal code, logic, and architecture.

Focus: Code coverage, logic paths, loops, and statement branches.

Who does it? Primarily developers during the coding phase.

Techniques: Statement coverage, Branch coverage, and Path testing.

Example: A developer writing a script to ensure that every if-else condition in a function is triggered at least once during the test.

Gray-Box Testing

This is a hybrid approach. The tester has "partial" knowledge of the internal workings—often they understand the database structure or system architecture but don't see the actual source code.

Focus: Integration and data flow between different parts of the system.

Who does it? Sophisticated QA testers or developers during integration.

Techniques: Regression testing, Matrix testing, and Pattern testing.

Example: Testing a web API where the tester knows the database schema. They send an API request and then manually check the database to see if the record was updated correctly.

Unit Testing

Unit Testing is the first level of software testing. It involves testing the smallest "units" of code—such as a single function, method, or class—in isolation from the rest of the system.

The Method: It is almost always White-Box testing because you must understand the function logic to test it.

The Goal: To catch bugs as early as possible, before they move to larger parts of the system where they become harder and more expensive to fix.

The Tools: Developers use frameworks like JUnit (for Java), PyTest (for Python), or NUnit (for .NET).

Integration Testing

Integration testing is the second level of software testing (after unit testing) where individual units or modules are combined and tested as a group. Its primary goal is to expose defects in the interaction between these modules, ensuring that data flows correctly and the interfaces communicate as expected.

Even if two modules work perfectly on their own, they may fail when joined due to mismatched data formats, timing issues, or broken API calls.

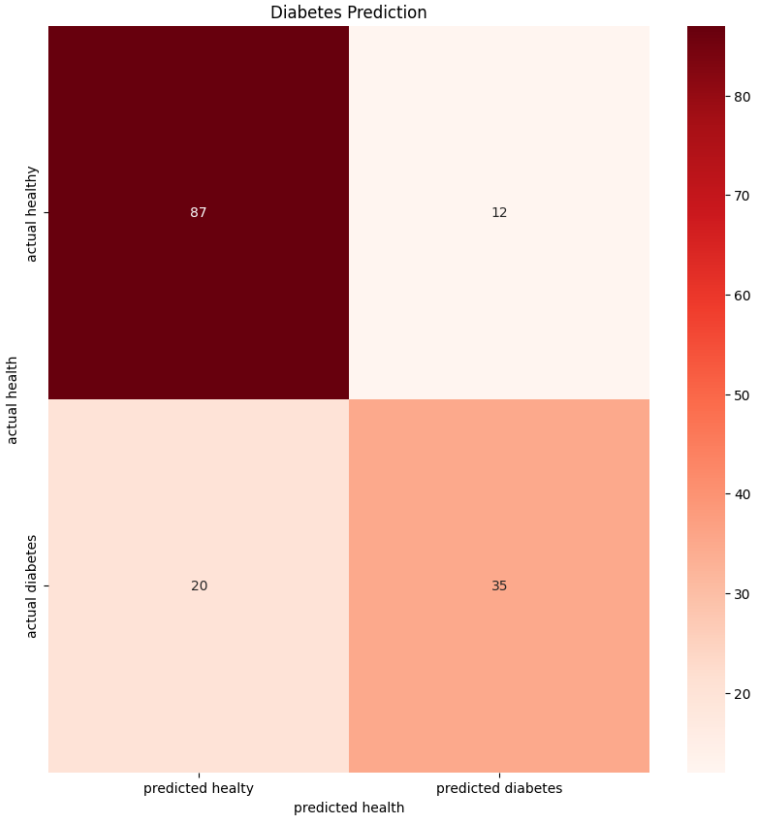
10.1.1 TEST CASES

Test case for Diabetes Disease prediction using Black-Box

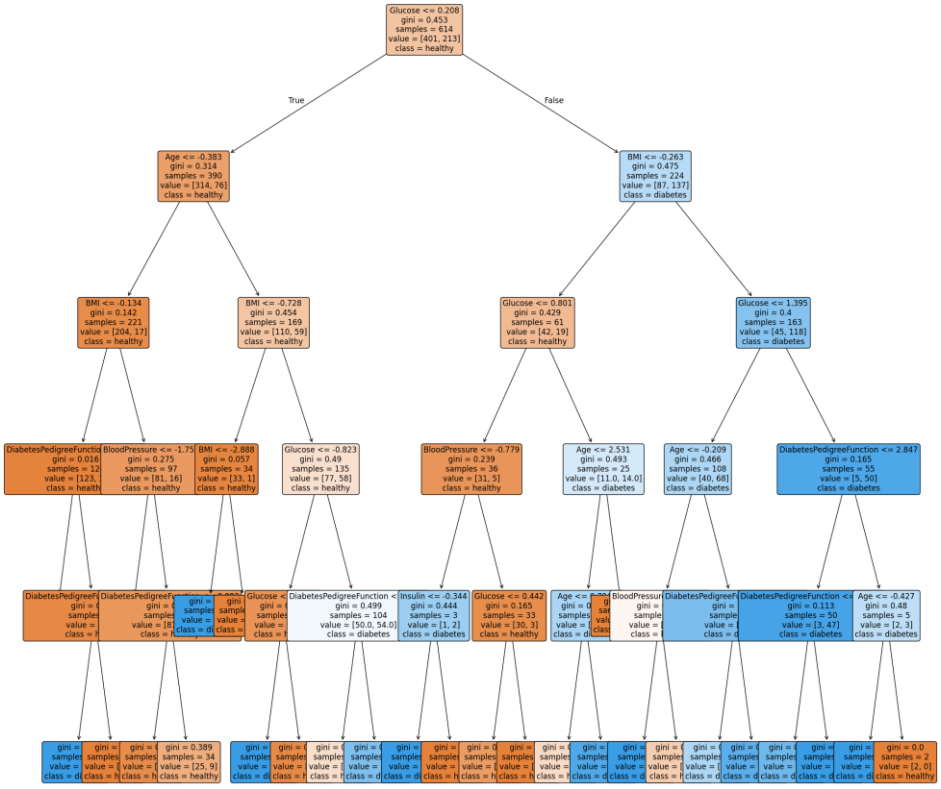
Test Case ID	Input Description	Age	Glucose	BP	Insulin	Expected Result
TC_01	**Valid Mid-Range** (Normal Case)	30	90	120	15	System accepts and saves data.
TC_02	**Min Boundary** (BVA)	1	50	80	0	System accepts and saves data.
TC_03	**Max Boundary** (BVA)	120	500	200	100	System accepts and saves data.
TC_04	**Invalid Low** (Below Range)	**0**	90	120	15	Error: "Age must be at least 1."
TC_05	**Invalid High** (Above Range)	30	**501**	120	15	Error: "Glucose value out of bounds."
TC_06	**Wrong Data Type**	30	90	**"High"**	15	Error: "Please enter a numeric value."
TC_07	**Negative Value**	30	90	120	**-5**	Error: "Insulin cannot be negative."

