ABSTRACT

The project "Classification of Multiple Diseases by applying technique of Random Forest " aims to leverage machine learning for the early detection and diagnosis of various diseases. The system utilizes the Random Forest algorithm, a robust and versatile machine learning model, to analyze clinical and demographic data and predict the likelihood of conditions such as Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease. The project involves several key components, including data collection, preprocessing, model training, and a user-friendly web interface for data input and result visualization. A comprehensive dataset was employed, with careful feature selection and data preprocessing to ensure model accuracy and reliability. The system's web interface, developed using Django, allows users to input their medical data and receive instant predictions, presented in an intuitive and accessible format. The model's performance is assessed using metrics like accuracy, precision, recall, and F1-score, demonstrating high efficacy in disease prediction. This project highlights the potential of machine learning in transforming healthcare diagnostics, offering a cost-effective and efficient solution for early disease detection. It also sets the stage for future enhancements, such as expanding the range of diseases covered, improving model accuracy, and integrating with electronic health records (EHR) systems. By providing actionable insights and facilitating early intervention, this system has the potential to significantly improve patient outcomes and support healthcare professionals in making informed clinical decisions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.No.	ABBREVIATIONS	FULL FORM
1	WHO	World Health Organization
2	RF	Random Forest
3	MLP	Multilayer perceptron
4	LR	Logistic Regression
5	SVM	Support Vector Machine
6	LSTM	Long Short Term Memory
7	LUAD	Lung adenocarcinoma
8	NSCLC	Non-small cell lung cancer
9	PCA	Principal Component Analysis
10	SMOTE	Synthetic Minority Oversampling Technique
11	PD	Parkinson’s Disease
12	SRS	System Requirements Specification
13	SDLC	Software development Life Cycle
14	BMI	Body Mass Index
15	UML	Unified Modeling Language
16	UI	User Interface

CHAPTER 1

INTRODUCTION

The healthcare industry has always sought to harness technology to improve patient outcomes and streamline diagnostic processes. In recent years, the application of machine learning has emerged as a transformative approach, offering sophisticated tools for analyzing vast amounts of medical data and predicting disease outcomes with remarkable accuracy. This project focuses on the development of a predictive system using the Random Forest machine learning technique to diagnose four critical diseases: Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease. Each of these diseases presents significant health challenges worldwide, with early detection being crucial for effective treatment and management.

Diabetes is a chronic condition characterized by high levels of glucose in the blood. According to the World Health Organization (WHO), diabetes affects millions of people globally and can lead to serious complications if not managed properly. Early prediction and diagnosis are essential for preventing severe health issues, including heart disease, kidney failure, and nerve damage.

Heart Disease remains one of the leading causes of death worldwide. It encompasses a range of conditions affecting the heart, including coronary artery disease, arrhythmias, and heart failure. Predictive modeling in heart disease can help identify at-risk individuals early on, allowing for timely interventions and lifestyle changes that can mitigate the risk of severe cardiac events.

Lung Cancer is the most common cause of cancer-related deaths globally. Early detection significantly improves survival rates, but symptoms often do not appear until the disease is advanced. Machine learning models can analyze patient data to identify early signs of lung cancer, thus facilitating early diagnosis and increasing the chances of successful treatment.

Parkinson's Disease is a progressive neurological disorder that affects movement. Early detection can improve the quality of life for patients through early treatment and management strategies. Machine learning can help identify early symptoms and patterns in patient data that are indicative of Parkinson's Disease.

The proposed project aims to develop a system that integrates predictive models for these four diseases using the Random Forest algorithm. Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks. It is known for its robustness, accuracy, and ability to handle large datasets with high dimensionality. By leveraging Random Forest, this project intends to create a reliable and efficient tool for early disease detection. The technology stack for this project will include Python programming language and its associated machine learning libraries such as scikit-learn, Pandas, and NumPy. Python is chosen due to its simplicity, versatility, and extensive support for machine learning and data analysis tasks. Scikit-learn provides a robust framework for implementing machine learning algorithms, while Pandas and NumPy are essential for data manipulation and numerical operations. It aims to demonstrate the practical application of machine learning techniques in solving real-world health problems, thereby contributing to the broader field of predictive analytics in medicine. The successful implementation of this project could pave the way for more advanced predictive systems that can assist healthcare professionals in making decisions, ultimately improving patient outcomes and reducing the burden on healthcare systems.

1.1 Main Objectives:

- Develop machine learning models to predict multiple diseases (Diabetes, Heart Disease, Lung Cancer, Parkinson's Disease). Enhance early detection to improve patient outcomes.
- Leverage the Random Forest algorithm for its robustness in handling large datasets and complex feature interactions.
- Integrate diverse medical datasets to enhance the model's accuracy and generalizability.
- Improve early detection rates to enable timely and effective treatment.
- Evaluate the model's performance using standard metrics and ensure its applicability in real-world healthcare settings.

1.2 PROBLEM STATEMENT:

Healthcare systems globally face significant challenges in the early and accurate diagnosis of chronic and life-threatening diseases. Traditional diagnostic methods, while effective, are often time-consuming, costly, and require significant medical expertise. These limitations can delay diagnosis and treatment, potentially leading to poorer patient outcomes. Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease are four conditions where early detection is particularly crucial. Diabetes and Heart Disease, for instance, benefit immensely from early lifestyle and medical interventions. Lung Cancer has a significantly higher survival rate if detected early, and Parkinson's Disease management can greatly improve patient quality of life when symptoms are identified early. Current diagnostic tools typically focus on a single disease, necessitating multiple tests for patients presenting with complex or unclear symptoms. This project seeks to address these issues by developing a unified machine learning model capable of predicting the likelihood of multiple diseases based on patient data.

By creating an integrated prediction system, this project aims to streamline the diagnostic process, reduce the time and cost associated with multiple diagnostic tests, and improve early detection rates. Ultimately, this system seeks to enhance patient care by providing healthcare professionals with a powerful tool to make quicker and more informed decisions.

