Mini Project Report

On

**Multiple Disease Prediction using ML**

Submitted in partial fulfilment of the Requirements for the award of the degree

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

Submitted By

**MIDIVELLI BIPIN KUMAR (22EG512C10)**

**SAI YESHWANTH (22EG512C08)**

**GATTU PRANEETH (21EG112C11)**

Under the guidance of

**Ms. Y. Sowjanya**

**Asst. Professor**



**Department of Information Technology**

# ANURAG UNIVERSITY

**(Approved by AICTE and NBA Accredited)**

**Venkatapur (V), Ghatkesar (M), Medchal district, Hyderabad, Telangana,500088**

2022-2025

**ANURAG UNIVERSITY**

**(Approved by AICTE and NBA Accredited)**

**Venkatapur (V), Ghatkesar (M), Medchal district, Hyderabad, Telangana,500088 Department of Information Technology**



# CERTIFICATE

This is to certify that the project report entitled **“Multiple Disease Prediction using ML**” is a Bonafide work done and submitted by **Midivelli Bipin Kumar(22EG512C10), Sai Yeshwanth(22EG512C08), Gattu Praneeth(21EG112C11)** in partial fulfilment of the requirements for the award of the degree of B.Tech in **Information Technology** from **Anurag University**, Hyderabad during the academic year 2024-2025 and the Bonafide work has not been submitted elsewhere for the award of any other degree.

**Internal Guide H.O.D**

**Ms. Y. Sowjanya Dr. K. Sudheer Reddy**

**Asst. Professor Professor**

**Dean, Academics and Planning**

**External Examiner**

# ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Ms. Y. Sowjanya,** Asst Prof**,** Department of Information Technology, Anurag University, Ghatkesar, whose motivation in the field of software development has made us to overcome all hardships during the course of study and successful completion of project.

We would like to express our profound sense of gratitude to all for having helped us in completing this dissertation. We would like to express our deep-felt gratitude and sincere thanks to our guide **Ms. Y. Sowjanya,** Asst Prof**,** Department of Information Technology, **Anurag University**, Ghatkesar, for his skillful guidance, timely suggestions and encouragement in completing this project.

We extend our sincere thanks to **Dr. V. Vijaya Kumar**, Dean, School of Engineering, **Dr. K. Sudheer Reddy**, Dean, Academics and Planning, Head of the Department of Information Technology, of **Anurag University**, Venkatapur(V), Ghatkesar(M), R.R. Dist, for their encouragement and constant help.

Finally, we would like to express our heartfelt thanks to our parents who were very supportive both financially and mentally and for their encouragement to achieve our set goals.

|  |  |
| --- | --- |
| MIDIVELLI BIPIN KUMAR | (22EG512C10) |
| SAI YESHWANTH | (22EG512C08) |
| GATTU PRANEETH | (21EG112C11) |

# DECLARATION

This is to Certify that the project work entitled “Multiple Disease Prediction using ML” submitted to Anurag University in partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology (B-Tech), is an original work carried out by Midivelli Bipin Kumar(22EG512C10), Sai Yeshwanth(22EG512C08), Praneeth(21EG112C44) under the guidance of Ms. Y. Sowjanya, Asst Prof, Department of Information Technology, of Anurag University. This matter embodied in this project is a genuine work, done by the students and has not been submitted whether the university or to any other university/Institute for the fulfillment of the requirement of any course of study.

|  |  |
| --- | --- |
| MIDIVELLI BIPIN KUMAR | (22EG512C10) |
| SAI YESHWANTH | (22EG512C08) |
| GATTU PRANEETH | (21EG112C11) |

ABSTRACT

There are multiple techniques in machine learning that can in a variety of industries, do

predictive analytics on large amounts of data. Predictive analytics in healthcare is a difficult

endeavour, but it can eventually assist practitioners in making timely decisions regarding

patients' health and treatment based on massive data. Diseases like Breast cancer, diabetes, and

heart-related diseases are causing many deaths globally but most of these deaths are due to the

lack of timely check-ups of the diseases. The above problem occurs due to a lack of medical

infrastructure and a low ratio of doctors to the population. The statistics clearly show the same,

WHO recommended, the ratio of doctors to patients is 1:1000 whereas India's doctor-to-

population ratio is 1:1456, this indicates the shortage of doctors.

The diseases related to heart, cancer, and diabetes can cause a potential threat to

mankind, if not found early. Therefore, early recognition and diagnosis of these diseases can

save a lot of lives. This work is all about predicting diseases that are harmful using machine

learning classification algorithms. In this work, breast cancer, heart, and diabetes are included.

To make this work seamless and usable by the mass public, our team made a medical test web

application that makes predictions about various diseases using the concept of machine

learning. In this work, our aim to develop a disease-predicting web app that uses the concept of

machine learning-based predictions about various diseases like Breast cancer, Diabetes, and

Heart diseases.

This project focuses on developing a web-based application designed to predict the likelihood of diabetes, heart disease, and Parkinson's disease using machine learning models. The application provides an interactive and user-friendly interface, allowing individuals to input relevant health parameters such as blood sugar levels, cholesterol levels, or motor function scores, depending on the disease in question.

Leveraging pre-trained machine learning models, the system processes the input data and generates predictive diagnostic results. This predictive capability can help in identifying the early onset of these diseases, thereby enabling timely medical interventions and potentially improving patient outcomes. By integrating technology with healthcare, the project aims to facilitate better disease management and prevention.

**CHAPTERS PAGE NO­**

**List of Figures** 7

**List of Abbreviations** 8

1. **Introduction**
   1. Overview 9
   2. Objectives 9
   3. [Problem](#_TOC_250015) Formulation 9
   4. Scope of the Project 10
   5. Feasibility 10
   6. Requirements of the Project
      1. Software Requirements 11
      2. Hardware Requirements 12
2. [Requirements](#_TOC_250011)

2.1 Requirements 13

2.2 [Non-Functional Requirements 13](#_TOC_250010)

2.3 Functional Requirements 14

1. [Analysis](#_TOC_250008)

3.1 Use Case Model 16

3.2 Class Diagram 17

3.3 Sequence Diagram 18

3.4 Deployment Diagram 19

3.5 Component Diagram 20

1. Implementation 21
2. Testing Methodology 48
3. Conclusions and Future Work
   1. Results 53

6.2 Conclusion & Future Work 56

1. **References**  57

LIST OF FIGURES

Fig.4.2.1

SYSTEM ARCHITECTURE

11

FIG.4.3.2

CLASS DIAGRAM

13

FIG.4.3.2

USE CASE DIAGRAM

14

FIG.4.3.3

SEQUENCE DIAGRAM

15

FIG.4.3.4

COMPONENT DIAGRAM

16

FIG.4.3.5

DEPLOYMENT DIAGRAM

17

FIG.5.3.1

K VALUE GRAPH

30

FIG.8.1.1

DIABETES PREDICTION HOME PAGE

45

FIG.8.1.2

DIABETES PREDICTION RESULT PAGE

45

FIG.8.2.1

HEART DISEASE PREDICTION HOME PAGE

46

FIG.8.2.2

HEART DISEASE PREDICTION HOME PAGE

46

FIG.8.3.1

PARKINSON’S DISEASE PREDICTION HOME PAGE

47

FIG.8.3.2

PARKINSON’S DISEASE PREDICTION RESULT PAGE

47

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| Figure Number | Name Of Figure | Page No |
| 3.1.1 | Use diagram | 16 |
| 3.2.1 | Class diagram | 17 |
| 3.3.1 | Sequence diagram | 18 |
| 3.4.1 | Deployment diagram | 19 |
| 3.5.1 | Component diagram | 20 |
| 4.1 | Methodology | 21 |
| 4.2 | Model comparison for parkinson’s | 45 |
| 4.3 | Model comparison for diabetes | 46 |
| 4.4 | Model comparison for heart | 46 |
| 6.1.1 | Result of Diabetes Diseases | 53 |
| 6.1.2 | Result of Heart Diseases | 54 |
| 6.1.3 | Result of Parkinsons Diseases | 55 |

LIST OF ABBREVATIONS

* Srs - software requirements specification
* Strs - stakeholder requirements specification
* Syrs - system requirements specification
* Uml - unified modeling language
* Ml - machine learning
* Svm - support vector machine
* Knn - k-nearest neighbors
* Xgboost - extreme gradient boosting
* Adaboost - adaptive boostingof
* Ui - user interface
* Cpu - central processing unit
* Ram - random access memory
* Api - application programming interface
* Csv - comma-separated values

**1. INTRODUCTION**

### **1.1 Overview**

The **Multiple Disease Prediction System using Machine Learning** project aims to develop a web-based application that predicts diseases like diabetes, heart disease, and Parkinson's disease. The platform uses machine learning models to provide predictions based on user input data (e.g., medical history, health parameters). By leveraging pre-trained machine learning models, the system enhances the early diagnosis of multiple diseases, leading to timely medical interventions and improved patient outcomes. The application is built using Python and the Streamlit framework, which provides an interactive, user-friendly interface for non-technical users.

### **1.2 Objectives**

The objectives of the project are:

* **Develop a Unified Disease Prediction Platform**: To create a web application that combines the prediction of multiple diseases (diabetes, heart disease, Parkinson's) in one platform.
* **Enable Early Diagnosis**: Provide a tool that facilitates early intervention by predicting the likelihood of diseases from user-provided health data.
* **Improve Accessibility**: The platform is designed to be easy to use by non-specialists and accessible through a web browser.
* **Streamline Medical Data Use**: Integrate machine learning models to analyze various health parameters and provide real-time predictions.
* **Cost-Effectiveness**: Reduce the need for expensive diagnostic procedures by offering a low-cost, easily accessible prediction tool.

### **1.3 Problem Formulation**

In traditional healthcare systems, diagnosing diseases like diabetes, heart disease, and Parkinson's disease often requires extensive tests, medical expertise, and significant time. These methods can be expensive and may not be accessible to everyone. Additionally, each disease often has its own specialized diagnostic tool, resulting in fragmented and inefficient diagnosis. The aim of this project is to consolidate the prediction of multiple diseases into a single system, making disease diagnosis faster, more accurate, and accessible to the general public. By using machine learning algorithms, the system is able to learn from vast datasets of patient information, offering predictions based on historical data and other relevant parameters.

### **1.4 Scope of the Project**

The scope of the project includes:

* **Multiple Disease Prediction**: The system will predict diseases such as diabetes, heart disease, and Parkinson's disease based on health parameters provided by the user.
* **Machine Learning Algorithms**: The system will use trained machine learning models to process and analyze data for disease prediction.
* **Web-Based Interface**: The application will be built on a web platform using Python and Streamlit, ensuring ease of access through any internet-enabled device.
* **Real-Time Predictions**: The system will provide immediate feedback on the likelihood of diseases based on input data, offering real-time predictions.
* **User Friendly**: The system will be easy to use, with a simple interface, and can be operated by non-specialists without technical knowledge.

### **1.5 Feasibility**

The feasibility of the project can be evaluated in terms of:

* **Technical Feasibility**: The technologies used, such as Python, Streamlit, and machine learning libraries like scikit-learn and pandas, are well-established and capable of handling the data processing and model computation required for this project. Machine learning models for disease prediction are also readily available and can be customized for this purpose.
* **Operational Feasibility**: Since the system is web-based, users can access it from any internet-enabled device. This enhances the system’s operational viability by eliminating geographical constraints.
* **Economic Feasibility**: The project is cost-effective since it uses open-source technologies (Python, Streamlit, scikit-learn, etc.), which do not require expensive licensing fees. Additionally, the web application eliminates the need for high-end hardware, making it accessible to a broad audience.
* **Legal and Ethical Feasibility**: Data security and privacy concerns must be addressed, especially when handling sensitive medical data. Implementing encryption and secure data storage mechanisms will ensure compliance with data protection regulations.