10.2 SCREENSHOTS

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age
0              6      148             72             35        0  33.6                     0.627   50
1              1       85             66             29        0  26.6                     0.351   31
2              8      183             64              0        0  23.3                     0.672   32
3              1       89             66             23       94  28.1                     0.167   21
4              0      137             40             35      168  43.1                     2.288   33
..           ...      ...            ...            ...      ...   ...                       ...  ...
763           10      101             76             48      180  32.9                     0.171   63
764            2      122             70             27        0  36.8                     0.340   27
765            5      121             72             23      112  26.2                     0.245   30
766            1      126             60              0        0  30.1                     0.349   47
767            1       93             70             31        0  30.4                     0.315   23
```

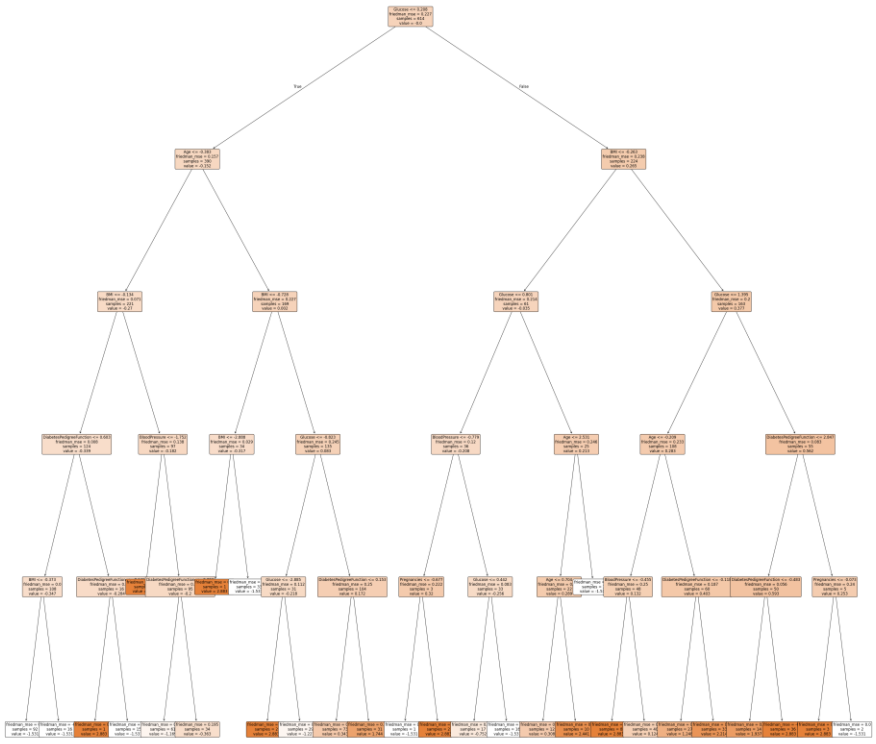



```
60             2       84              0              0        0   0.0                       0.304   21
618            9      112             82             24        0  28.2                       1.282   50
346            1      139             46             19       83  28.7                       0.654   22
294            0      161             50              0        0  21.9                       0.254   65
231            6      134             80             37      370  46.2                       0.238   46
..           ...      ...            ...            ...      ...   ...                         ...  ...
71             5      139             64             35      140  28.6                       0.411   26
106            1       96            122              0        0  22.4                       0.207   27
270           10      101             86             37        0  45.6                       1.136   38
435            0      141              0              0        0  42.4                       0.205   29
102            0      125             96              0        0  22.5                       0.262   21

[614 rows x 8 columns]
60     0
618    1
346    0
294    0
231    1
      ..
71     0
106    0
270    1
435    1
102    0
Name: Outcome, Length: 614, dtype: int64
accuracy score for decision tree classifier 0.7922077922077922
[[87 12]
 [20 35]]
```

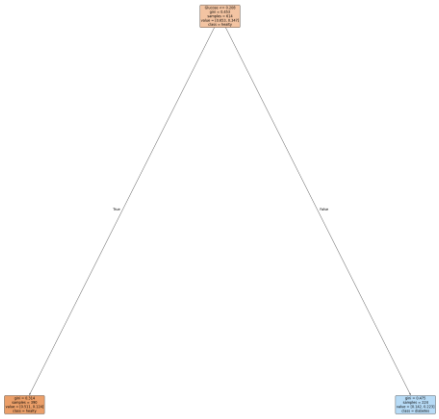