1.3.1 METHODOLOGY:

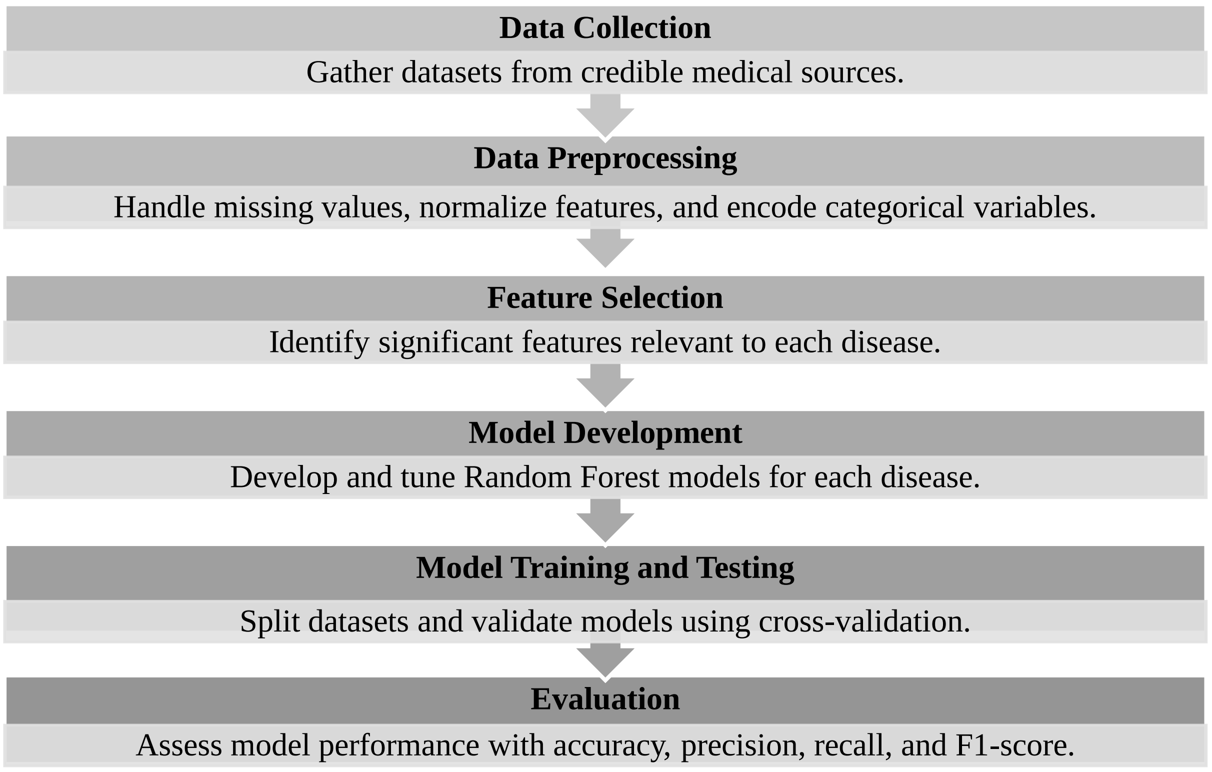

Data Collection
Gather datasets from credible medical sources.
Data Preprocessing
Handle missing values, normalize features, and encode categorical variables.
Feature Selection
Identify significant features relevant to each disease.
Model Development
Develop and tune Random Forest models for each disease.
Model Training and Testing
Split datasets and validate models using cross-validation.
Evaluation
Assess model performance with accuracy, precision, recall, and F1-score.

Figure 1.1 Methodology

1. **Data Collection:**

   - Gather datasets from credible medical sources such as hospital records, public health databases, and specialized medical research repositories.
   - Ensure the datasets contain relevant features such as patient demographics, medical history, laboratory test results, and clinical observations for Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease.

2. **Data Preprocessing:**

   - Handle missing values through techniques like imputation or removal, ensuring the dataset remains robust.
   - Normalize numerical features to ensure uniformity and improve the performance of the Random Forest model.
   - Encode categorical variables using methods like one-hot encoding to convert them into a format suitable for machine learning algorithms.

3. **Feature Selection:**

   - Identify significant features that contribute to the prediction of each disease using techniques like correlation analysis and feature importance from initial Random Forest runs.
   - Select the most relevant features for each disease to enhance model accuracy and reduce computational complexity.

4. **Model Development:**

- Develop individual Random Forest models for each disease prediction.
- Tune hyperparameters (e.g., number of trees, maximum depth) using techniques like grid search or random search to optimize model performance.

5. **Model Training and Testing:**

- Split the dataset into training and testing sets to evaluate model performance.
- Train the models on the training data and validate their accuracy on the testing data using cross-validation techniques to prevent overfitting.

6. **Evaluation:**

- Evaluate the performance of each model using metrics such as accuracy, precision, recall, and F1-score.
- Use confusion matrices to understand the classification performance and make necessary adjustments to the models.

CHAPTER 2

LITERATURE SURVEY:

Diabetes Prediction:

[1] Machine Learning Based Diabetes Classification and Prediction for Healthcare Applications

Authors: Umair Muneer Butt,Sukumar Letchmunan, Mubashir Ali, Fadratul Hafinaz Hassan, Anees Baqir, and Hafiz Husnain Raza Sherazi

Link: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8500744/

Abstract: The remarkable advancements in biotechnology and public healthcare infrastructures have led to a momentous production of critical and sensitive healthcare data. By applying intelligent data analysis techniques, many interesting patterns are identified for the early and onset detection and prevention of several fatal diseases. Diabetes mellitus is an extremely life-threatening disease because it contributes to other lethal diseases, i.e., heart, kidney, and nerve damage. In this paper, a machine learning based approach has been proposed for the classification, early-stage identification, and prediction of diabetes. Furthermore, it also presents an IoT-based hypothetical diabetes monitoring system for a healthy and affected person to monitor his blood glucose (BG) level. For diabetes classification, three different classifiers have been employed, i.e., random forest (RF), multilayer perceptron (MLP), and logistic regression (LR). For predictive analysis, we have employed long short-term memory (LSTM), moving averages (MA), and linear regression (LR). For experimental evaluation, a benchmark PIMA Indian Diabetes dataset is used. During the analysis, it is observed that MLP outperforms other classifiers with 86.08% of accuracy and LSTM improves the significant prediction with 87.26% accuracy of diabetes.