### **1.6 Requirements of the Project**

#### **1.6.1 Software Requirements**

The software needed for the development and deployment of the system includes:

* **Python**: A versatile programming language used for building the machine learning models and the web application.
* **Streamlit**: A web framework that simplifies the deployment of machine learning models by allowing Python code to be converted into a shareable web app.
* **Machine Learning Libraries**: Libraries like scikit-learn, pandas, and numpy will be used to build and train models, handle data, and perform mathematical operations.
  + **Scikit-learn**: Used for implementing machine learning algorithms such as classification and regression models.
  + **Pandas**: Used for data manipulation and analysis, particularly when handling large datasets.
  + **Numpy**: For numerical computations and array operations.
* **Model Files**: Pre-trained models for disease prediction will be used to make predictions about diabetes, heart disease, and Parkinson’s disease based on input data.
* **Database (Optional)**: If you plan to store user data for future analysis, a database system like mysql can be integrated to manage and store data securely.

#### **1.6.2 Hardware Requirements**

For smooth operation, the following hardware is recommended:

* **RAM**: A minimum of **16 GB** of RAM to handle the machine learning model computations efficiently. Machine learning models, especially when dealing with large datasets, require significant memory to process and load data quickly.
* **Processor**: A **Core i7** processor (or equivalent) is recommended for faster processing speeds, especially when handling real-time predictions.
* **Internet Connection**: A stable internet connection is essential for hosting the web application and accessing real-time health data inputs.
* **Storage**: Adequate storage space for hosting the application, models, and any datasets required for model training or updates.

### **2.REQUIREMENTS**

### **2.1 Requirements**

The **requirements** for the project are divided into two major categories: functional and non-functional. These requirements define the overall behavior and constraints of the system.

* **Functional Requirements**: These specify the actions the system should perform, including interactions with users, data handling, and predictions based on machine learning models.
* **Non-Functional Requirements**: These define the overall system qualities, such as performance, security, and usability. They do not dictate specific behaviors but establish standards for system operation.

Both categories of requirements must be addressed to ensure the success of the system.

### **2.1.1 Non-Functional Requirements**

1. **Prediction Confidence Intervals**:
   * The system must specify the **range of values** or confidence intervals associated with each predicted disease outcome. This will help users understand the **uncertainty** inherent in machine learning predictions.
   * For example, the system might predict a **70% likelihood of heart disease**, but it could also display that the **confidence interval is 60% to 80%**. This allows users to recognize that predictions have a degree of variability, encouraging them to consider professional medical advice for further confirmation.
2. **Reliability**:
   * **High Availability**: The system should be available for use at any time without downtime, ensuring that users can access the platform when needed.
   * **Consistency**: The system must provide consistent results given the same input data. Even though machine learning models have a slight inherent randomness, the results should not vary dramatically for the same input over time.
   * **Error Handling**: If the system encounters issues (e.g., faulty user input), it must handle errors gracefully by displaying appropriate error messages and guiding users to correct the inputs.
   * Since you are **not using a database**, the system must not fail due to any dependency on stored data or history. All operations are performed based on current inputs.
3. **Security**:
   * Despite not storing any data, the system must ensure that user inputs (especially medical data) are securely transmitted over the network. **HTTPS** should be used for secure data transmission to avoid unauthorized access during the session.
   * **Data Privacy**: The system should clarify to users that their data is not stored or shared with any third parties. This transparency will help in building trust, especially when handling sensitive medical data.
4. **Performance**:
   * The system should return predictions quickly, ideally within **a few seconds** after the user submits their data. Machine learning models should be optimized to minimize computation time while maintaining accuracy.
   * Since you're not using a database, the system will rely purely on **in-memory computations**, which may boost performance as there's no delay from database queries or data retrieval processes.
5. **Usability**:
   * **User-friendly Interface**: The web interface must be intuitive, guiding users seamlessly from input to results. Since non-specialists (patients) are the primary users, minimal technical jargon should be used.
   * **Mobile-Friendly**: The application should be responsive, working well across different devices, including smartphones and tablets, to ensure that users can access it from anywhere.
6. **Maintainability**:
   * The system should be easy to maintain and update. When new machine learning models are developed or current ones need improvement, they should be easily integrated without significant downtime.
   * Although the system is stateless (not using databases), its modular design should allow adding more prediction models in the future (e.g., for different diseases) with minimal changes to the core system.

### **2.1.2 Functional Requirements**

1. **User Input Interface**:
   * The system must provide a **simple and intuitive interface** for users (patients) to input various types of information such as:
     + **Symptoms**: Details related to any ongoing health issues (e.g., chest pain, dizziness, etc.).
     + **Medical History**: Information about existing conditions like diabetes, hypertension, etc.
     + **Demographic Data**: Age, gender, weight, height, etc., which are often relevant factors for disease prediction.
     + **Other Pertinent Factors**: Lifestyle habits (smoking, alcohol consumption), family history, and any other health-related data.
   * The interface should validate the user inputs (e.g., ensuring that age is a positive integer, or blood pressure values fall within a realistic range).
2. **Data Processing and Disease Prediction**:
   * After receiving the user’s inputs, the system will process this data using the **machine learning models** that have been pre-trained for predicting diseases such as diabetes, heart disease, and Parkinson’s.
   * The system will run these inputs through the appropriate model(s) and generate disease predictions.
   * **Clear Output Display**: The system must display the results in a user-friendly format. The predictions should indicate:
     + The likelihood (probability score) of each predicted disease.
     + Suggested next steps (e.g., "Consult a medical professional for further testing").
     + A message clarifying that the prediction is not a medical diagnosis but an aid for understanding potential risks.
3. **Prediction Results Summary**:
   * The results must show which disease(s) the user is most likely to have and include a summary of the user's input health data alongside the results.
   * **Confidence Scores**: Along with the predicted diseases, the system should show confidence intervals or probability scores (e.g., "72% chance of diabetes"). This gives users more context regarding the certainty of the prediction.

**3.ANALYSIS**

**3.1 Use case diagram**

Use case diagrams model behavior within a system and helps the developers understand of what the user require. Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can’t do. Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

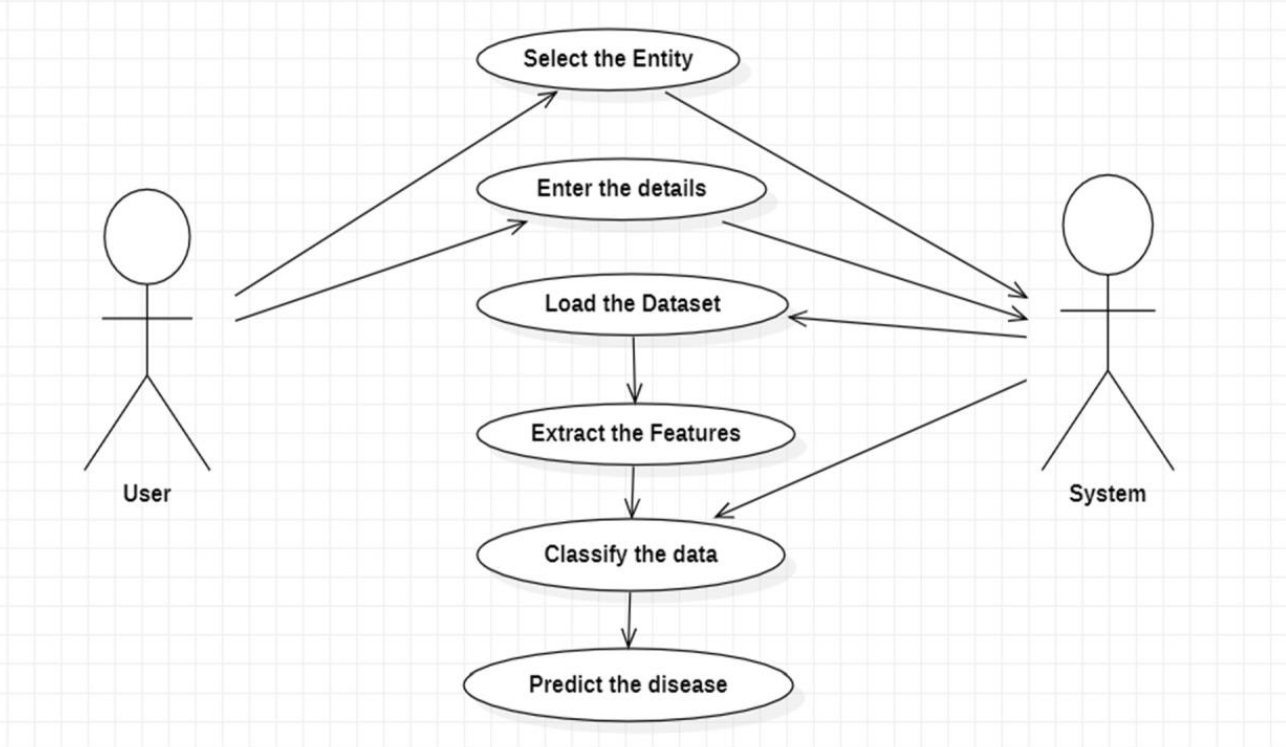


fig 3.1.1 use diagram

Above figure 3.1.1 use case diagram consists of two actors named as user and system. User can perform actions like select the entity and enter the details. System perform actions select the entity means select the disease and enter the patient details then load the data set and classify the data finally predict the disease.

**3.2 Class diagram**

A class diagram gives an overview of a system by showing its classes and the relationships among them. Class diagrams are static they display what interacts but not what happens when they do interact. The class chart delineates the attributes and operations of a class moreover the goals constrained on the structure. The class frameworks are extensively used as a piece of the showing of article arranged structures in light of the way that they are the primary uml diagrams which can be mapped direct with thing orchestrated vernaculars. The class graph shows a collection of classes, interfaces, affiliations, joint endeavors and confinements. It is generally called an assistant layout.