```
[614 rows x 8 columns]
60     0
618    1
346    0
294    0
231    1
      ..
71     0
106    0
270    1
435    1
102    0
Name: Outcome, Length: 614, dtype: int64
accuracy score for decision tree classifier 0.7922077922077922
[[87 12]
 [20 35]]
[1 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1
 1 0 1 0 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 0 1 1 0 0 0
 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 1 0 0 1 1 0 0 1 0 1 0
 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0
```

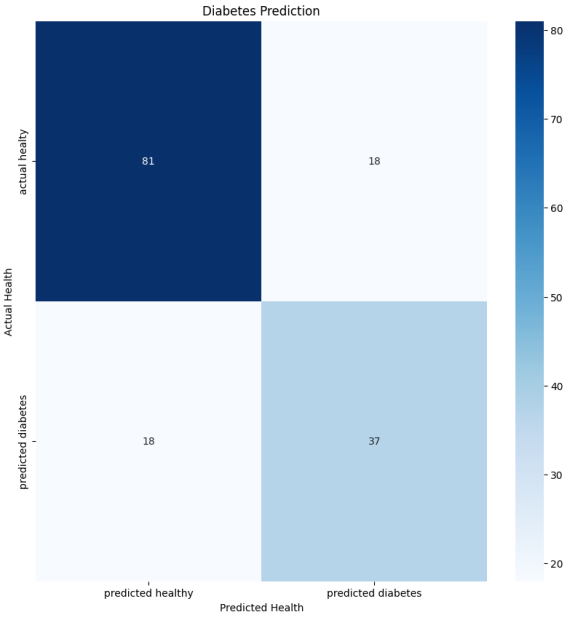
```
accuracy score for gradient boosting classifier 0.7402597402597403
[0 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 0 1 1
 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0
 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
```

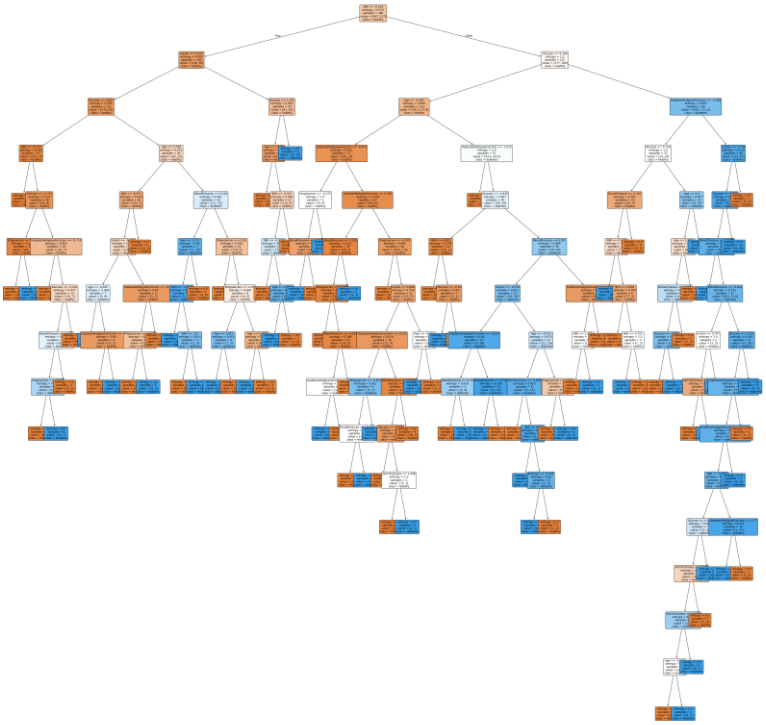


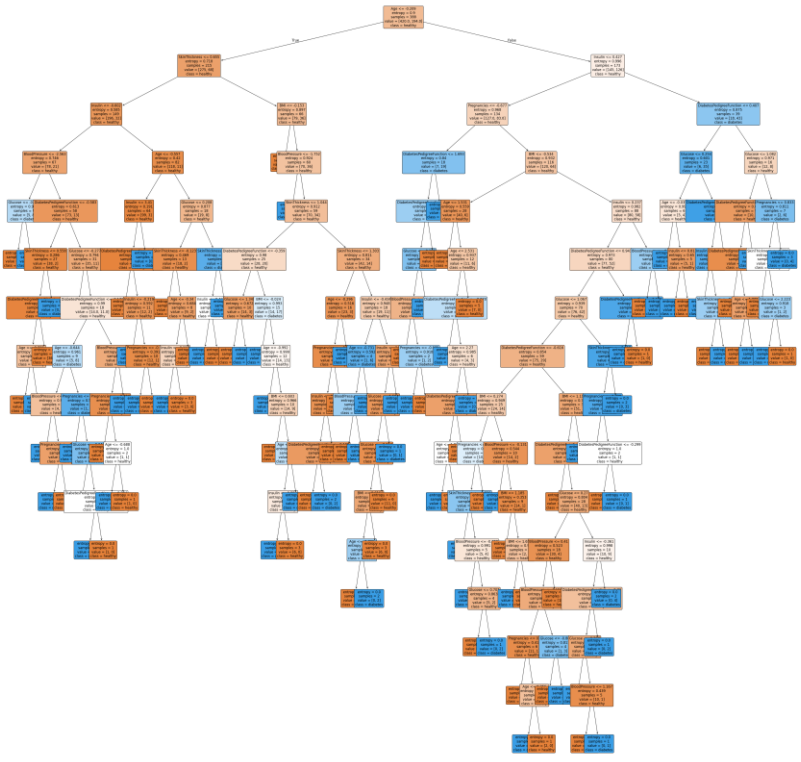
```
 0 1 0 0 0 0]
accuracy score for adaboost classifier 0.7792207792207793
accuracy score for random forest 0.7662337662337663
[[81 18]
 [18 37]]
```

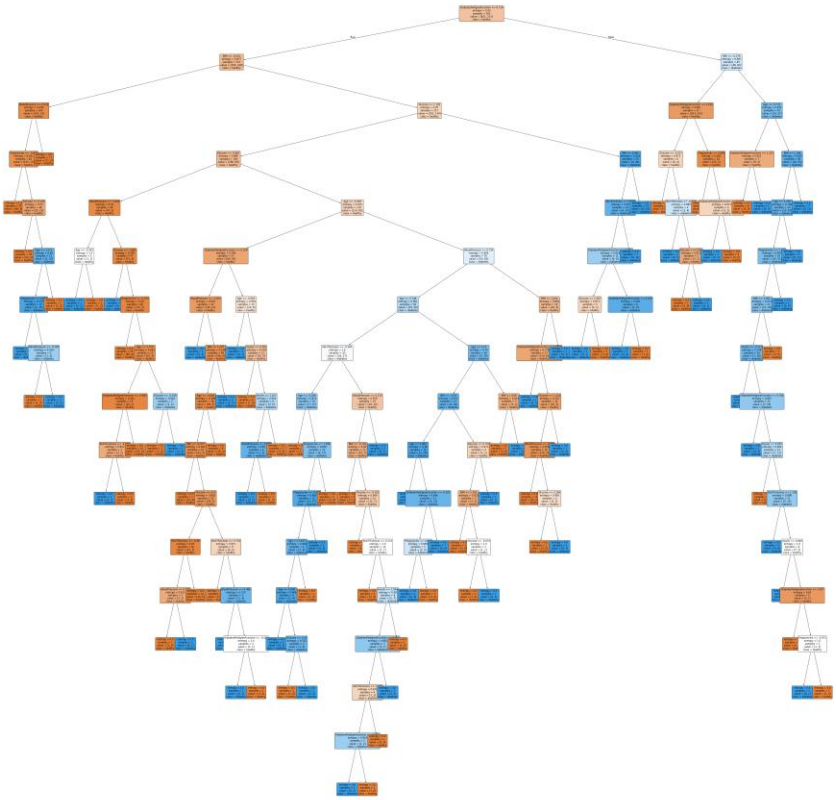