Heart Disease Prediction:

[2] Prediction of Heart Disease Based on Machine Learning Using Jellyfish Optimization Algorithm

Authors: Ahmad Ayid Ahmad, Huseyin Polat, Yanwu Xu

Link:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10378171/

Abstract: Heart disease is one of the most known and deadly diseases in the world, and many people lose their lives from this disease every year. Early detection of this disease is vital to save people's lives. Machine Learning (ML), an artificial intelligence technology, is one of the most convenient, fastest, and low-cost ways to detect disease. In this study, we aim to obtain an ML model that can predict heart disease with the highest possible performance using the Cleveland heart disease dataset. The features in the dataset used to train the model and the selection of the ML algorithm have a significant impact on the performance of the model. To avoid overfitting (due to the curse of dimensionality) due to the large number of features in the Cleveland dataset, the dataset was reduced to a lower dimensional subspace using the Jellyfish optimization algorithm. The Jellyfish algorithm has a high convergence speed and is flexible to find the best features. The models obtained by training the featureselected dataset with different ML algorithms were tested, and their performances were compared. The highest performance was obtained for the SVM classifier model trained on the dataset with the Jellyfish algorithm, with Sensitivity, Specificity, Accuracy, and Area Under Curve of 98.56%, 98.37%, 98.47%, and 94.48%, respectively. The results show that the combination of the Jellyfish optimization algorithm and SVM classifier has the highest performance for use in heart disease prediction.

Lung Cancer Prediction:

[3] ENHANCING LUNG CANCER CLASSIFICATION AND PREDICTION WITH DEEP LEARNING AND MULTI-OMICS DATA

Authors: Tehnan I. A. Mohamed, Absalom El-Shamir Ezugwu

Link: https://ieeexplore.ieee.org/document/10508786

Abstract: Lung adenocarcinoma(LUAD), a prevalent histological type of lung cancer and a subtype of non-small cell lung cancer(NSCLC) accounts for 45–55% of all lung cancer cases. Various factors, including environmental influences and genetics, have been identified as contributors to the initiation and progression of LUAD. In this study, we devised an innovative deep-learning model for lung cancer detection by integrating markers from mRNA, miRNA, and DNA methylation. The initial phase involved meticulous data preparation, encompassing multiple steps, followed by a differential analysis aimed at identifying genes exhibiting differential expression across different lung cancer stages (Stages I, II, III, and IV). The DESeq2 technique was employed for RNASeq data, while the LIMMA package was utilized for miRNA and DNA methylation datasets during the differential analysis. Subsequently, integration of all prepared omics data types was achieved by selecting common samples, resulting in a consolidated dataset comprising 448 samples and 8228 features (genes). To streamline features, principal components analysis (PCA) was implemented, and the synthetic minority over-sampling technique (SMOTE) algorithm was applied to ensure class balance. The integrated and processed data were then input into the PCA-SMOTE-CNN model for the classification process. The deep learning model, specifically designed for classifying and predicting lung cancer using an integrated omics dataset, was evaluated using various metrics, including precision, recall, F1-score, and accuracy. Experimental results emphasized the superior predictive performance of the proposed model, attaining an accuracy, precision, recall, and F1-score of 0.97 each, surpassing recent competitive methods.

Parkinson's Disease Prediction:

[4] DEEP TRANSFER LEARNING BASED PARKINSON’S DISEASE DETECTION USING OPTIMIZED FEATURE SELECTION.

Authors: Sura Mahmood Abdullah, Thekra Abbas, Munzir Hubiba Bashir, Ishfaq Ahmad Khaja, Musheer Ahmad, Naglaa F.Soliman, Walid El-Shafai

Link: **https://ieeexplore.ieee.org/document/10005184/authors#authors**

Abstract:

Parkinson’s disease (PD) is one of the chronic neurological diseases whose progression is slow and symptoms have similarities with other diseases. Early detection and diagnosis of PD is crucial to prescribe proper treatment for patient’s productive and healthy lives. The disease’s symptoms are characterized by tremors, muscle rigidity, slowness in movements, balancing along with other psychiatric symptoms. The dynamics of handwritten records served as one of the dominant mechanisms which support PD detection and assessment. Several machine learning methods have been investigated for the early detection of this disease. But most of these handcrafted feature extraction techniques predominantly suffer from low performance accuracy issues. To this end, an efficient deep learning model is proposed which can assist to have early detection of Parkinson’s disease. The significant contribution of the proposed model is to select the most optimum features which have the effect of getting the high-performance accuracies. The feature optimization is done through genetic algorithm wherein K -Nearest Neighbour technique. The proposed novel model results into detection accuracy higher than 95%, precision of 98%, area under curve of 0.90 with a loss of 0.12 only. The performance of proposed model is compared with some state-of-the-art machine learning and deep learning-based PD detection approaches to demonstrate the better detection ability of our model.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System:

1. Manual Diagnosis:
   - Disease diagnosis often relies on manual evaluation by healthcare professionals, which can be time-consuming and prone to human error.
   - Diagnostic procedures may involve multiple tests and assessments, leading to delays in diagnosis and treatment.
2. Single Disease Prediction Models:
   - Existing systems typically focus on predicting a single disease at a time, requiring multiple models to be developed and maintained for different diseases.
   - This approach can lead to inefficiencies in resource utilization and increased complexity in managing multiple models.
3. Limited Use of Machine Learning:
   - Traditional methods may not leverage the full potential of machine learning techniques, which can limit the accuracy and efficiency of disease prediction.
   - Systems may not utilize advanced algorithms that can handle large datasets and extract complex patterns.
4. Data Management Challenges:
   - Existing systems might struggle with handling and processing large volumes of medical data.
   - Data cleaning, preprocessing, and integration from various sources can be cumbersome and error-prone.

3.2 Proposed System

1. Automated Multi-Disease Prediction:

- The proposed system utilizes a Random Forest classifier to predict multiple diseases simultaneously, including Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease.
- Automation of the diagnostic process reduces the need for manual intervention, improving efficiency and consistency in disease prediction.

2. Integration of Machine Learning Techniques:

- The system leverages advanced machine learning techniques, specifically the Random Forest algorithm, to analyze and interpret complex medical data.
- This approach enhances the accuracy of predictions by combining the results of multiple decision trees.

3. Comprehensive Data Handling:

- The proposed system incorporates robust data cleaning, preprocessing, and feature selection methods to ensure high-quality input data for the model.
- The use of standardized processes for handling data from various sources ensures better data integration and management.

4. Enhanced Performance Metrics:

- The system evaluates model performance using metrics such as accuracy, precision, recall, and F1-score, ensuring a thorough assessment of the model's effectiveness.
- Continuous monitoring and validation of the model ensure sustained performance over time.

5. User-Friendly Interface:

- The system includes a user-friendly frontend developed using Django, making it accessible to healthcare professionals and users without technical expertise.
- The interface allows for easy input of patient data and provides clear and concise prediction results.

6. Scalability and Flexibility:

- The proposed system is designed to be scalable, allowing for the addition of new diseases and the incorporation of larger datasets as needed.
- The modular design ensures that components can be updated or replaced independently, facilitating ongoing improvements and customization.