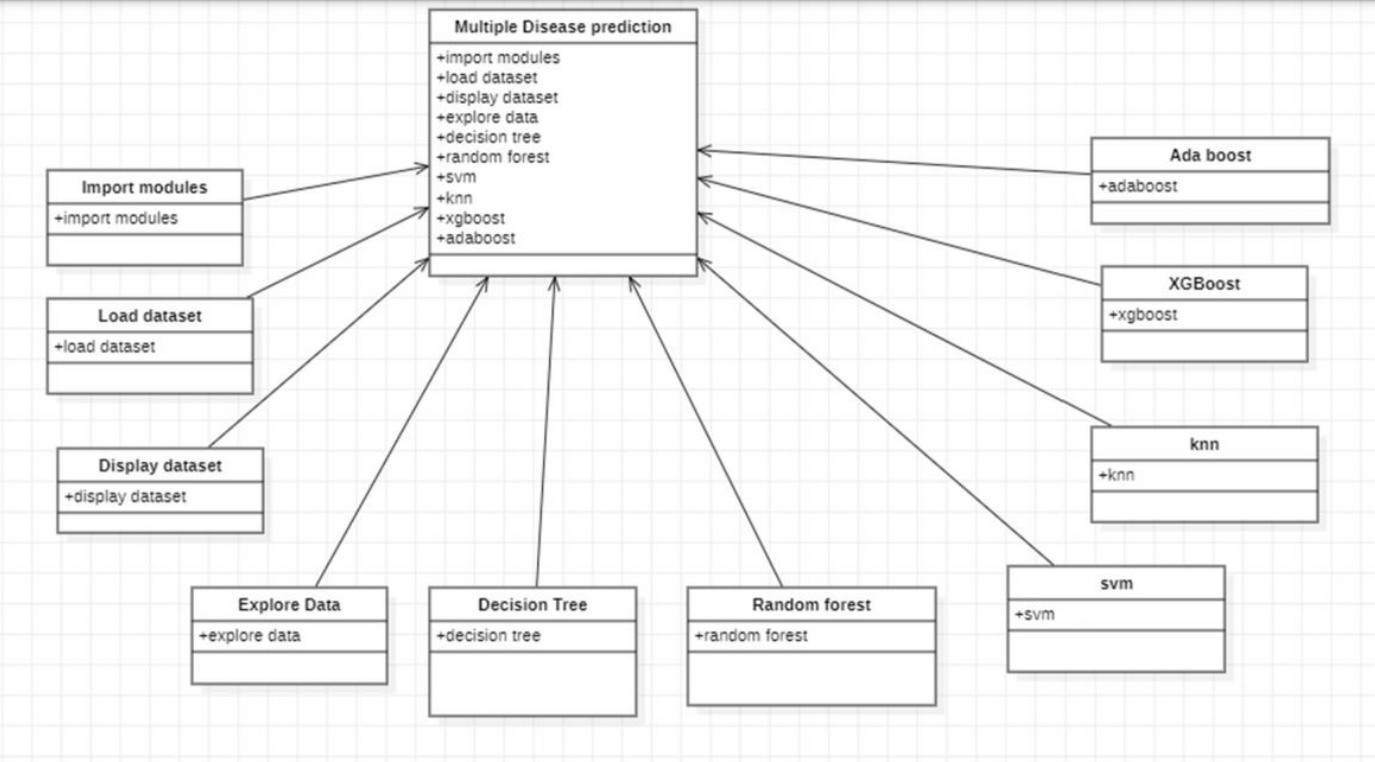


Fig 3.2.1 class diagram

From the fig 3.2.1 class diagram consists of multiple disease prediction,import data modules,load dataset,display dataset, explore data set, decision tree, random forest,svm,knn,xgboost,ada boost can perform the classes of class diagram. The multiple disease prediction can load the datasets and find diseases by applying algorithms like decision tree,random forest ,svm,knn,xgboost and ada boost.

**3.3 Sequence diagram**

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios. One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams.

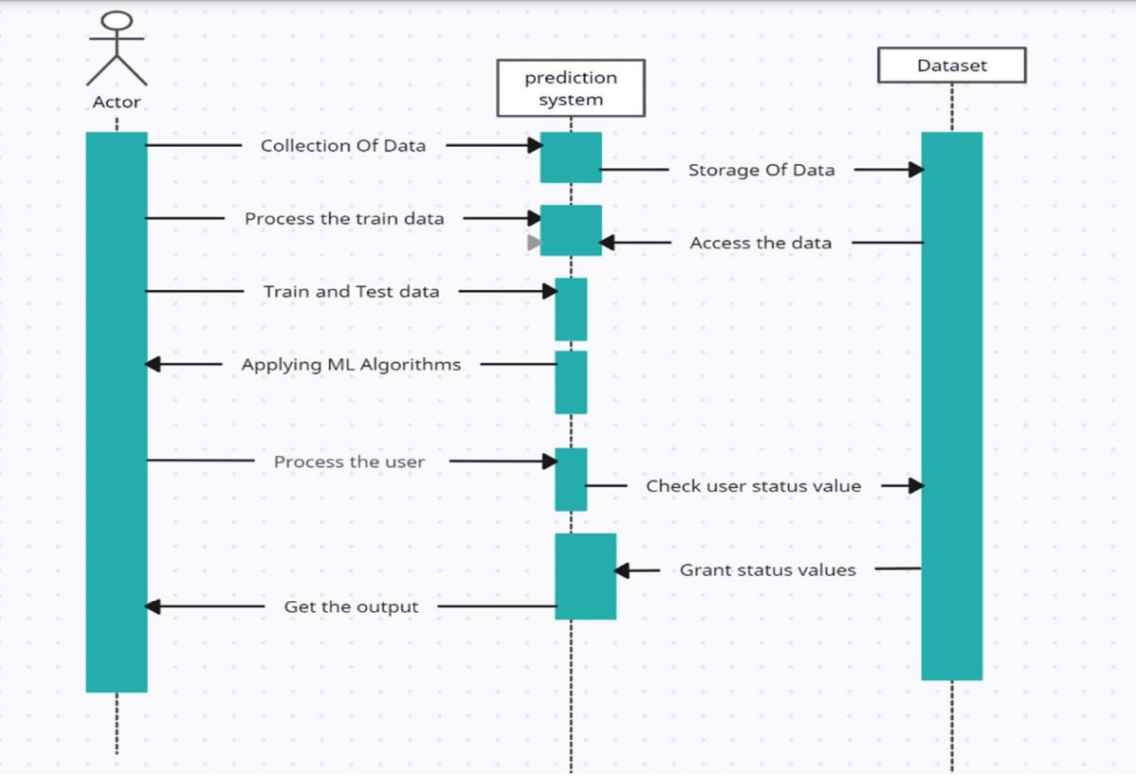


Fig 3.3.1 sequence diagram

from the fig:3.3.1 sequence diagram the prediction system can collect the data from actor and store the data in dataset.prediction system processes the train data and access the data from dataset then prediction system use the train and test data and apply ml algorithms and check user status value and grand status values then get the output.

**3.4 Deployment diagram**

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

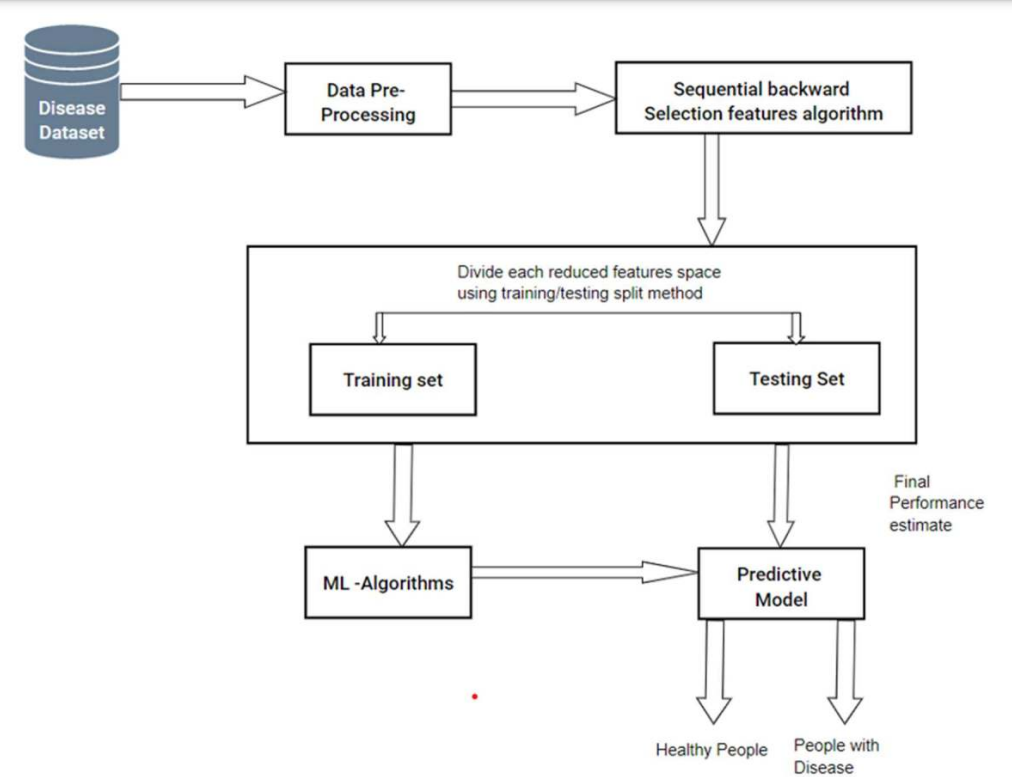
****

Fig.3.4.1 deployment diagram

A deployment diagram for multiple disease prediction includes components such as disease dataset,data preprocessing, , ml algorithms, predictive model . The user interface collects input data from disease dataset and processes using ml algorithms and then predict the disease using predict model.

**3.5 Component diagram**

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. That resides within the node. It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces. This diagram is also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements. System administrators can use component diagrams to plan ahead, using the view of the logical software components and their relationship on system.

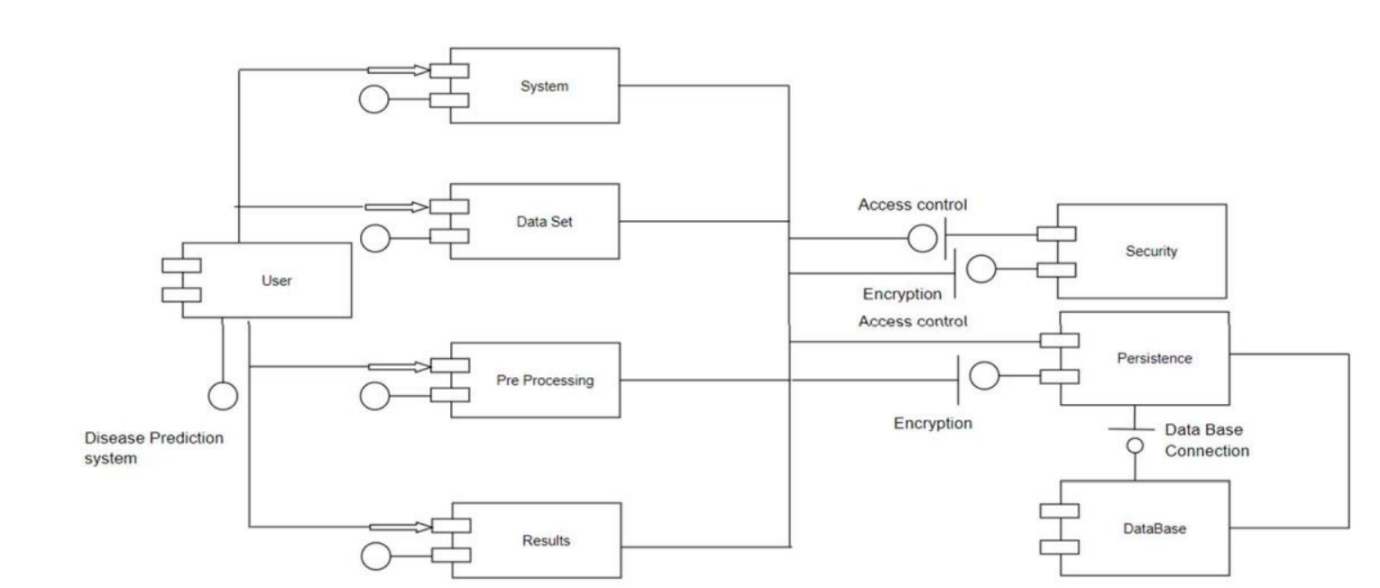


Fig.3.5.1 component diagram

from the fig:3.5.1 component diagram has components like user,system,data set,pre processing,results,security, persistence and data base these are tha components of multiple disease prediction system.

**4. Implementation**

Predicting multi diseases simultaneously is a complex task that requires a comprehensive methodology. The methodology for the Multi Disease Prediction project can be summarized as follows:

**1. Data Collection**: Data is collected from various sources such as electronic health records (ehrs), medical literature, public health databases. The data is obtained specifically for diabetes, heart disease and Parkinson's disease.