```
accuracy score for random forest 0.7662337662337663
[[81 18]
 [18 37]]
[0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1
 1 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0
 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
```

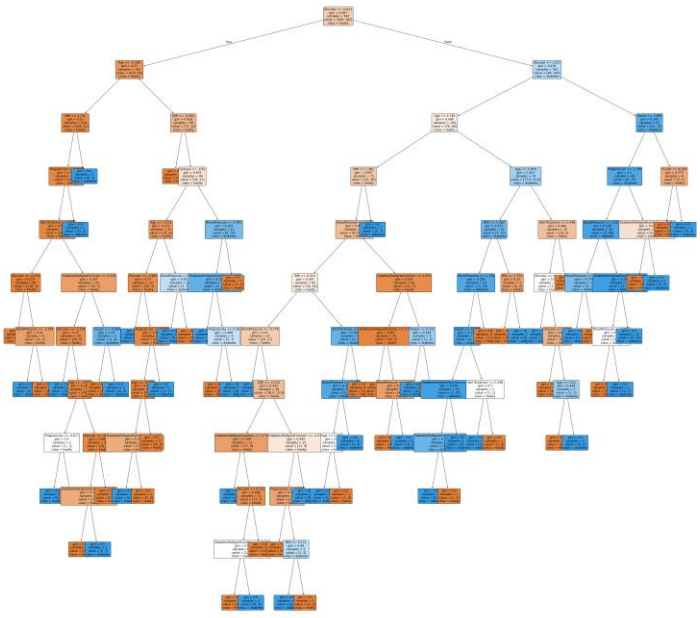




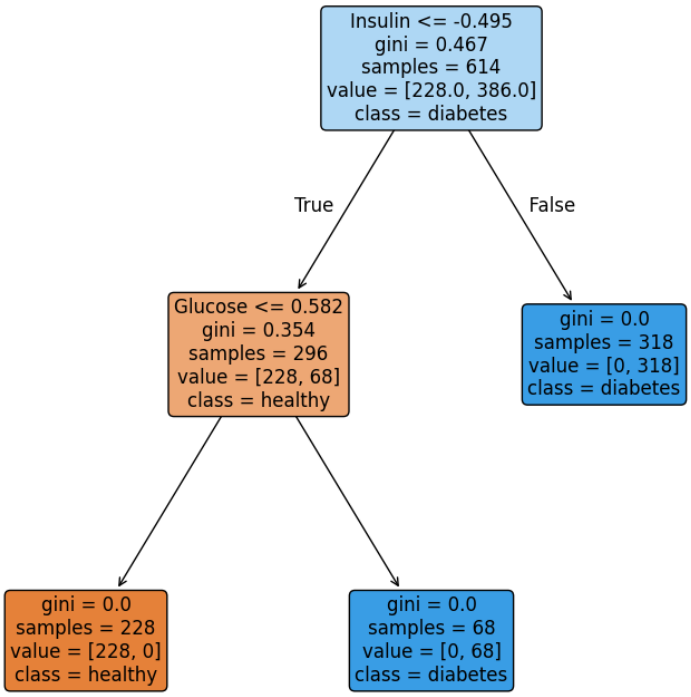


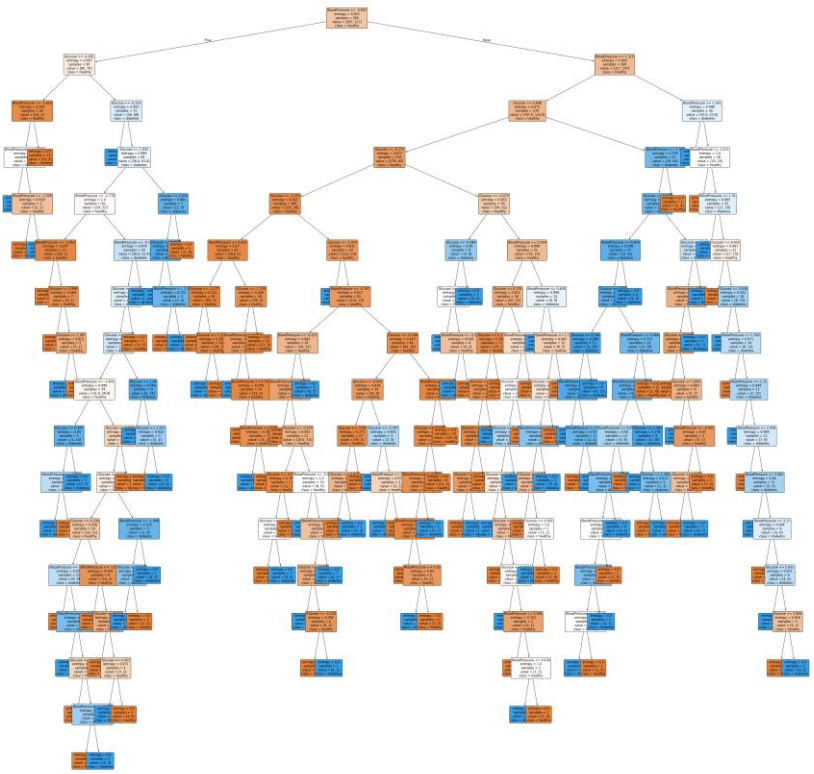