3.3 Implementation Plan

1. Data Collection and Preprocessing:

- Collect data from reputable sources such as Kaggle, UC Irvine ML Repository, and Mendeley Data.
- Clean and preprocess the data using libraries like NumPy and Pandas, and visualize it using Seaborn and Matplotlib.

2. Feature Selection and Model Training:

- Use the Holdout Method to split the data into training and testing sets.
- Train the Random Forest classifier on the training data and evaluate its performance on the testing data.

3. Model Deployment:

- Store the trained model in a pickle file for easy deployment.
- Develop a Django-based frontend to facilitate user interaction with the model.

4. Performance Monitoring and Maintenance:

- Continuously monitor the performance of the model and update it with new data as needed.
- Ensure regular maintenance of the system to address any issues and incorporate improvements.

CHAPTER 4

SOFTWARE REQUIREMENTS

4.1 Software Requirements:

- Programming Language: Python
- Modules: scikit-learn, Pandas, NumPy, Matplotlib
- Software: Anaconda Jupyter Notebook
- Tools: PIP
- Framework: Django
- Database: SQLite

4.2 Hardware Requirements:

- OS: Windows only
- Processor: i5
- RAM: 8GB or above
- Hard Disk: 25GB or above

4.3 SOFTWARE REQUIREMENT SPECIFICATION:

What is SRS?

Software Requirements Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase).

The SRS phase consists of two basic activities:

Problem/Requirement Analysis:

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

Requirement Specification:

Here, the focus is on specifying what has been found giving analysis such as representation, Specification languages and tools, and checking the specifications are addressed during this activity.

The requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic of this phase.

Role of SRS:

The purpose of the SRS is to reduce the communication gap between the clients and the developers. SRS is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

Purpose:

The purpose of this document is to describe all external requirements for the E-learning System. It also describes the interfaces for the system.

<u>Scope</u>**:**

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also by the basis for validating the final deliver system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for asking for clarifications, where necessary, and will not make any alternations without the permission of the client.

Overview**:**

The SRS begins the translation process that converts the software Requirements into the language the developers will use. The SRS draws on the Use Cases from the user Requirement Document and analyses the situations from a number of perspectives to discover and eliminate inconsistencies, ambiguities and omissions before development progresses significantly under mistaken assumptions.

Software Development Life Cycle:

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.

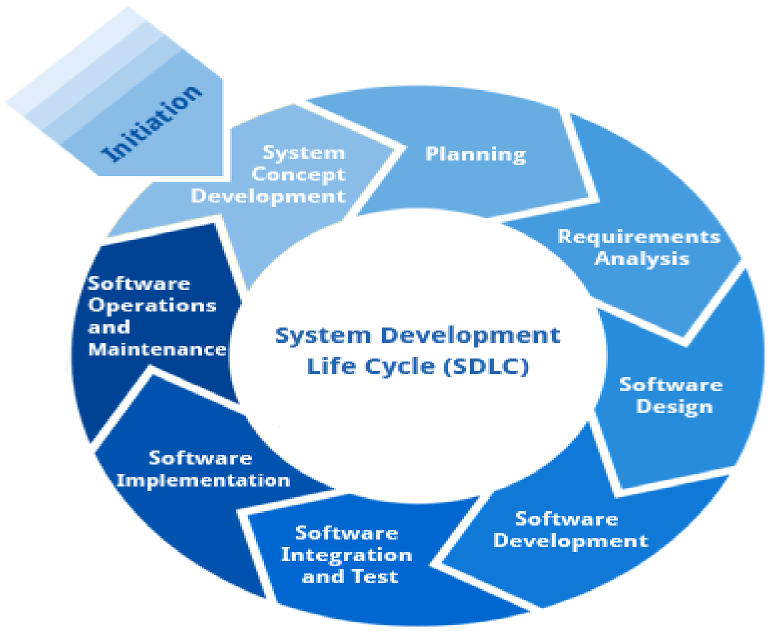

Initiation
System Concept Development
Planning
Requirements Analysis
Software Design
Software Development
Software Integration and Test
Software Implementation
Software Operations and Maintenance
System Development Life Cycle (SDLC)

Figure 4.1 SDLC

Requirement Analysis and Design:

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

Implementation:

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like PYTHON 3.6, Anaconda Cloud are used for coding. With respect to the type of application, the right programming language is chosen.

Testing:

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

Maintenance:

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

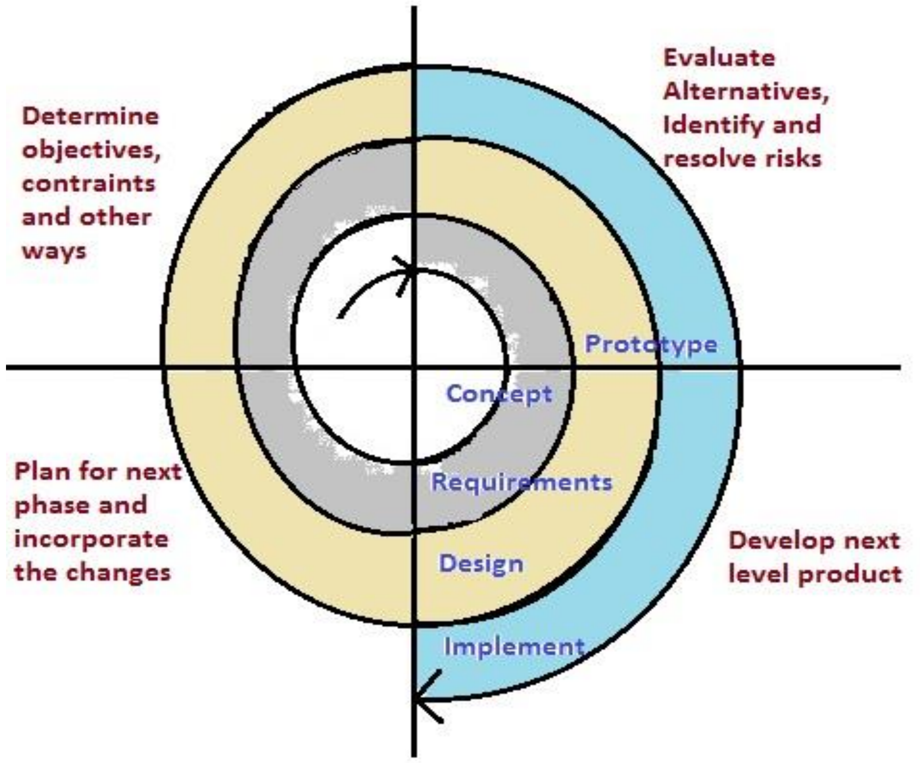
SDLC METHDOLOGIES:

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The following diagram shows how a spiral model acts like:



Determine objectives, contraints and other ways
Evaluate Alternatives, Identify and resolve risks
Prototype
Concept
Requirements
Design
Implement
Plan for next phase and incorporate the changes
Develop next level product

Figure 4.2 Spiral Model

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible.
- This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

- A second prototype is evolved by a fourfold procedure:

    1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
    2. Defining the requirements of the second prototype.
    3. Planning a designing the second prototype.
    4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

CHAPTER 5

SYSTEM STUDY

The system study involves understanding the requirements, challenges, and opportunities associated with implementing a machine learning-based solution for predicting multiple diseases using the Random Forest technique. The goal is to design and develop a system that automates the diagnostic process, ensuring accuracy and efficiency.