**2. Data preprocessing**: It involves two main tasks: data cleaning and transformation, which are essential for preparing the raw data into a suitable format for machine learning algorithms

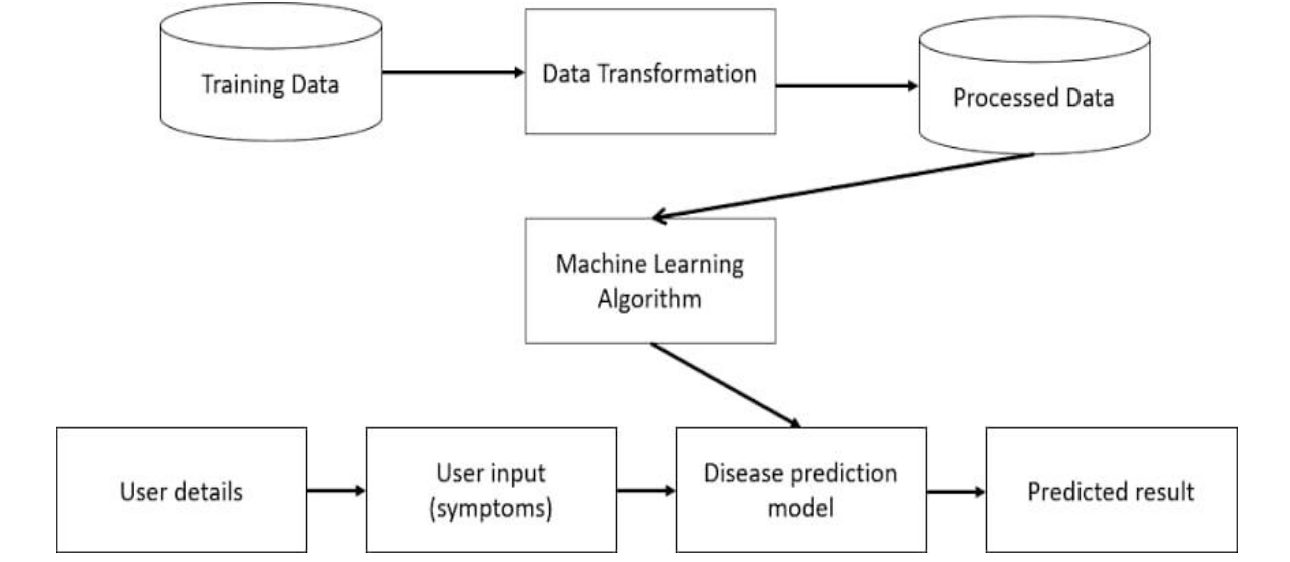
 **3. Model Selection:** Model selection is the initial phase of model training, where the most suitable machine learning algorithms are chosen for each disease. These selected algorithms will then undergo training using the preprocessed data and subsequently be evaluated based on performance metrics such as accuracy, precision to enhance the overall model performance.

Fig.4.1 Methodology

**4. Data Splitting:** The preprocessed data will be split into training and testing sets to evaluate the performance of the machine learning models. The models will be trained on the training set, and their performance will be assessed on the testing set.

**5. Deployment and Integration:** It involve deploying the trained models accompanied by its cloud deployment functionalities to develop an interactive web application. This application provides offering with options for predicting various diseases, prompting users to input required parameters upon selection, thus enhancing accessibility and usability.

**Code:**

Importing the Dependencies

from google.colab import drive  
drive.mount('/content/drive')

Mounted at /content/drive

import numpy as np  
import pandas as pd  
from sklearn.preprocessing import StandardScaler  
from sklearn.model\_selection import train\_test\_split  
from sklearn import svm  
from sklearn.metrics import accuracy\_score

Data Collection and Analysis

PIMA Diabetes Dataset

# loading the diabetes dataset to a pandas DataFrame  
diabetes\_dataset = pd.read\_csv('/content/diabetes.csv')

pd.read\_csv?

# printing the first 5 rows of the dataset  
diabetes\_dataset.head()# number of rows and Columns in this dataset  
diabetes\_dataset.shape

(768, 9)

# getting the statistical measures of the data  
diabetes\_dataset.describe()

diabetes\_dataset['Outcome'].value\_counts()

Outcome  
0 500  
1 268  
Name: count, dtype: int64

0 --> Non-Diabetic

1 --> Diabetic

diabetes\_dataset.groupby('Outcome').mean()

# separating the data and labels  
X = diabetes\_dataset.drop(columns = 'Outcome', axis=1)  
Y = diabetes\_dataset['Outcome']

print(X)

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \  
0 6 148 72 35 0 33.6   
1 1 85 66 29 0 26.6   
2 8 183 64 0 0 23.3   
3 1 89 66 23 94 28.1   
4 0 137 40 35 168 43.1   
.. ... ... ... ... ... ...   
763 10 101 76 48 180 32.9   
764 2 122 70 27 0 36.8   
765 5 121 72 23 112 26.2   
766 1 126 60 0 0 30.1   
767 1 93 70 31 0 30.4   
  
 DiabetesPedigreeFunction Age   
0 0.627 50   
1 0.351 31   
2 0.672 32   
3 0.167 21   
4 2.288 33   
.. ... ...   
763 0.171 63   
764 0.340 27   
765 0.245 30   
766 0.349 47   
767 0.315 23   
  
[768 rows x 8 columns]

print(Y)

0 1  
1 0  
2 1  
3 0  
4 1  
 ..  
763 0  
764 0  
765 0  
766 1  
767 0  
Name: Outcome, Length: 768, dtype: int64

Data Standardization

scaler = StandardScaler()

scaler.fit(X)

StandardScaler()

standardized\_data = scaler.transform(X)

print(standardized\_data)

[[ 0.63994726 0.84832379 0.14964075 ... 0.20401277 0.46849198  
 1.4259954 ]  
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078  
 -0.19067191]  
 [ 1.23388019 1.94372388 -0.26394125 ... -1.10325546 0.60439732  
 -0.10558415]  
 ...  
 [ 0.3429808 0.00330087 0.14964075 ... -0.73518964 -0.68519336  
 -0.27575966]  
 [-0.84488505 0.1597866 -0.47073225 ... -0.24020459 -0.37110101  
 1.17073215]  
 [-0.84488505 -0.8730192 0.04624525 ... -0.20212881 -0.47378505  
 -0.87137393]]

X = standardized\_data  
Y = diabetes\_dataset['Outcome']

print(X)  
print(Y)

[[ 0.63994726 0.84832379 0.14964075 ... 0.20401277 0.46849198  
 1.4259954 ]  
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078  
 -0.19067191]  
 [ 1.23388019 1.94372388 -0.26394125 ... -1.10325546 0.60439732  
 -0.10558415]  
 ...  
 [ 0.3429808 0.00330087 0.14964075 ... -0.73518964 -0.68519336  
 -0.27575966]  
 [-0.84488505 0.1597866 -0.47073225 ... -0.24020459 -0.37110101  
 1.17073215]  
 [-0.84488505 -0.8730192 0.04624525 ... -0.20212881 -0.47378505  
 -0.87137393]]  
0 1  
1 0  
2 1  
3 0  
4 1  
 ..  
763 0  
764 0  
765 0  
766 1  
767 0  
Name: Outcome, Length: 768, dtype: int64

Train Test Split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y, test\_size = 0.2, stratify=Y, random\_state=2)

print(X.shape, X\_train.shape, X\_test.shape)

(768, 8) (614, 8) (154, 8)

Training the Model

classifier = svm.SVC(kernel='linear')

#training the support vector Machine Classifier  
classifier.fit(X\_train, Y\_train)

SVC(kernel='linear')

Model Evaluation

Accuracy Score

# accuracy score on the training data  
X\_train\_prediction = classifier.predict(X\_train)  
training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print('Accuracy score of the training data : ', training\_data\_accuracy)

Accuracy score of the training data : 0.7866449511400652

# accuracy score on the test data  
X\_test\_prediction = classifier.predict(X\_test)  
test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print('Accuracy score of the test data : ', test\_data\_accuracy)

Accuracy score of the test data : 0.7727272727272727

Making a Predictive System

input\_data = (5,166,72,19,175,25.8,0.587,51)  
  
# changing the input\_data to numpy array  
input\_data\_as\_numpy\_array = np.asarray(input\_data)  
  
# reshape the array as we are predicting for one instance  
input\_data\_reshaped = input\_data\_as\_numpy\_array.reshape(1,-1)  
  
# standardize the input data  
std\_data = scaler.transform(input\_data\_reshaped)  
print(std\_data)  
  
prediction = classifier.predict(std\_data)  
print(prediction)  
  
if (prediction[0] == 0):  
 print('The person is not diabetic')  
else:  
 print('The person is diabetic')

[[ 0.3429808 1.41167241 0.14964075 -0.09637905 0.82661621 -0.78595734  
 0.34768723 1.51108316]]  
[1]  
The person is diabetic

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names  
 warnings.warn(

import streamlit as st

import numpy as np

from PIL import Image

import pickle

from streamlit\_option\_menu import option\_menu

# Load models

diabetes\_model = pickle.load(open("C:/Users/bipin/OneDrive/Desktop/Mini Project/saved models/diabetes\_model.sav", 'rb'))

heart\_disease\_model = pickle.load(open("C:/Users/bipin/OneDrive/Desktop/Mini Project/saved models/heart\_disease\_model.sav", 'rb'))

parkinsons\_model = pickle.load(open("C:/Users/bipin/OneDrive/Desktop/Mini Project/saved models/parkinsons\_model.sav", 'rb'))

Breast\_Cancer\_model = pickle.load(open("C:/Users/bipin/OneDrive/Desktop/Mini Project/saved models/breast\_cancer.sav", 'rb'))

# Custom CSS for background and font

st.markdown("""

<style>

.reportview-container {

background: #f0f2f6;

color: #31333f;

}

.sidebar .sidebar-content {

background: #3e3f47;

}

.stButton>button {

color: white;

background-color: #4caf50;

}

.stTextInput>div>div>input {

font-size: 16px;

color: #3a3a3a;

}

.stMarkdown h1, .stMarkdown h2 {

color: #2c3e50;

}

</style>

""", unsafe\_allow\_html=True)

# Sidebar for navigation

with st.sidebar:

selected = option\_menu('Multiple Disease Prediction System',

['Diabetes Prediction', 'Heart Disease Prediction', 'Parkinsons Prediction', 'Breast Cancer Prediction'],

icons=['activity', 'heart', 'person', 'medkit'],

default\_index=0)

# Diabetes Prediction Page

if selected == 'Diabetes Prediction':

st.title('Diabetes Prediction using Machine Learning 🩺')

img1 = Image.open('C:/Users/bipin/Downloads/di.png')

st.image(img1, caption="Stay Healthy, Stay Safe", use\_column\_width=True)

# Form layout with input handling

st.markdown("### Please enter your details below:")

col1, col2, col3 = st.columns(3)

with col1:

Pregnancies = st.number\_input('Number of Pregnancies (Range: 0-17) [Normal: 0-5, High: 6+]', min\_value=0, max\_value=17, value=1, step=1)

with col2:

Glucose = st.number\_input('Glucose Level (Range: 70-200) [Normal: 70-100, High: >140]', min\_value=70, max\_value=200, value=100)

with col3:

BloodPressure = st.number\_input('Blood Pressure (Range: 70-180) [Normal: 80-120, High: >130]', min\_value=70, max\_value=180, value=120)

with col1:

SkinThickness = st.number\_input('Skin Thickness (Range: 10-100) [Normal: 10-40, High: >50]', min\_value=10, max\_value=100, value=20)

with col2:

Insulin = st.number\_input('Insulin Level (Range: 0-900) [Normal: 16-166, High: >200]', min\_value=0, max\_value=900, value=100)

with col3:

BMI = st.number\_input('BMI Value (Range: 15.0-60.0) [Normal: 18.5-24.9, High: >30]', min\_value=15.0, max\_value=60.0, value=25.0)

with col1:

DiabetesPedigreeFunction = st.number\_input('Diabetes Pedigree Function (Range: 0.0-2.5) [Normal: 0.2-0.8]', min\_value=0.0, max\_value=2.5, value=0.5)

with col2:

Age = st.number\_input('Age (Range: 0-120) [Normal: 20-60, High: >60]', min\_value=0, max\_value=120, value=25)

# Prediction and precautions

diab\_diagnosis = ''

if st.button('Predict Diabetes'):

diab\_prediction = diabetes\_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age]])

if diab\_prediction[0] == 1:

diab\_diagnosis = 'The person is diabetic 🩸'

else:

diab\_diagnosis = 'The person is not diabetic 🟢'

st.success(diab\_diagnosis)

st.markdown("### Preventive Measures for Diabetes 🛡️")

st.write("""

- \*\*Healthy Diet:\*\* Choose low-sugar foods.