```
[0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1
 1 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1
 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
accuracy score for bagging classifier 0.7662337662337663
```

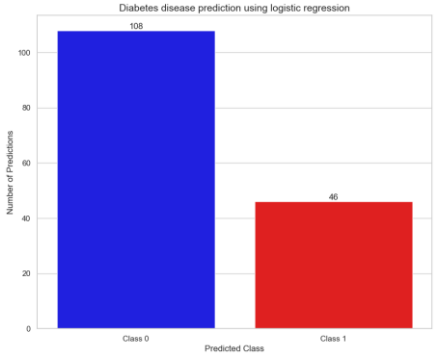


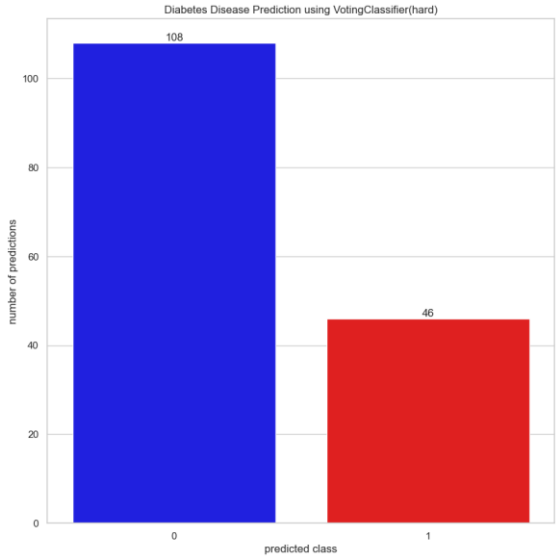
```
0      1
1      0
2      1
3      1
4      1
      ..
763    1
764    0
765    1
766    0
767    0
Length: 768, dtype: int64
[1 0 0 0 0 1 1 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1
 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1
 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 0 0
 1 0 1 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0
 0 1 1 0 1 1]
1.0
[[64  0]
 [ 0 90]]
[[-3.21472582  1.23103637]
 [ 0.55013979 -1.64684153]
 [-0.66730723  1.15013599]
 ...
 [-0.06284539 -1.21984621]
 [-1.76996766  1.25386619]
 [-0.05115164 -0.41458404]]
[0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1
 0 0 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 0
 0 0 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 0 0
```



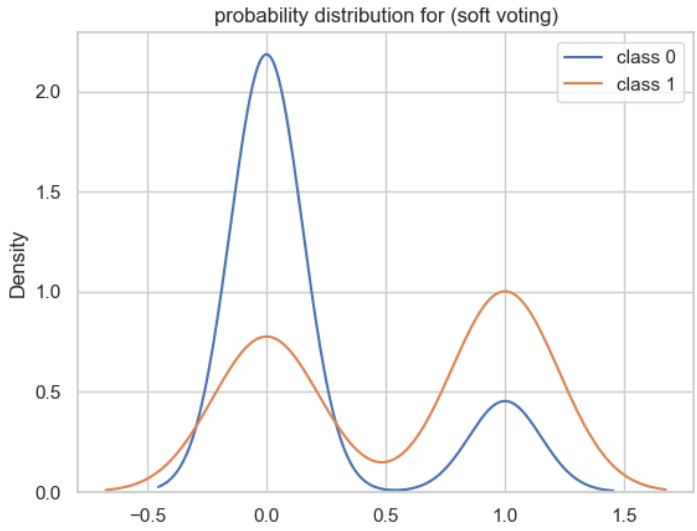


```
C:\Users\tonyr\OneDrive\py\personal\dib.py:225: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`
 and set `legend=False` for the same effect.

  ax=sns.countplot(x=ypred,palette=['blue','red'])
accuracy score for values predicted by logistic regression 0.7532467532467533
accuracy score for values predicted by support vector classifier 0.7532467532467533
confusion matrix for values predicted by logistic regression
[[79 20]
 [18 37]]
confusion matrix for values predicted by support vector machine
[[79 20]
 [18 37]]
[0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 0
 0 0 1 0 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0
 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
voting classifier hard voting prediction 0.7467532467532467
['lr', 'svm']
[0.7467532467532467, 0.7467532467532467]