5.1 Feasibility Study:

The feasibility study assesses the practicality and viability of implementing the proposed system. It considers technical, operational, economic, and legal factors to determine if the project is achievable and worthwhile.

1. Technical Feasibility:

- ➢ Data Availability:
    - Sufficient medical datasets for multiple diseases, such as Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease, are available from sources like Kaggle, UC Irvine ML Repository, and Mendeley Data.
- ➢ Technological Requirements:
    - The system requires machine learning libraries (e.g., Scikit-learn, NumPy, Pandas), data visualization tools (e.g., Matplotlib, Seaborn), and a web development framework (e.g., Django).
    - Computational resources, such as a server with adequate processing power and memory, are necessary for training and deploying the model.

- Model Selection:
    - The Random Forest classifier is chosen due to its ability to handle large datasets, provide high accuracy, and manage feature importance, making it suitable for this multi-disease prediction task.

2. Operational Feasibility

- User Requirements:
    - The system should be user-friendly, allowing healthcare professionals and users to input data and receive predictions without requiring technical expertise.
- Implementation Strategy:
    - The system will be implemented in stages, starting with data collection and preprocessing, followed by model training, deployment, and frontend development.

3. Economic Feasibility

- Cost-Benefit Analysis:
    - Initial costs include data acquisition, software development, and computational resources.
    - Benefits include improved diagnostic accuracy, reduced time for diagnosis, and potential cost savings in healthcare by early detection and treatment.
- Return on Investment (ROI):
    - The system is expected to reduce diagnostic errors and streamline the healthcare process, leading to potential cost savings and improved patient outcomes.

4. Legal Feasibility

- Data Privacy and Security:
    - The system must comply with healthcare regulations and data protection laws (e.g., GDPR, HIPAA) to ensure patient data privacy and security.
    - Secure data storage and transmission methods will be implemented to protect sensitive information.
- Ethical Considerations:
    - Ethical guidelines will be followed to ensure fair and unbiased predictions, particularly in handling sensitive medical data.

5.2 Feasibility Analysis:

Based on the feasibility study, the proposed project "Classification of Multiple Diseases Using Random Forest Technique" is determined to be feasible. The analysis highlights:

- Technical Viability: The availability of suitable data and technology, along with the chosen model (Random Forest), supports the technical feasibility of the project.
- Operational Feasibility: The system can be designed to meet user requirements, ensuring ease of use and effective functionality.
- Economic Viability: The project presents a favorable cost-benefit ratio, with potential for significant ROI through improved diagnostics and healthcare efficiency.
- Legal and Ethical Feasibility: Compliance with legal and ethical standards can be maintained, ensuring the responsible handling of medical data.

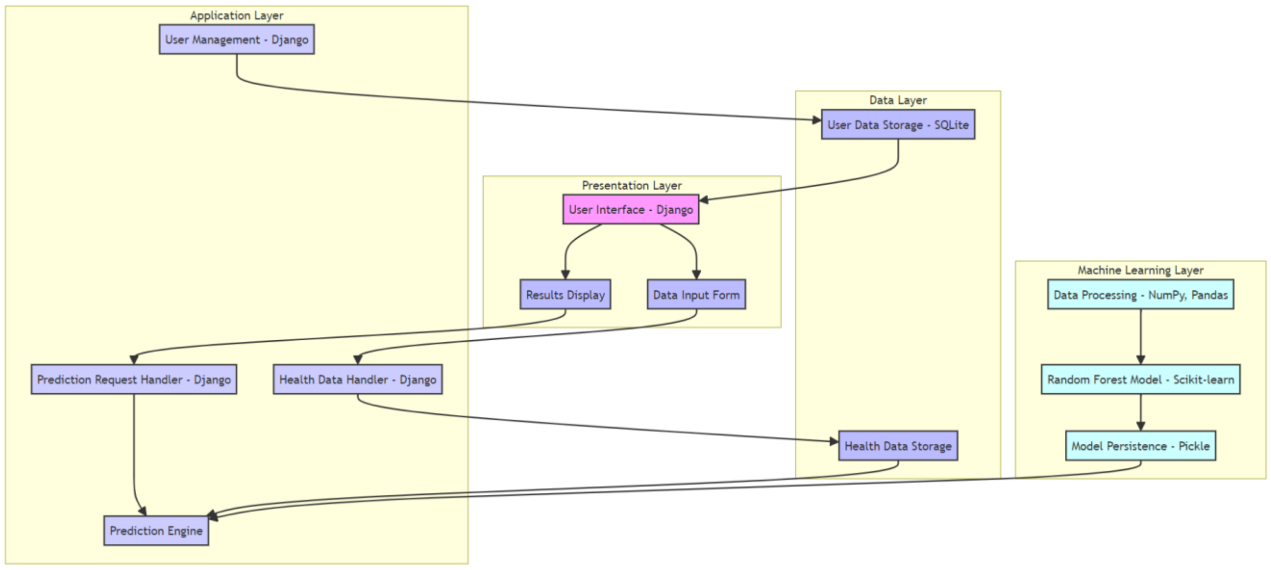
CHAPTER 6

SYSTEM DESIGN

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

6.1 SYSTEM ARCHITECTURE:



Application Layer
User Management - Django
Data Layer
User Data Storage - SQLite
Presentation Layer
User Interface - Django
Results Display
Data Input Form
Machine Learning Layer
Data Processing - NumPy, Pandas
Random Forest Model - Scikit-learn
Model Persistence - Pickle
Prediction Request Handler - Django
Health Data Handler - Django
Health Data Storage
Prediction Engine

Figure 6.1 SYSTEM ARCHITECTURE

Presentation Layer (Frontend):

- User Interface: Provides the web interface for user interactions.
- Data Input Form: Allows users to enter their health data.
- Results Display: Shows the predicted results to the users.