- \*\*Exercise Regularly:\*\* Aim for 30 minutes daily.

- \*\*Routine Check-ups:\*\* Regular health checkups are crucial.

""")

# Heart Disease Prediction Page

if selected == 'Heart Disease Prediction':

st.title('Heart Disease Prediction using Machine Learning ❤️')

img2 = Image.open('C:/Users/bipin/Downloads/si.png')

st.image(img2, use\_column\_width=True)

st.markdown("### Please enter the following details:")

col1, col2, col3 = st.columns(3)

with col1:

age = st.number\_input('Age (Range: 0-120) [Normal: 20-60, High: >60]', min\_value=0, max\_value=120, value=30)

with col2:

sex = st.selectbox('Sex', [1, 0], format\_func=lambda x: 'Male' if x == 1 else 'Female')

with col3:

cp = st.selectbox('Chest Pain Type', [0, 1, 2, 3], format\_func=lambda x: ['Asymptomatic', 'Typical Angina', 'Atypical Angina', 'Non-Anginal Pain'][x])

with col1:

trestbps = st.number\_input('Resting Blood Pressure (Range: 50-200) [Normal: 80-120, High: >130]', min\_value=50, max\_value=200, value=120)

with col2:

chol = st.number\_input('Serum Cholesterol (Range: 100-500) [Normal: 150-200, High: >240]', min\_value=100, max\_value=500, value=200)

with col3:

fbs = st.selectbox('Fasting Blood Sugar > 120 mg/dl', [1, 0], format\_func=lambda x: 'Yes' if x == 1 else 'No')

with col1:

restecg = st.selectbox('Resting ECG Results', [0, 1, 2])

with col2:

thalach = st.number\_input('Maximum Heart Rate Achieved (Range: 50-200) [Normal: 60-100]', min\_value=50, max\_value=200, value=150)

with col3:

exang = st.selectbox('Exercise Induced Angina', [0, 1], format\_func=lambda x: 'No' if x == 0 else 'Yes')

with col1:

oldpeak = st.number\_input('ST Depression Induced by Exercise (Range: 0.0-6.0) [Normal: 0.0-1.0]', min\_value=0.0, max\_value=6.0, value=1.0)

with col2:

slope = st.selectbox('Slope of the Peak Exercise ST Segment', [0, 1, 2])

with col3:

ca = st.number\_input('Number of Major Vessels Colored by Fluoroscopy (Range: 0-3) [Normal: 0-1]', min\_value=0, max\_value=3, value=0)

with col1:

thal = st.selectbox('Thalassemia', [0, 1, 2], format\_func=lambda x: ['Normal', 'Fixed Defect', 'Reversible Defect'][x])

# Heart Disease Prediction

heart\_diagnosis = ''

if st.button('Predict Heart Disease'):

input\_data = np.array([[age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]])

heart\_prediction = heart\_disease\_model.predict(input\_data)

if heart\_prediction[0] == 1:

heart\_diagnosis = 'The person has heart disease 💔'

else:

heart\_diagnosis = 'The person does not have heart disease 💖'

st.success(heart\_diagnosis)

st.markdown("### Preventive Measures for Heart Disease 💡")

st.write("""

- \*\*Balanced Diet:\*\* Reduce saturated fats.

- \*\*Physical Activity:\*\* 30 minutes daily.

- \*\*Quit Smoking:\*\* Improve your heart health instantly.

""")

# Add similar layouts for Parkinson's and Breast Cancer as needed...

# Parkinson's Prediction Page

# Parkinson's Prediction Page

if selected == "Parkinsons Prediction":

img4 = Image.open('C:/Users/bipin/Downloads/park.png')

# Page title with some styling

st.markdown("<h1 style='text-align: center; color: #4CAF50;'>Parkinson's Disease Prediction using ML</h1>", unsafe\_allow\_html=True)

st.image(img4, caption="Stay Healthy, Stay Safe", use\_column\_width=True)

st.markdown("<p style='text-align: center;'>Fill in the parameters below to predict the likelihood of Parkinson's disease.</p>", unsafe\_allow\_html=True)

# Add a description or instructions

st.markdown("<p style='text-align: center;'>The input features are based on voice measurements. Use the sliders to input values within recommended ranges.</p>", unsafe\_allow\_html=True)

# Dividing input fields into columns and using sliders for better UX with range limits

st.markdown("### Voice Measurements")

col1, col2, col3, col4, col5 = st.columns(5)

with col1:

fo = st.slider('MDVP:Fo(Hz)', min\_value=50.0, max\_value=300.0, value=120.0, step=1.0, help="Fundamental frequency of voice (range: 50-300 Hz)")

with col2:

fhi = st.slider('MDVP:Fhi(Hz)', min\_value=100.0, max\_value=400.0, value=200.0, step=1.0, help="Maximum vocal fundamental frequency (range: 100-400 Hz)")

with col3:

flo = st.slider('MDVP:Flo(Hz)', min\_value=50.0, max\_value=300.0, value=100.0, step=1.0, help="Minimum vocal fundamental frequency (range: 50-300 Hz)")

with col4:

Jitter\_percent = st.slider('MDVP:Jitter(%)', min\_value=0.0, max\_value=1.0, value=0.01, step=0.001, help="Variation in fundamental frequency (range: 0-1%)")

with col5:

Jitter\_Abs = st.slider('MDVP:Jitter(Abs)', min\_value=0.0, max\_value=0.01, value=0.002, step=0.0001, help="Absolute Jitter (range: 0-0.01)")

with col1:

RAP = st.slider('MDVP:RAP', min\_value=0.0, max\_value=0.01, value=0.0025, step=0.0001, help="Relative amplitude perturbation (range: 0-0.01)")

with col2:

PPQ = st.slider('MDVP:PPQ', min\_value=0.0, max\_value=0.01, value=0.0025, step=0.0001, help="Five-point period perturbation quotient (range: 0-0.01)")

with col3:

DDP = st.slider('Jitter:DDP', min\_value=0.0, max\_value=0.03, value=0.0075, step=0.001, help="Difference of differences of periods (range: 0-0.03)")

with col4:

Shimmer = st.slider('MDVP:Shimmer', min\_value=0.0, max\_value=0.1, value=0.01, step=0.001, help="Variation in amplitude (range: 0-0.1)")

with col5:

Shimmer\_dB = st.slider('MDVP:Shimmer(dB)', min\_value=0.0, max\_value=1.0, value=0.1, step=0.01, help="Shimmer in decibels (range: 0-1 dB)")

with col1:

APQ3 = st.slider('Shimmer:APQ3', min\_value=0.0, max\_value=0.05, value=0.015, step=0.001, help="Three-point amplitude perturbation quotient (range: 0-0.05)")

with col2:

APQ5 = st.slider('Shimmer:APQ5', min\_value=0.0, max\_value=0.05, value=0.0175, step=0.001, help="Five-point amplitude perturbation quotient (range: 0-0.05)")

with col3:

APQ = st.slider('MDVP:APQ', min\_value=0.0, max\_value=0.05, value=0.02, step=0.001, help="Amplitude perturbation quotient (range: 0-0.05)")

with col4:

DDA = st.slider('Shimmer:DDA', min\_value=0.0, max\_value=0.15, value=0.05, step=0.005, help="Difference of differences of amplitudes (range: 0-0.15)")

with col5:

NHR = st.slider('NHR', min\_value=0.0, max\_value=1.0, value=0.01, step=0.01, help="Noise-to-Harmonics ratio (range: 0-1)")

with col1:

HNR = st.slider('HNR', min\_value=0.0, max\_value=50.0, value=20.0, step=1.0, help="Harmonics-to-Noise ratio (range: 0-50)")

with col2:

RPDE = st.slider('RPDE', min\_value=0.0, max\_value=1.0, value=0.5, step=0.01, help="Recurrence period density entropy (range: 0-1)")

with col3:

DFA = st.slider('DFA', min\_value=0.0, max\_value=1.0, value=0.5, step=0.01, help="Detrended fluctuation analysis (range: 0-1)")

with col4:

spread1 = st.slider('Spread1', min\_value=-10.0, max\_value=5.0, value=-5.0, step=0.1, help="Nonlinear measure of fundamental frequency variation (range: -10 to 5)")

with col5:

spread2 = st.slider('Spread2', min\_value=0.0, max\_value=0.6, value=0.2, step=0.01, help="Nonlinear measure of fundamental frequency variation (range: 0-0.6)")

with col1:

D2 = st.slider('D2', min\_value=0.0, max\_value=4.0, value=2.0, step=0.1, help="Nonlinear dynamical complexity measure (range: 0-4)")

with col2:

PPE = st.slider('PPE', min\_value=0.0, max\_value=0.5, value=0.1, step=0.01, help="Pitch period entropy (range: 0-0.5)")

# Code for Prediction

parkinsons\_diagnosis = ''

# Creating a button for Prediction

if st.button("Parkinson's Test Result"):

try:

# Prediction model takes floats, converting text inputs to float

parkinsons\_prediction = parkinsons\_model.predict([[fo, fhi, flo, Jitter\_percent, Jitter\_Abs, RAP, PPQ, DDP,

Shimmer, Shimmer\_dB, APQ3, APQ5, APQ, DDA, NHR, HNR, RPDE,

DFA, spread1, spread2, D2, PPE]])

if parkinsons\_prediction[0] == 1:

parkinsons\_diagnosis = "The person has Parkinson's disease"

else:

parkinsons\_diagnosis = "The person does not have Parkinson's disease"

st.success(parkinsons\_diagnosis)

except ValueError:

st.error("Please enter valid numeric values.")

if selected == "Breast Cancer Prediction":

# Title page with a catchy title and subtitle

st.markdown("<h1 style='text-align: center; color: #4CAF50;'>Breast Cancer Detection Model</h1>", unsafe\_allow\_html=True)

st.markdown("<p style='text-align: center;'>Predict the likelihood of breast cancer with input features</p>", unsafe\_allow\_html=True)

# Add a description or instructions

st.markdown("<p style='text-align: center;'>Please fill in the details below for an accurate prediction:</p>", unsafe\_allow\_html=True)

# Using an expander to hide detailed info if needed

with st.expander("Input Features"):