  model  accuracy
0    lr  0.746753
1   svm  0.746753
C:\Users\tonyr\OneDrive\py\personal\dib.py:261: FutureWarning:
```

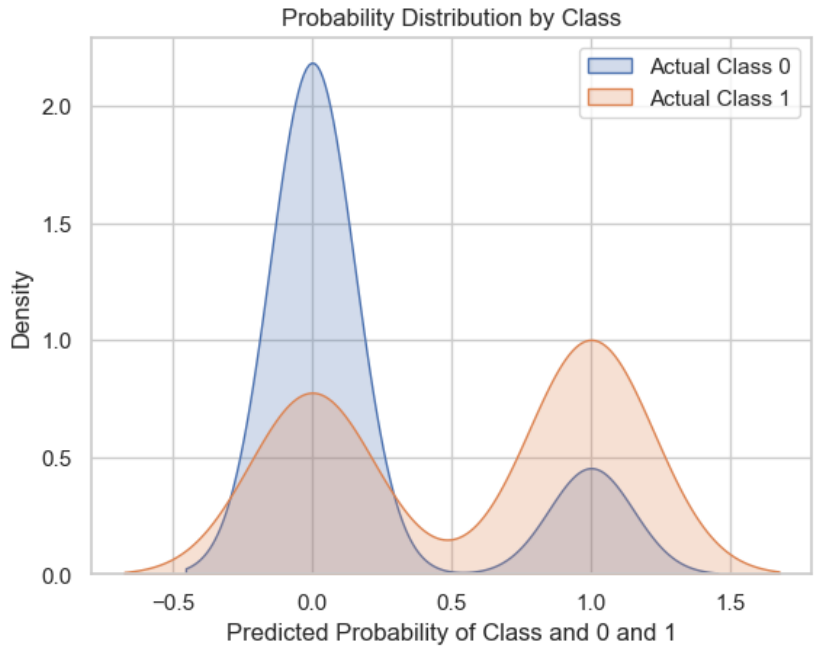




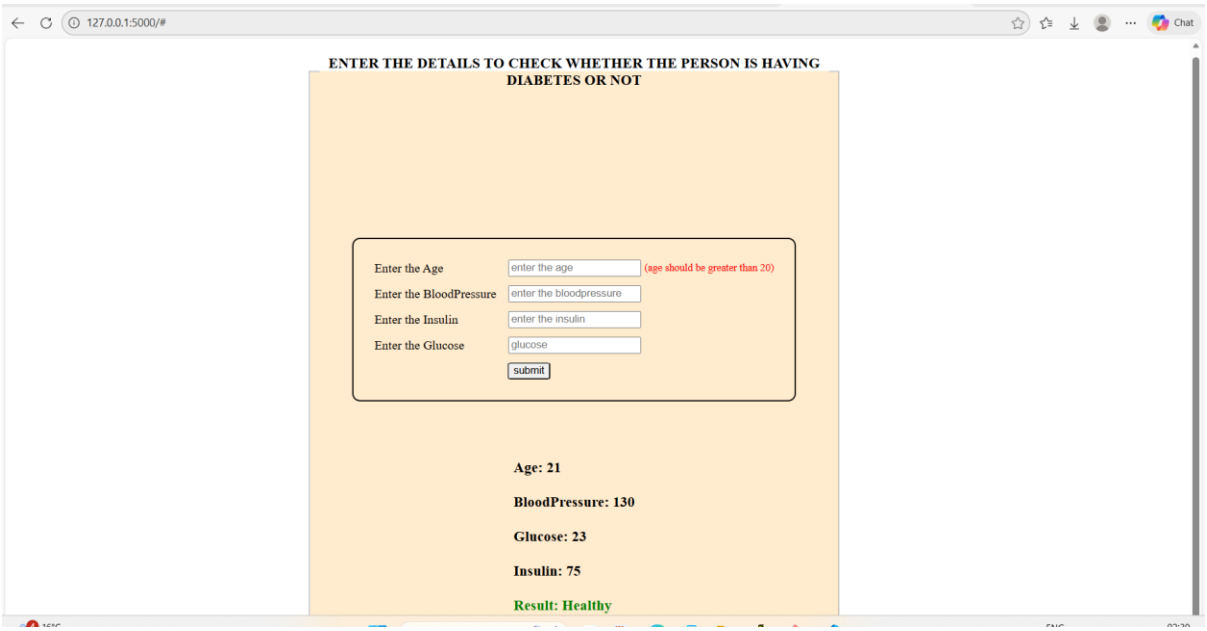

```
C:\Users\tonyr\OneDrive\py\personal\dib.py:261: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`
 and set `legend=False` for the same effect.