Application Layer (Backend):

- Health Data Handler: Validates and processes health data from the input form.
- Prediction Request Handler: Manages requests for disease prediction, communicates with the ML layer, and returns results.
- User Management: Manages user authentication and profile data.

Data Layer (Database):

- User Data Storage: Stores user credentials and profile data securely.
- Health Data Storage: Stores the health data submitted by users.

Machine Learning Layer:

- Data Processing: Prepares and preprocesses health data for prediction.
- Random Forest Model: Uses the pre-trained Random Forest algorithm to predict diseases.
- Model Persistence: Manages the saving and loading of the trained model.

6.2 UML DIAGRAMS:

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

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

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

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

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

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

GOALS:

The Primary goals in the design of the UML are as follows:
Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
Provide extendibility and specialization mechanisms to extend the core concepts.
Be independent of particular programming languages and development process. Provide a formal basis for understanding the modeling language.
Encourage the growth of OO tools market.

Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

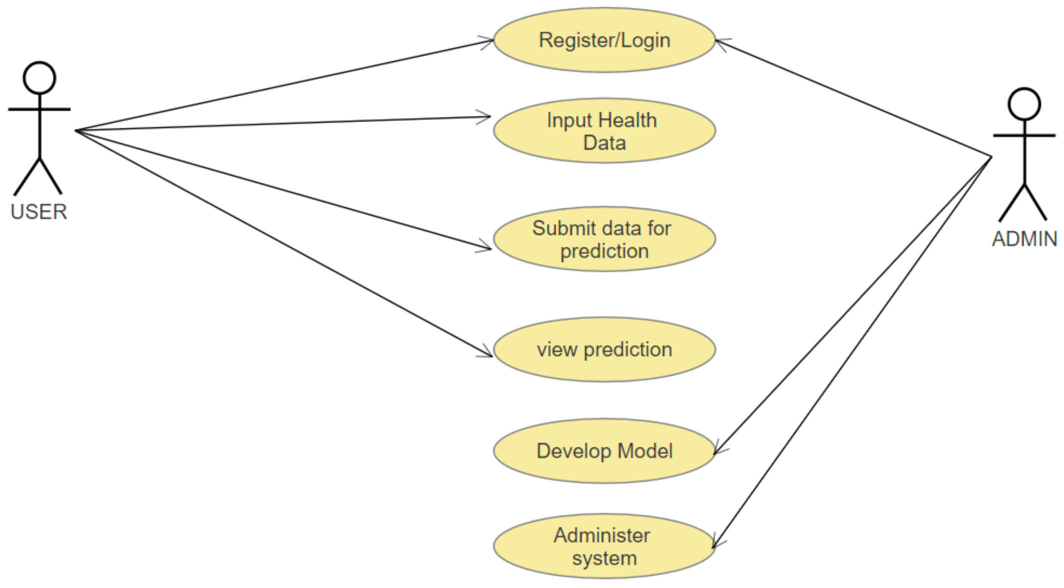


Figure 6.2 Use case diagram

Sequence diagram:

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioural classifier that represents a declaration of an offered behaviour. Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behaviour of the subject without reference to its internal structure. These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment.

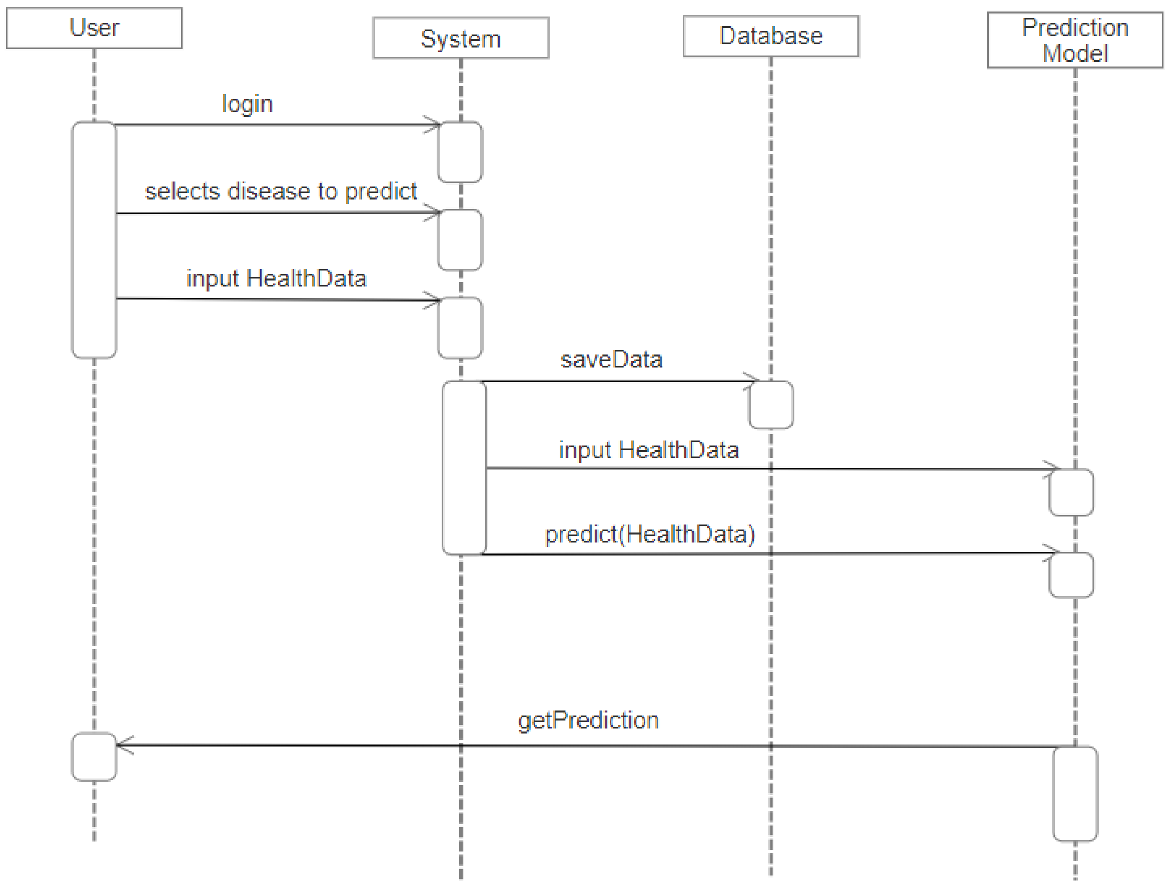


Figure 6.3 Sequence diagram

Activity diagram:

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

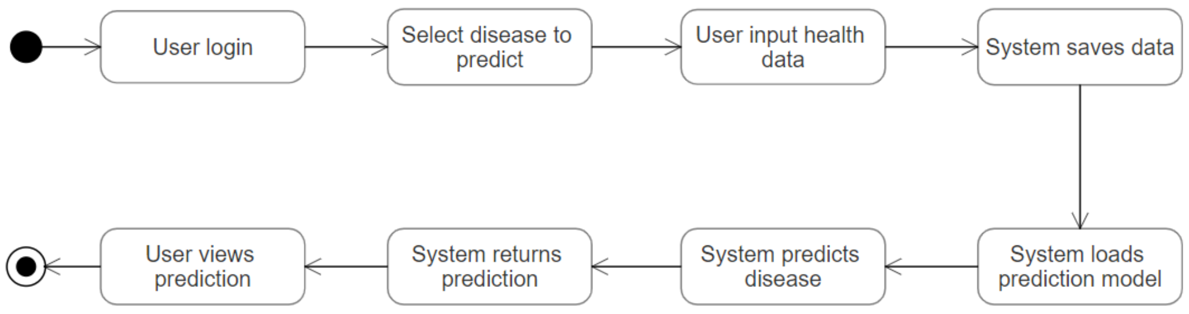


Figure 6.4 Activity diagram

CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

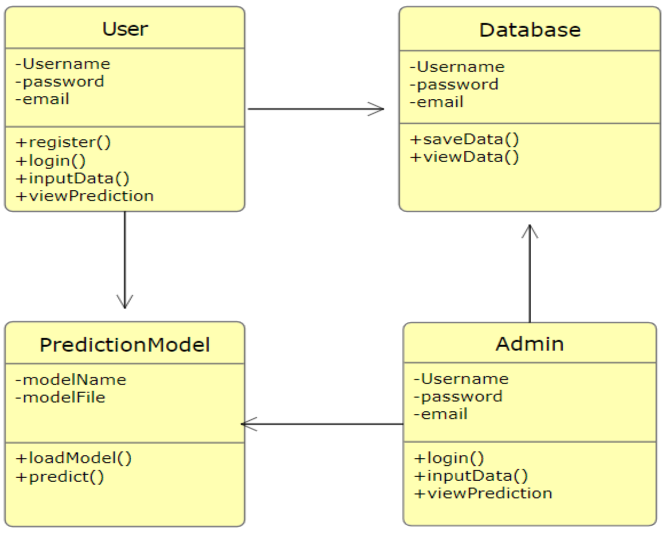


Figure 6.5 Class diagram

Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

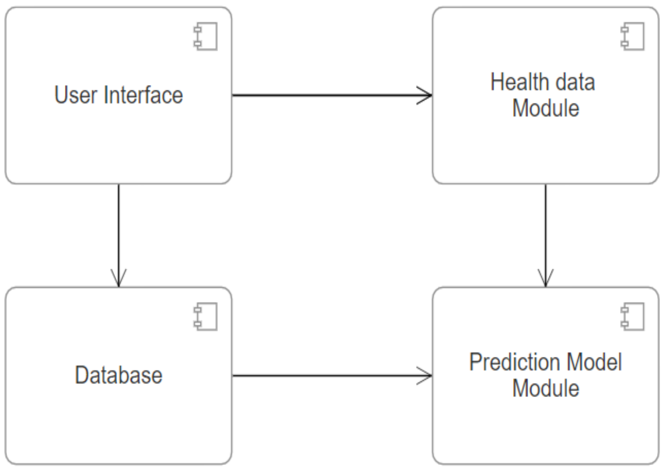


Figure 6.6 Component diagram

CHAPTER 7

INPUT DESIGN AND OUTPUT DESIGN

7.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

7.1.1OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed

in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

7.1.2 Inputs for each disease are:

Diabetes Disease:

Age: The age of the individual, typically measured in years.
Gender: The biological sex of the individual, often categorized as male, female.
Polyuria: A condition characterized by excessive urination, which can be a symptom of diabetes and other conditions.
Polydipsia: Excessive thirst, often associated with diabetes.
Sudden Weight Loss: A rapid and unexplained decrease in body weight, which can be indicative of several medical conditions, including diabetes.
Weakness: A general feeling of fatigue or lack of strength, which can be associated with various diseases.
Polyphagia: Excessive hunger or increased appetite, commonly seen in diabetes.
Genital Thrush: A fungal infection that affects the genital area, which can be a symptom of diabetes due to high sugar levels.
Visual Blurring: Difficulty seeing clearly, which can be related to a variety of health issues, including diabetes.
Itching: A sensation that provokes the desire to scratch, which can be caused by skin conditions or systemic issues.
Irritability: An increased tendency to become annoyed or angry, which can be a symptom of various mental or physical health conditions.
Delayed Healing: A slower than normal healing process for wounds, which can be a sign of underlying health issues, such as diabetes.

Partial Paresis: Weakness affecting only one part of the body, potentially indicating neurological issues.

Muscle Stiffness: Difficulty in moving muscles or joints, which can be associated with conditions like Parkinson's disease or arthritis.
Alopecia: Loss of hair, which can occur in localized patches or all over the body, sometimes associated with autoimmune conditions or stress.
Obesity: A condition characterized by excessive body fat, often defined by a high Body Mass Index (BMI).

Heart Disease:

cp (Chest Pain Type): The type of chest pain experienced, often classified into several categories, :
0: Typical angina, 1: Atypical angina, 2: Non-anginal pain
trestbps (Resting Blood Pressure): The resting blood pressure (in mm Hg) of the individual, typically measured at the hospital.
chol (Serum Cholesterol): The individual's serum cholesterol level in mg/dL.
fbs (Fasting Blood Sugar): Indicates whether the fasting blood sugar is greater than 120 mg/dL (1 = true; 0 = false).
->restecg (Resting Electrocardiographic Results): Results of the resting electrocardiogram, which can include:
 0: Normal
 1: Having ST-T wave abnormality (e.g., T wave inversions and/or ST elevation or depression of >0.05 mV)
 2: Showing probable or definite left ventricular hypertrophy according to Estes' criteria
thalach (Maximum Heart Rate Achieved): The maximum heart rate achieved during a stress test.
exang (Exercise Induced Angina): Indicates whether the individual experienced angina induced by exercise (1 = yes; 0 = no).
oldpeak (ST Depression Induced by Exercise Relative to Rest): This measures the amount of depression of the ST segment during exercise

relative to rest,
which can indicate the severity of coronary artery disease.
slope (Slope of the Peak Exercise ST Segment): The slope of the peak exercise ST segment, with possible values:
0: Upsloping, 1: Flat, 2: Downsloping
ca (Number of Major Vessels Colored by Fluoroscopy): The number of major vessels (0-3) colored by fluoroscopy.
thal (Thalassemia): A blood disorder with the following possible values:

0: Normal
1: Fixed defect
2: Reversible defect

Lung Cancer:

Age: The age of the individual, typically measured in years.