# Columns for input fields grouped in a structured format

st.markdown("### Mean Features")

col1, col2, col3 = st.columns(3)

with col1:

mean\_radius = st.number\_input('Mean Radius', min\_value=6.0, max\_value=30.0, value=14.0, format="%.3f", help="Range: 6 - 30")

with col2:

mean\_texture = st.number\_input('Mean Texture', min\_value=9.0, max\_value=40.0, value=20.0, format="%.3f", help="Range: 9 - 40")

with col3:

mean\_perimeter = st.number\_input('Mean Perimeter', min\_value=40.0, max\_value=190.0, value=90.0, format="%.3f", help="Range: 40 - 190")

with col1:

mean\_area = st.number\_input('Mean Area', min\_value=100.0, max\_value=2500.0, value=700.0, format="%.3f", help="Range: 100 - 2500")

with col2:

mean\_smoothness = st.number\_input('Mean Smoothness', min\_value=0.05, max\_value=0.2, value=0.1, format="%.3f", help="Range: 0.05 - 0.2")

with col3:

mean\_compactness = st.number\_input('Mean Compactness', min\_value=0.01, max\_value=0.35, value=0.15, format="%.3f", help="Range: 0.01 - 0.35")

with col1:

mean\_concavity = st.number\_input('Mean Concavity', min\_value=0.0, max\_value=0.45, value=0.20, format="%.3f", help="Range: 0 - 0.45")

with col2:

mean\_concave\_points = st.number\_input('Mean Concave Points', min\_value=0.0, max\_value=0.25, value=0.1, format="%.3f", help="Range: 0 - 0.25")

with col3:

mean\_symmetry = st.number\_input('Mean Symmetry', min\_value=0.1, max\_value=0.5, value=0.2, format="%.3f", help="Range: 0.1 - 0.5")

with col1:

mean\_fractal\_dimension = st.number\_input('Mean Fractal Dimension', min\_value=0.05, max\_value=0.1, value=0.06, format="%.3f", help="Range: 0.05 - 0.1")

st.markdown("### Error Features")

col1, col2, col3 = st.columns(3)

with col1:

radius\_error = st.number\_input('Radius Error', min\_value=0.1, max\_value=3.0, value=0.5, format="%.3f", help="Range: 0.1 - 3.0")

with col2:

texture\_error = st.number\_input('Texture Error', min\_value=0.1, max\_value=5.0, value=1.0, format="%.3f", help="Range: 0.1 - 5.0")

with col3:

perimeter\_error = st.number\_input('Perimeter Error', min\_value=1.0, max\_value=25.0, value=5.0, format="%.3f", help="Range: 1 - 25")

with col1:

area\_error = st.number\_input('Area Error', min\_value=5.0, max\_value=550.0, value=40.0, format="%.3f", help="Range: 5 - 550")

with col2:

smoothness\_error = st.number\_input('Smoothness Error', min\_value=0.002, max\_value=0.04, value=0.01, format="%.3f", help="Range: 0.002 - 0.04")

with col3:

compactness\_error = st.number\_input('Compactness Error', min\_value=0.002, max\_value=0.14, value=0.03, format="%.3f", help="Range: 0.002 - 0.14")

with col1:

concavity\_error = st.number\_input('Concavity Error', min\_value=0.0, max\_value=0.4, value=0.05, format="%.3f", help="Range: 0 - 0.4")

with col2:

concave\_points\_error = st.number\_input('Concave Points Error', min\_value=0.0, max\_value=0.2, value=0.02, format="%.3f", help="Range: 0 - 0.2")

with col3:

symmetry\_error = st.number\_input('Symmetry Error', min\_value=0.01, max\_value=0.08, value=0.03, format="%.3f", help="Range: 0.01 - 0.08")

with col1:

fractal\_dimension\_error = st.number\_input('Fractal Dimension Error', min\_value=0.0001, max\_value=0.03, value=0.004, format="%.3f", help="Range: 0.0001 - 0.03")

st.markdown("### Worst Features")

col1, col2, col3 = st.columns(3)

with col1:

worst\_radius = st.number\_input('Worst Radius', min\_value=7.0, max\_value=40.0, value=16.0, format="%.3f", help="Range: 7 - 40")

with col2:

worst\_texture = st.number\_input('Worst Texture', min\_value=12.0, max\_value=50.0, value=25.0, format="%.3f", help="Range: 12 - 50")

with col3:

worst\_perimeter = st.number\_input('Worst Perimeter', min\_value=50.0, max\_value=250.0, value=120.0, format="%.3f", help="Range: 50 - 250")

with col1:

worst\_area = st.number\_input('Worst Area', min\_value=100.0, max\_value=3000.0, value=880.0, format="%.3f", help="Range: 100 - 3000")

with col2:

worst\_smoothness = st.number\_input('Worst Smoothness', min\_value=0.05, max\_value=0.25, value=0.15, format="%.3f", help="Range: 0.05 - 0.25")

with col3:

worst\_compactness = st.number\_input('Worst Compactness', min\_value=0.02, max\_value=1.1, value=0.35, format="%.3f", help="Range: 0.02 - 1.1")

with col1:

worst\_concavity = st.number\_input('Worst Concavity', min\_value=0.0, max\_value=1.25, value=0.45, format="%.3f", help="Range: 0 - 1.25")

with col2:

worst\_concave\_points = st.number\_input('Worst Concave Points', min\_value=0.0, max\_value=0.3, value=0.2, format="%.3f", help="Range: 0 - 0.3")

with col3:

worst\_symmetry = st.number\_input('Worst Symmetry', min\_value=0.1, max\_value=0.7, value=0.3, format="%.3f", help="Range: 0.1 - 0.7")

with col1:

worst\_fractal\_dimension = st.number\_input('Worst Fractal Dimension', min\_value=0.05, max\_value=0.15, value=0.09, format="%.3f", help="Range: 0.05 - 0.15")

# Code for prediction

breast\_cancer\_diagnosis = ''

# Creating a button for prediction

if st.button('Get Breast Cancer Test Result'):

# Convert all inputs to float as necessary for model prediction

try:

input\_features = [

float(mean\_radius), float(mean\_texture), float(mean\_perimeter), float(mean\_area),

float(mean\_smoothness), float(mean\_compactness), float(mean\_concavity),

float(mean\_concave\_points), float(mean\_symmetry), float(mean\_fractal\_dimension),

float(radius\_error), float(texture\_error), float(perimeter\_error), float(area\_error),

float(smoothness\_error), float(compactness\_error), float(concavity\_error),

float(concave\_points\_error), float(symmetry\_error), float(fractal\_dimension\_error),

float(worst\_radius), float(worst\_texture), float(worst\_perimeter), float(worst\_area),

float(worst\_smoothness), float(worst\_compactness), float(worst\_concavity),

float(worst\_concave\_points), float(worst\_symmetry), float(worst\_fractal\_dimension)

]

breast\_cancer\_prediction = Breast\_Cancer\_model.predict([input\_features])

if breast\_cancer\_prediction[0] == 0:

breast\_cancer\_diagnosis = 'Breast Cancer is Malignant'

else:

breast\_cancer\_diagnosis = 'Breast Cancer is Benign'

st.success(breast\_cancer\_diagnosis)

except ValueError:

st.error("Please enter valid numeric values for all inputs.")

# Breast cancer prevention tips

with st.expander("Breast Cancer Prevention Tips"):

st.markdown("""

- \*\*Maintain a healthy weight\*\*: Obesity can increase the risk of breast cancer.

- \*\*Exercise regularly\*\*: Physical activity helps regulate hormones that may influence breast cancer risk.

- \*\*Limit alcohol consumption\*\*: Drinking alcohol is a significant risk factor.

- \*\*Stay aware of family history\*\*: Early detection can be lifesaving, especially if there is a family history of breast cancer.

- \*\*Quit smoking\*\*: Smoking is linked to a higher risk of breast cancer, especially in younger women.

- \*\*Breastfeed if possible\*\*: Breastfeeding for at least several months may lower breast cancer risk.

- \*\*Regular self-exams and screenings\*\*: Early detection through mammograms and self-exams is critical.

""")

**Modules**

The document describes three specific disease prediction modules:

1. **Parkinson's Disease Prediction**
   * Uses data on vocal characteristics to predict Parkinson's disease.
   * Employs various machine learning algorithms like KNN, xgboost, SVM, Random Forest etc.

Fig.4.2 Model Comparison For Parkinson’s

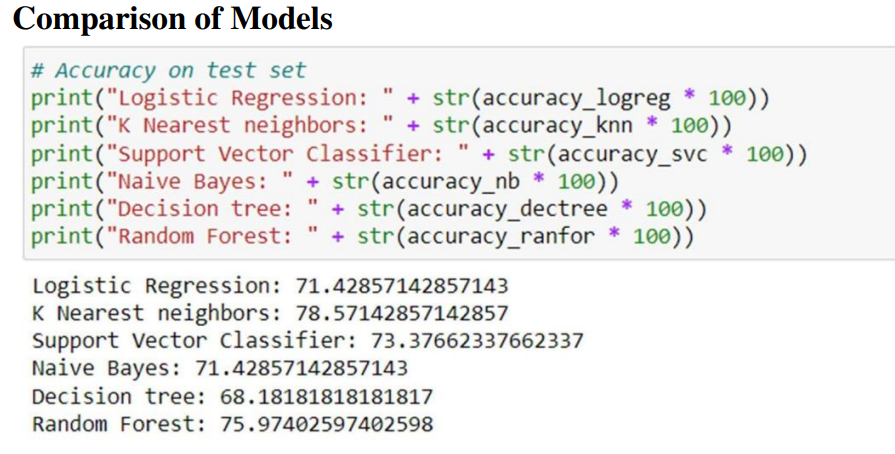
1. **Diabetes Disease Prediction**.

Fig.4.3 Model Comparison For Diabetes

Aims to predict diabetes early using supervised machine learning methods

Analyzes data on factors like pregnancies, blood pressure, and body mass index

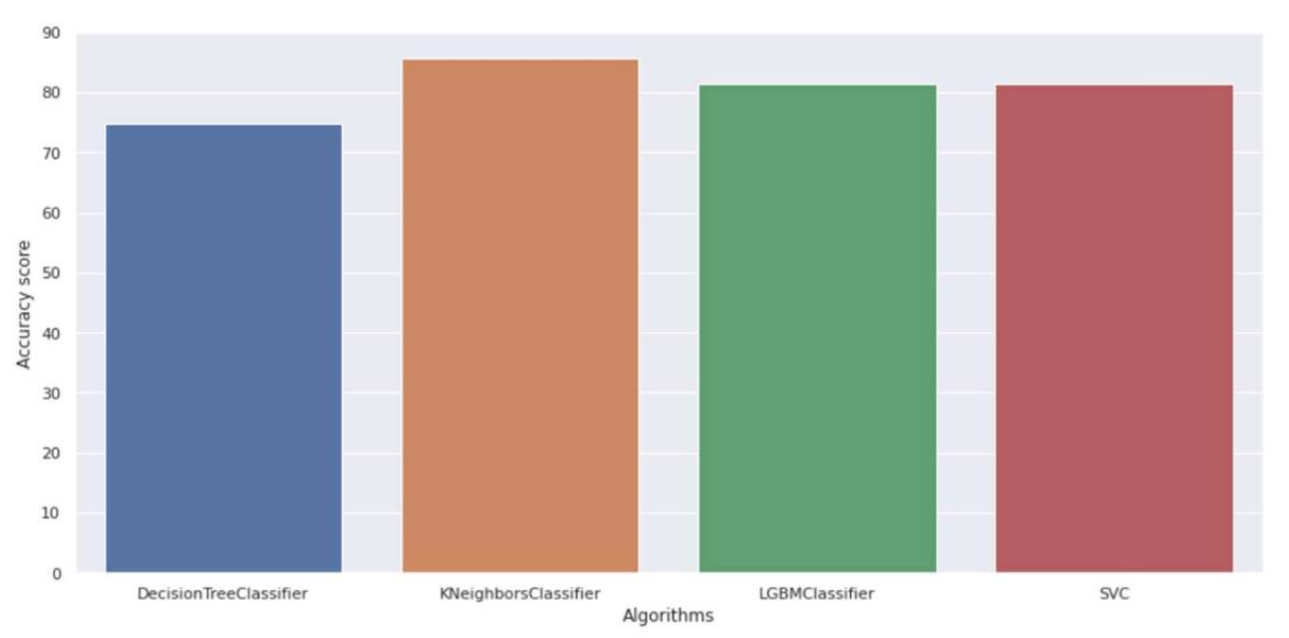
1. **Heart Disease Prediction**
   * Analyzes data on factors like age, blood pressure, and cholesterol levels to predict heart disease.
   * Employs various machine learning algorithms similar to Parkinson's disease prediction.

Fig.4.4 Model Comparison For Heart

**Technologies Used**

* **Python:** A high-level, general-purpose programming language popular for machine learning due to its readability and extensive libraries.
* **Streamlit:** An open-source Python framework for creating user-friendly web applications for data science and machine learning projects.
* **Jupyter Notebook:** An open-source web application for creating and sharing documents that combine live code, visualizations, and text.