  ax=sns.countplot(x=ypred, palette=['blue','red'])
[0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1
 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0
 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
voting classifier soft voting prediction 0.7662337662337663
['lr', 'svm']
[0.7337662337662337, 0.7337662337662337]
```

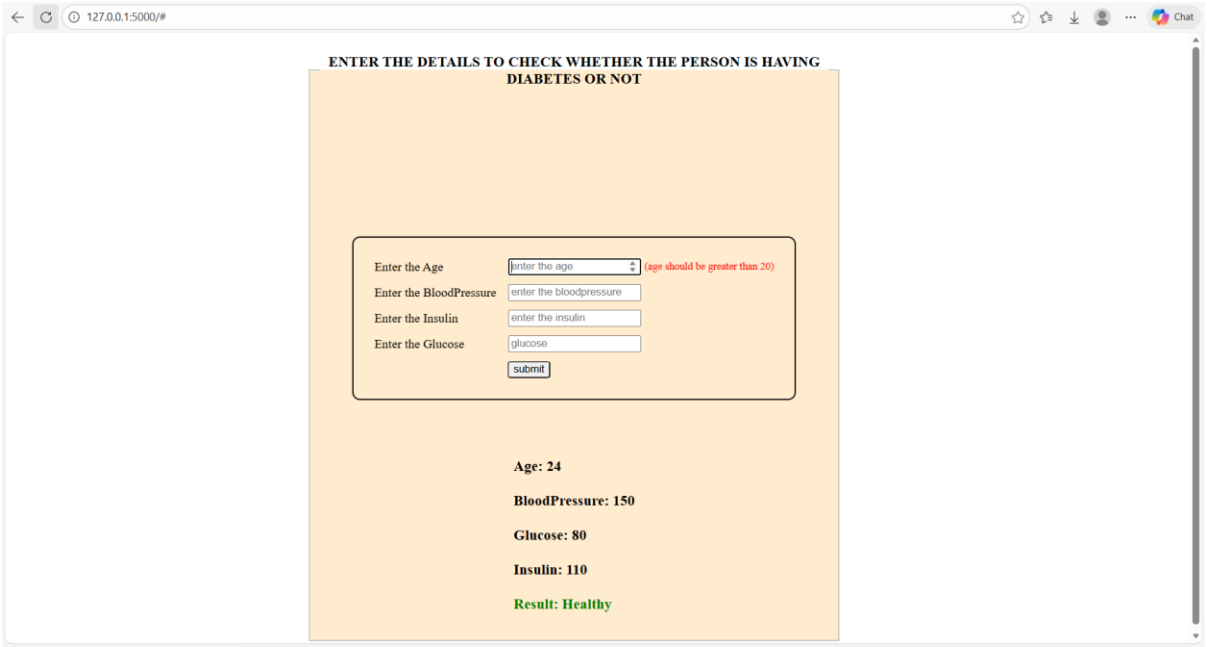

probability distribution for (soft voting)
class 0
class 1
Density
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5

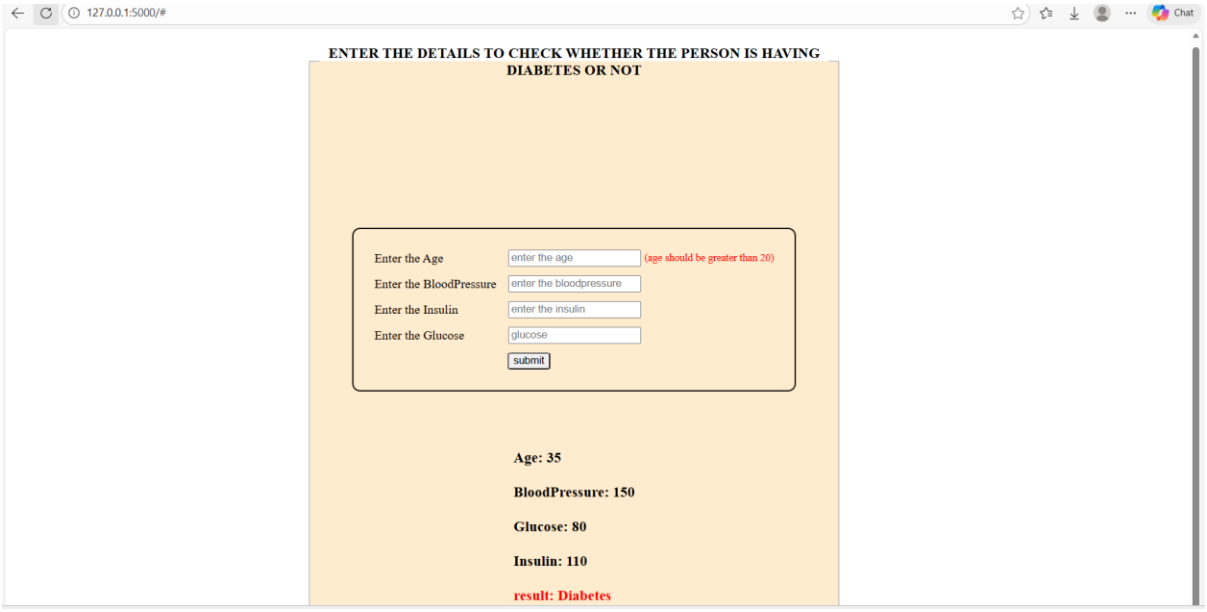
```
[0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1
 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0
 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
stacking classifier prediction 0.7662337662337663
[0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1
 0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0
 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
stacking classifier prediction 0.7467532467532467
[[82 17]
 [22 33]]
[0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1
 0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0
 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
stacking classifier prediction 0.7467532467532467
C:\Users\tonyr\OneDrive\py\personal\dib.py:319: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(ypred[y_test == 0], label='Actual Class 0', shade=True)
C:\Users\tonyr\OneDrive\py\personal\dib.py:320: FutureWarning:
```

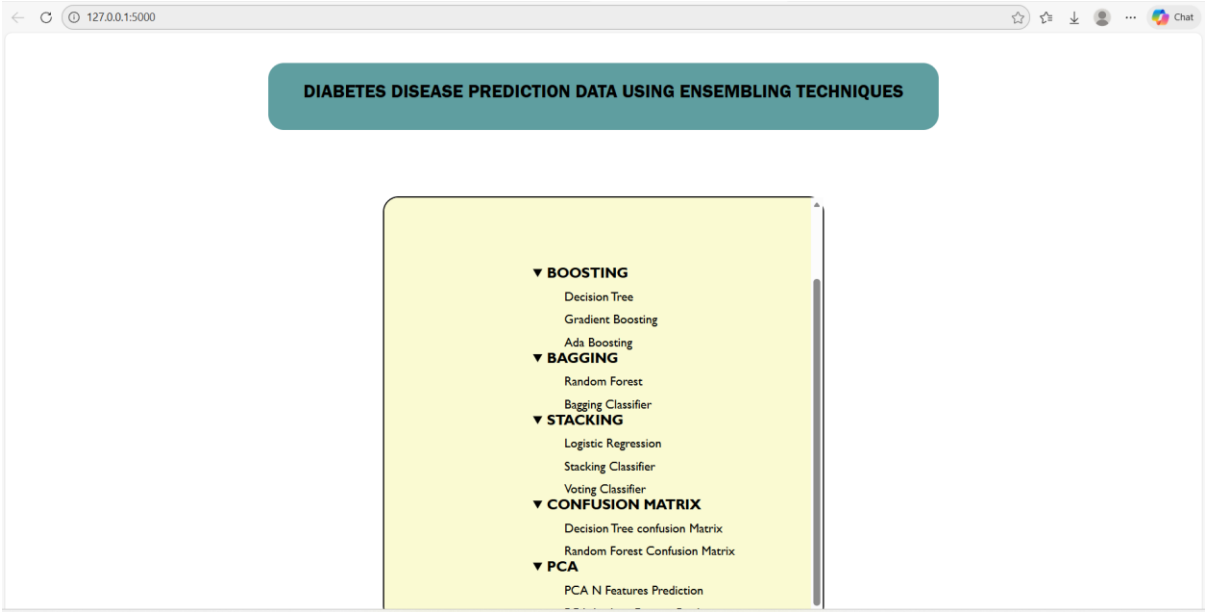