**Algorithms**

The document mentions several machine learning algorithms used for disease prediction, including:

* **Decision Tree Classifiers:** Create tree-like models for classification based on a series of yes/no questions.
* **Gradient Boosting:** Builds a model by combining multiple weak decision trees to improve accuracy.
* **K-Nearest Neighbors (KNN):** Classifies data points based on the similarity to their nearest neighbors in the training data.
* **Logistic Regression Classifiers:** Used for binary classification problems (presence or absence of disease).
* **Naive Bayes:** A classification algorithm based on the assumption of independence between features.
* **Random Forest:** Creates an ensemble of decision trees for improved performance and reduced overfitting.
* **SVM (Support Vector Machine):** Finds a hyperplane that best separates data points of different classes.

**5. Testing Methodology**

#### **1. Unit Testing**

**Focus:** Each individual module of the disease prediction system is tested in isolation to ensure that it functions correctly.

* **Purpose:** Verify that each algorithm (e.g., decision trees, neural networks) behaves as expected.
* **Actions:** Test individual components such as data preprocessing functions, feature extraction methods, and model training scripts.
* **Expected Results:** Each module should return the expected output for given inputs, with all error-handling paths also validated.

#### **2. Integration Testing**

**Focus:** After unit testing, the integration of various modules is assessed to ensure they work together seamlessly.

* **Types:**
  + **Top-Down Integration:** Begin testing from the main prediction module, integrating lower-level modules like data input and preprocessing.
  + **Bottom-Up Integration:** Start with data preprocessing modules, integrating them into the model training and prediction modules as they are tested.
* **Purpose:** Verify that the entire system works as intended when different modules interact.
* **Expected Results:** Successful data flow and accurate predictions across integrated modules.

#### **3. User Acceptance Testing (UAT)**

**Focus:** Ensuring the system meets user needs and expectations before final deployment.

* **Purpose:** Gather feedback from potential end-users (e.g., healthcare professionals) during development.
* **Actions:** Conduct workshops to demonstrate system functionality, making adjustments based on user feedback.
* **Expected Results:** Users find the system intuitive and capable of meeting their disease prediction requirements.

#### **4. Output Testing**

**Focus:** Validating the output generated by the prediction system.

* **Purpose:** Ensure that the outputs are accurate and presented in a user-friendly format.
* **Actions:** Verify that predicted disease results match expected outputs based on predefined conditions and user inputs.
* **Expected Results:** The output format is correct, with clear communication of predicted diseases to the user.

#### **5. Validation Testing**

**Focus:** Ensuring the accuracy and integrity of input data.

* **Input Fields:**
  + **Text Fields:** Symptoms entered by users should adhere to character limits and formats (e.g., alphabetical for symptoms).
  + **Numeric Fields:** Any quantitative inputs (e.g., age, test scores) must only contain numeric values.
* **Purpose:** Confirm that invalid inputs are properly handled and that users receive informative error messages.
* **Expected Results:** System only accepts valid inputs and provides feedback for incorrect entries.

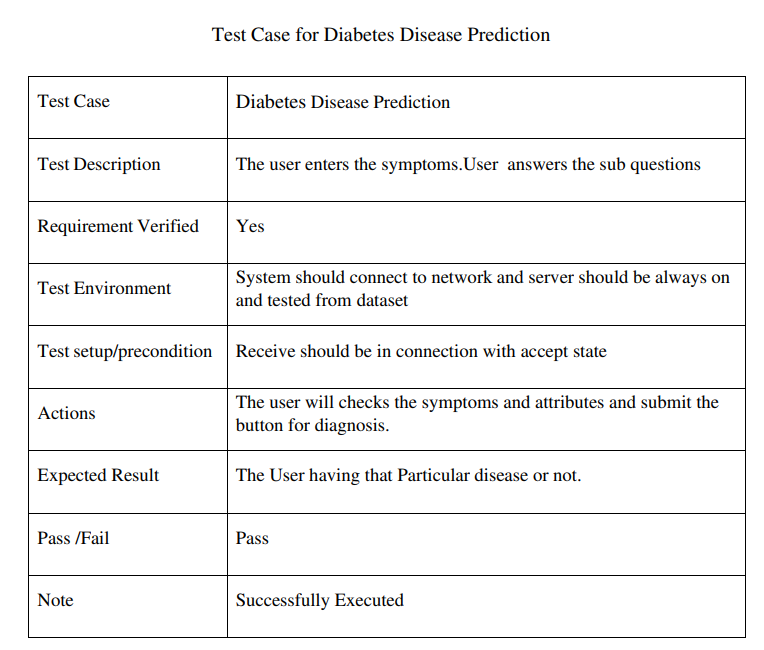
#### Test Data Preparation

**Focus:** Creating diverse datasets for comprehensive testing.

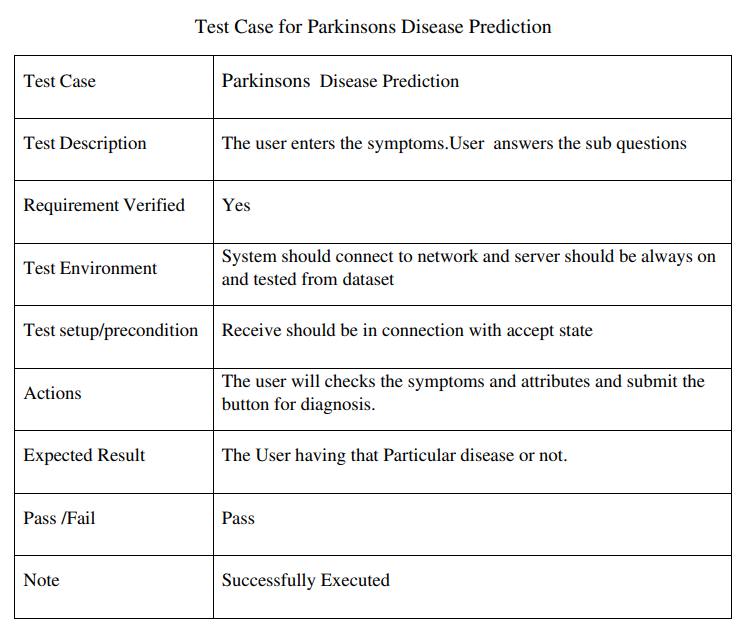
* **Live Test Data:**
  + **Source:** Collect data from real patient records (with appropriate permissions).
  + **Purpose:** To test how the model performs with realistic data inputs.
* **Artificial Test Data:**
  + **Source:** Generate synthetic datasets to cover a range of symptoms and demographic profiles.
  + **Purpose:** To ensure the model is robust and can handle all possible scenarios and edge cases.

#### Example Test Cases

1. **Test Case for Heart Disease Prediction:**
   * **Description:** User inputs symptoms related to brain diseases.
   * **Requirements Verified:** Yes.
   * **Test Environment:** Network connection required.
   * **Setup:** System must be in an active state.
   * **Actions:** User checks symptoms, submits for diagnosis.
   * **Expected Result:** Indicates if the user has a specific Heart disease.
   * **Pass/Fail:** Pass (Successfully executed).
2. **Test Case for Diabetes Disease Prediction:**
   * **Description:** User inputs symptoms related to diabetes.
   * **Requirements Verified:** Yes.
   * **Test Environment:** Network connection required.
   * **Setup:** System must be in an active state.
   * **Actions:** User checks symptoms, submits for diagnosis.
   * **Expected Result:** Indicates if the user has diabetes.
   * **Pass/Fail:** Pass (Successfully executed).



1. **Test Case for Parkinson's Disease Prediction:**
   * **Description:** User inputs symptoms related to Parkinson's disease.
   * **Requirements Verified:** Yes.
   * **Test Environment:** Network connection required.
   * **Setup:** System must be in an active state.
   * **Actions:** User checks symptoms, submits for diagnosis.
   * **Expected Result:** Indicates if the user has Parkinson's disease.
   * **Pass/Fail:** Pass (Successfully executed).



**6. Conclusions And Future Work**

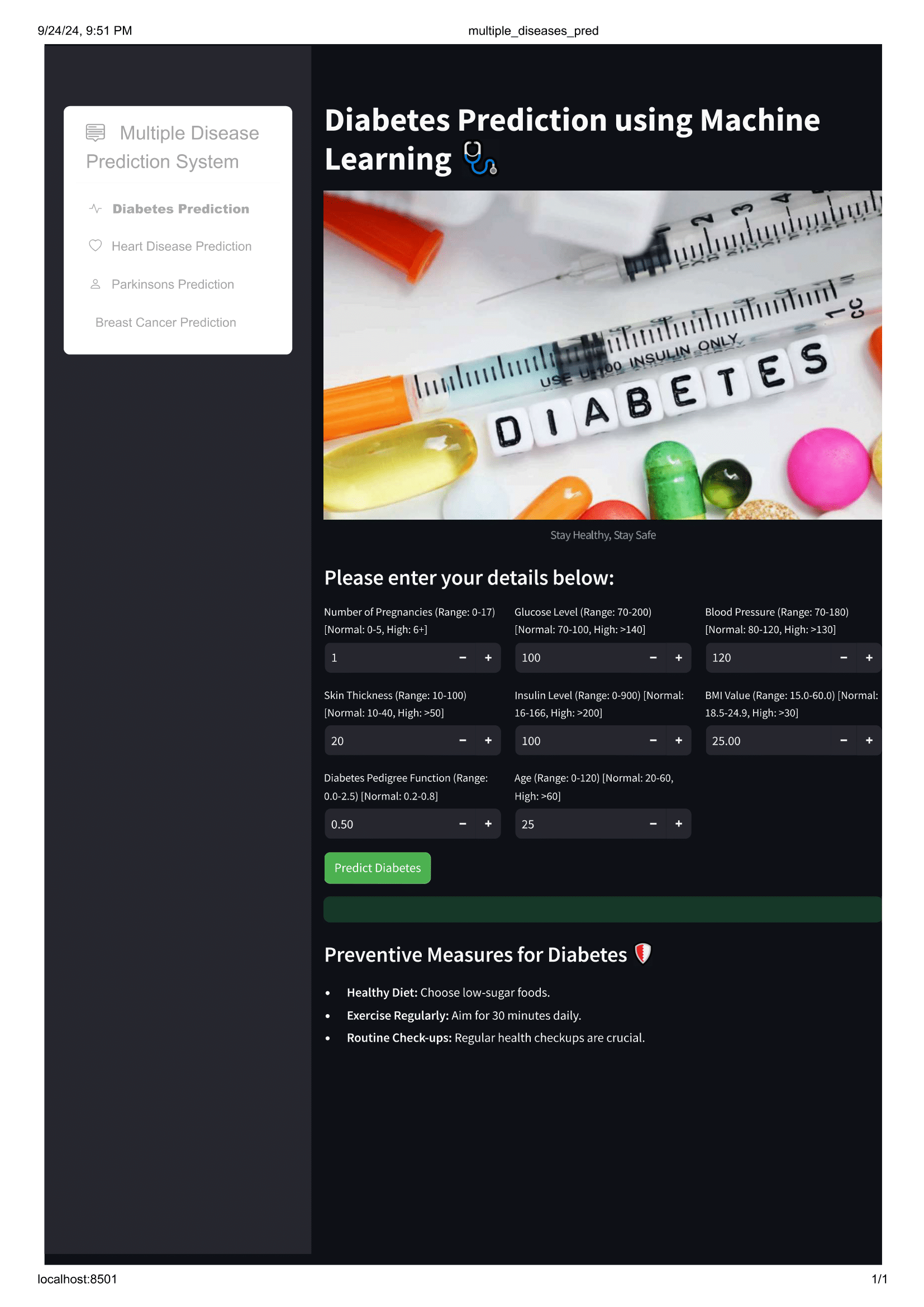
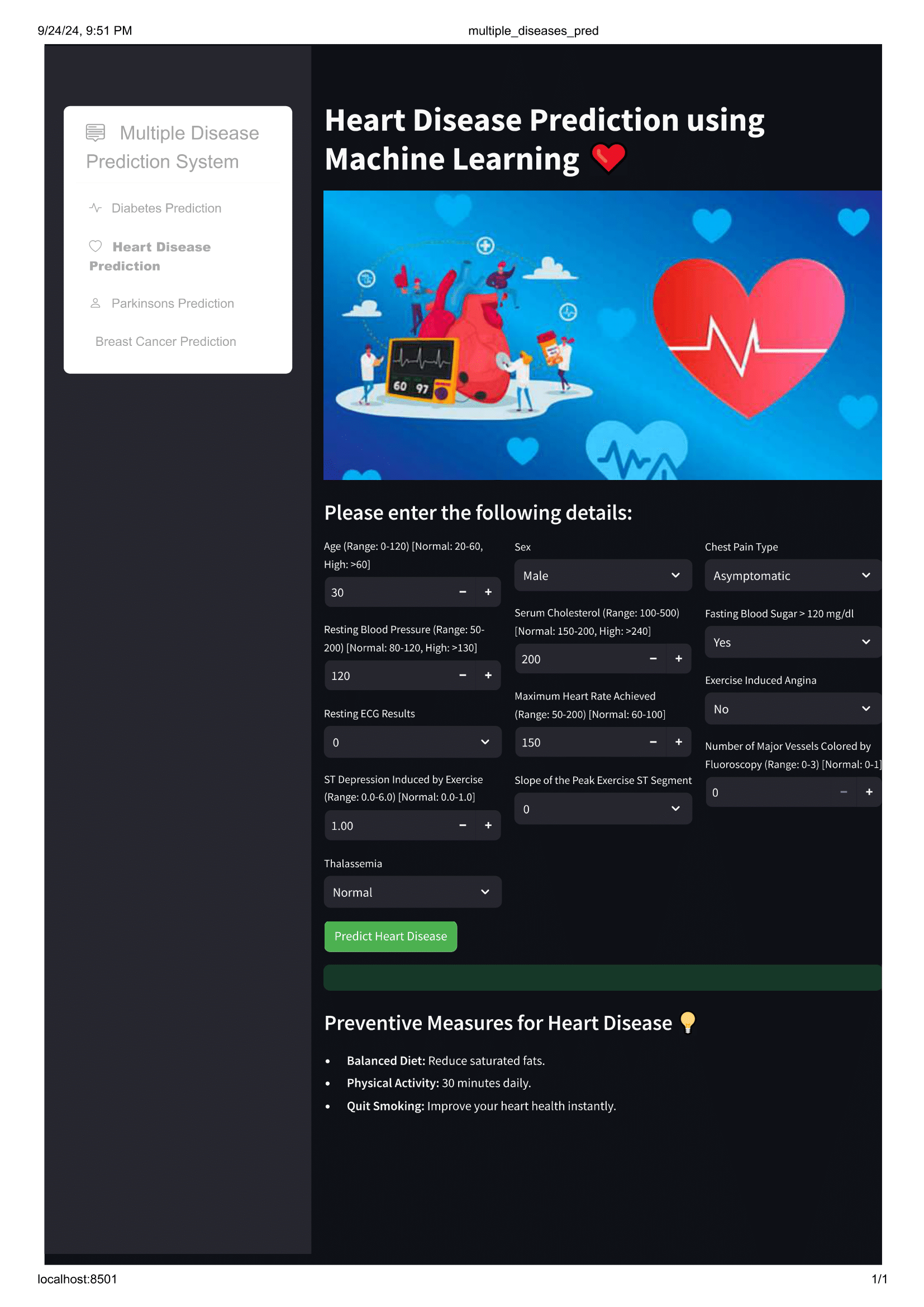
** 6.1 RESULTS**

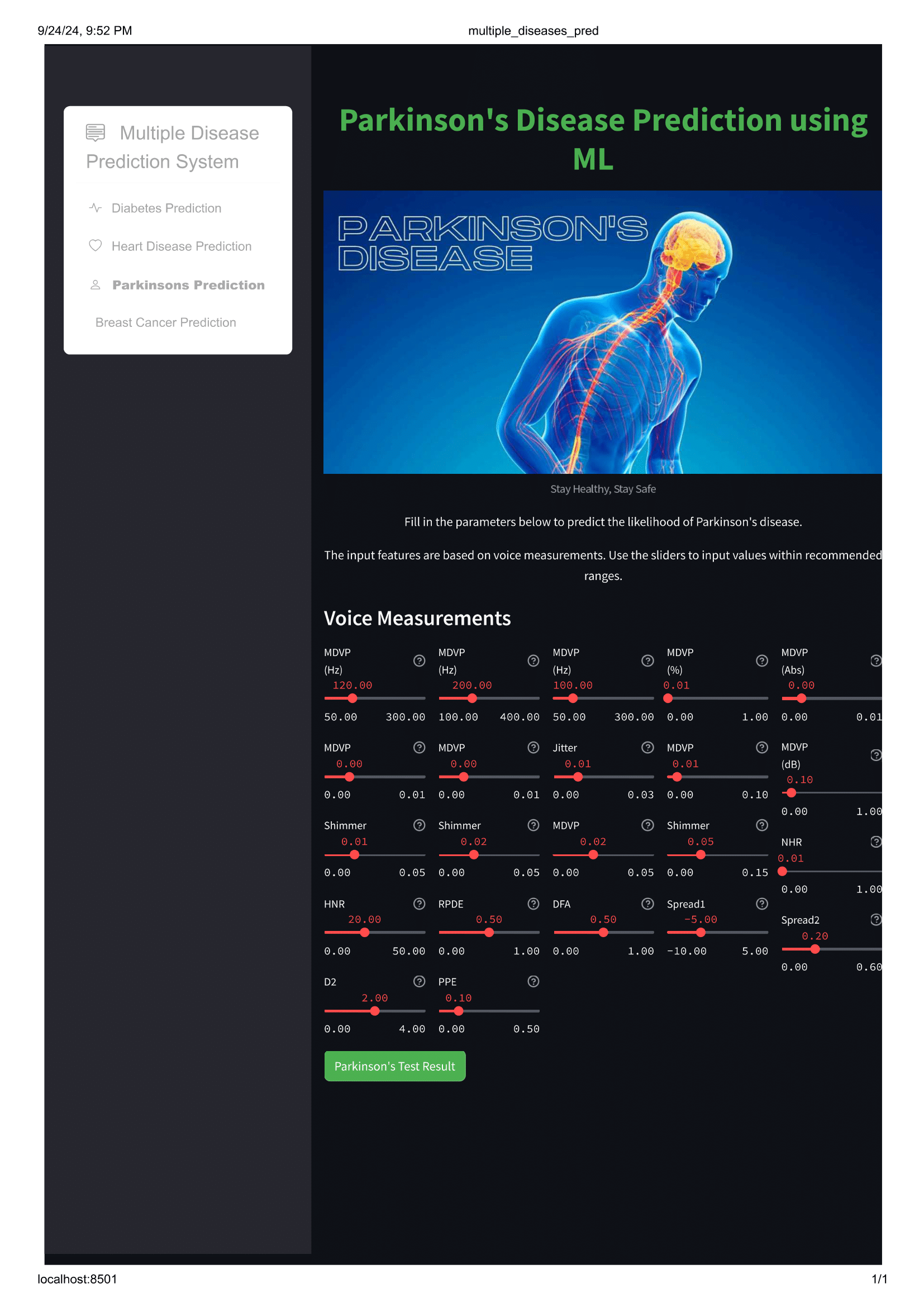
Fig: 6.1.1 Result Diabetes Diseases

**Diabetes Prediction**:

* This module predicts the likelihood of diabetes based on inputs such as number of pregnancies, glucose level, blood pressure, skin thickness, insulin level, BMI, and age.
* It includes a "Predict Diabetes" button to display results.
* Preventive measures suggested: a healthy diet, regular exercise, and routine check-ups.
* ****Fig6.1.2 Result Heart Diseases

 **Heart Disease Prediction**:

* This module assesses heart disease risk using factors like age, sex, blood pressure, cholesterol, fasting blood sugar, and exercise-induced angina.
* Users can click "Predict Heart Disease" to see their risk level.
* Preventive advice: balanced diet, regular physical activity, and avoiding smoking

****Fig 6.1.3 Result Parkinsons Diseases

**Parkinson's Disease Prediction**:

* This module predicts Parkinson's disease likelihood using voice measurements like frequency, jitter, shimmer, and other voice-related parameters.
* Users can adjust values and click "Parkinson's Test Result" for the prediction.
* Focused primarily on inputting accurate voice measurements for reliable results.

6.2 Conclusions And Future Work

6.2.1 Conclusions

The Multiple Disease Prediction System using Machine Learning provides a unified platform for the early detection of diabetes, heart disease, and Parkinson's disease. By integrating various machine learning models into a single web-based application, the system offers users a convenient and accessible solution to predict these diseases based on inputted health data.

This system not only reduces the time and cost associated with traditional diagnostic methods but also enables users, regardless of their technical expertise, to receive quick, accurate predictions. However, the success of the system depends on the quality of input data, and it should be seen as a complementary tool rather than a replacement for professional medical consultation. With the advancements in machine learning and healthcare, this project holds significant potential to enhance early disease detection, contributing to better patient outcomes and more effective management of chronic conditions

6.2.2 Future Work

Incorporating more data sources: Currently, multiple disease prediction systems typically rely on electronic health records and medical imaging data. In the future, other data sources such as wearable devices, social media, and environmental data could be integrated into these systems to provide a more comprehensive picture of a patient's health. Addressing data bias: As with all machine learning algorithms, bias in the training data can lead to inaccurate predictions and perpetuate health disparities. Future work should focus on developing methods to address and mitigate data bias, such as using more diverse and representative datasets, and incorporating fairness and equity considerations into the algorithm development process.

Advancing personalized medicine: Multiple disease prediction using machine learning has the potential to enable more personalized and precise medicine, by predicting an individual's risk of developing specific diseases based on their unique medical history and other factors. Future work should focus on developing personalized treatment plans based on these predictions, including targeted prevention strategies and personalized treatment options.

7. References

1. Tiwari, A. K. (2016). Machine Learning Based Approaches For Prediction Of Parkinson’s Disease. Machine Learning And Applications: An International Journal
2. Ricciardi, C., Et Al. (2019). Using Gait Analysis Parameters To Classify Parkinsonism: A Data Mining Approach. Computer Methods And Programs In Biomedicine, 180.
3. Bhatia, A., & Sulekh, R. (2017). Predictive Model For Parkinson’s Disease Through Naive Bayes Classification. International Journal Of Computer Science & Communication, 9, 194-202.

LIST OF FIGURES

Fig.4.2.1

SYSTEM ARCHITECTURE

11

FIG.4.3.2

CLASS DIAGRAM

13

FIG.4.3.2

USE CASE DIAGRAM

14

FIG.4.3.3

SEQUENCE DIAGRAM

15

FIG.4.3.4

COMPONENT DIAGRAM

16

FIG.4.3.5

DEPLOYMENT DIAGRAM

17

FIG.5.3.1

K VALUE GRAPH

30

FIG.8.1.1

DIABETES PREDICTION HOME PAGE

45

FIG.8.1.2

DIABETES PREDICTION RESULT PAGE

45

FIG.8.2.1

HEART DISEASE PREDICTION HOME PAGE

46

FIG.8.2.2

HEART DISEASE PREDICTION HOME PAGE

46

FIG.8.3.1

PARKINSON’S DISEASE PREDICTION HOME PAGE

47

FIG.8.3.2

PARKINSON’S DISEASE PREDICTION RESULT PAGE

47