```
 * Serving Flask app 'dib'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```


127.0.0.1:5000/#
Chat
ENTER THE DETAILS TO CHECK WHETHER THE PERSON IS HAVING DIABETES OR NOT
Enter the Age
enter the age
(age should be greater than 20)
Enter the BloodPressure
enter the bloodpressure
Enter the Insulin
enter the insulin
Enter the Glucose
glucose
submit
Age: 21
BloodPressure: 130
Glucose: 23
Insulin: 75
Result: Healthy


127.0.0.1:5000/#
Chat
ENTER THE DETAILS TO CHECK WHETHER THE PERSON IS HAVING DIABETES OR NOT
Enter the Age
enter the age
(age should be greater than 20)
Enter the BloodPressure
enter the bloodpressure
Enter the Insulin
enter the insulin
Enter the Glucose
glucose
submit
Age: 24
BloodPressure: 150
Glucose: 80
Insulin: 110
Result: Healthy


127.0.0.1:5000/#
Chat
ENTER THE DETAILS TO CHECK WHETHER THE PERSON IS HAVING DIABETES OR NOT
Enter the Age
enter the age
(age should be greater than 20)
Enter the BloodPressure
enter the bloodpressure
Enter the Insulin
enter the insulin
Enter the Glucose
glucose
submit
Age: 35
BloodPressure: 150
Glucose: 80
Insulin: 110
result: Diabetes


127.0.0.1:5000
Chat
DIABETES DISEASE PREDICTION DATA USING ENSEMBLING TECHNIQUES
▼ BOOSTING
Decision Tree
Gradient Boosting
Ada Boosting
▼ BAGGING
Random Forest
Bagging Classifier
▼ STACKING
Logistic Regression
Stacking Classifier
Voting Classifier
▼ CONFUSION MATRIX
Decision Tree confusion Matrix
Random Forest Confusion Matrix
▼ PCA
PCA N Features Prediction

CHAPTER – 11

CONCLUSION

11.1 CONCLUSION

Ensemble machine learning has solidified its position as a superior diagnostic framework for diabetes prediction by effectively neutralizing the weaknesses of individual algorithms. By integrating diverse models like XGBoost, Random Forest, and Stacking Classifiers, these systems achieve near-perfect predictive accuracy—often exceeding **98%**—while significantly reducing the risks of overfitting and bias inherent in smaller medical datasets. This collaborative approach not only ensures more robust performance across diverse patient demographics but also provides a reliable foundation for data-driven clinical decision-making, ultimately leading to earlier interventions and a reduction in long-term diabetic complications.

The integration of ensemble machine learning represents a paradigm shift in chronic disease management, moving beyond the limitations of standalone algorithms to provide a highly resilient diagnostic tool. By strategically combining diverse models, researchers have successfully mitigated common data challenges such as class imbalance and high variance, consistently achieving accuracy rates as high as **97% to 98.5%**.

11.2 FUTURE SCOPE

The future of diabetes prediction is moving toward Explainable AI (XAI) and real-time biometric integration, shifting from static hospital records to continuous, proactive monitoring. Upcoming research will focus on making these "Black Box" ensemble models fully transparent through tools like SHAP, allowing doctors to understand the specific physiological triggers behind every high-risk alert. Furthermore, the integration of Federated Learning will allow global healthcare systems to collaboratively train powerful ensemble models without compromising patient privacy, while the inclusion of multi-omics data (genomics and metabolomics) will enable a "Precision Medicine" era where diabetes risk is predicted and managed based on an individual's unique genetic and lifestyle blueprint.

CHAPTER – 12

REFERENCES

1. **Dr. Rashi Rastogi**
    a. *Title:* "Effective Diabetes Prediction using an IoT-based Integrated Ensemble Machine Learning Framework" (2025).
2. **Shahid Mohammad Ganie**
    a. *Title:* "An Ensemble Machine Learning Approach for Predicting Type-II Diabetes Mellitus Based on Lifestyle Indicators" (2023/2024).
3. **Ifra Shaheen**
    a. *Title:* "Hi-Le and HiTCLe: Ensemble Learning Approaches for Early Diabetes Detection Using Deep Learning and Explainable Artificial Intelligence" (2024).
4. **Md. Alamin Talukder**
    a. *Title:* "Toward Reliable Diabetes Prediction: Innovations in Data Engineering and Machine Learning Applications" (2024).
5. **Dip Das**
    a. *Title:* "Diabetes Prediction using Ensemble Learning Techniques" (2025).
6. **McDonald Otieno Ogutu**
    a. *Title:* "Leveraging Machine Learning for Diabetes Prediction: Ensemble Model" (2025).
7. **Sarker S.**
    a. *Title:* "A Novel Two-Layer Hybrid Stacked Ensemble for Early-Stage Diabetes Classification" (2024).
8. **Aniket K. Shahade**
    a. *Title:* "Enhancing Diabetes Prediction Using Ensemble Machine Learning Model" (2025).
9. **Khondokar Oliullah**
    a. *Title:* "A Stacked Ensemble Machine Learning Approach for the Prediction of Diabetes" (2024).