Gender: The biological sex of the individual, often categorized as male, female, or possibly other/unknown.

Polyuria: A condition characterized by excessive urination, which can be a symptom of diabetes and other conditions.

Polydipsia: Excessive thirst, often associated with diabetes.

Sudden Weight Loss: A rapid and unexplained decrease in body weight, which can be indicative of several medical conditions, including diabetes.

Weakness: A general feeling of fatigue or lack of strength, which can be associated with various diseases.

Polyphagia: Excessive hunger or increased appetite, commonly seen in diabetes.

Genital Thrush: A fungal infection that affects the genital area, which can be a symptom of diabetes due to high sugar levels.

Visual Blurring: Difficulty seeing clearly, which can be related to a variety of health issues, including diabetes.

Itching: A sensation that provokes the desire to scratch, which can be caused by skin conditions or systemic issues.

Irritability: An increased tendency to become annoyed or angry, which can be a symptom of various mental or physical health conditions.

Delayed Healing: A slower than normal healing process for wounds, which can be a sign of underlying health issues, such as diabetes.

Partial Paresis: Weakness affecting only one part of the body, potentially indicating neurological issues.

Muscle Stiffness: Difficulty in moving muscles or joints, which can be associated with conditions like Parkinson's disease or arthritis.

Alopecia: Loss of hair, which can occur in localized patches or all over the body, sometimes associated with autoimmune conditions or stress.

Obesity: A condition characterized by excessive body fat, often defined by a high Body Mass Index (BMI).

Parkinson's Disease:

MDVP(Hz) - Average vocal fundamental frequency.

MDVP(Hz) - Maximum vocal fundamental frequency.

MDVP(Hz) - Minimum vocal fundamental frequency.

MDVP(%) - Measure of variation in fundamental frequency.

MDVP(Abs) - Absolute measure of variation in fundamental frequency.

MDVP- Relative amplitude perturbation.

MDVP- Five-point period perturbation quotient.

Jitter- Average absolute difference of differences between consecutive periods.

MDVP- Measure of variation in amplitude.

MDVP(dB) - Variation in amplitude in decibels.

Shimmer- Three-point amplitude perturbation quotient.

Shimmer- Five-point amplitude perturbation quotient.

MDVP- Eleven-point amplitude perturbation quotient.

Shimmer- Average absolute difference between consecutive periods.

NHR - Noise-to-harmonics ratio.

HNR - Harmonics-to-noise ratio.

RPDE - Recurrence period density entropy.

DFA - Detrended fluctuation analysis.

spread1 - Nonlinear measure of fundamental frequency variation.

spread2 - Nonlinear measure of fundamental frequency variation.

D2 - Correlation dimension.

PPE - Pitch period entropy.

7.2 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

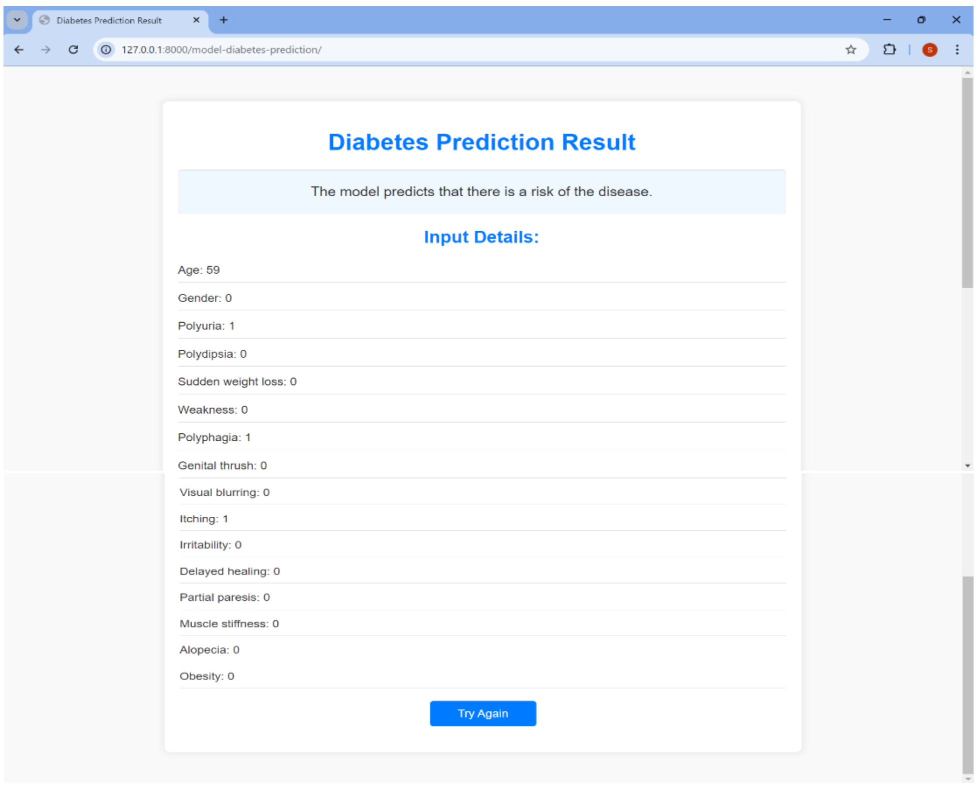
1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.


Diabetes Prediction Result
127.0.0.1:8000/model-diabetes-prediction/

Diabetes Prediction Result

The model predicts that there is a risk of the disease.

Input Details:

Age: 59
Gender: 0
Polyuria: 1
Polydipsia: 0
Sudden weight loss: 0
Weakness: 0
Polyphagia: 1
Genital thrush: 0
Visual blurring: 0
Itching: 1
Irritability: 0
Delayed healing: 0
Partial paresis: 0
Muscle stiffness: 0
Alopecia: 0
Obesity: 0

Figure 7.1 Output

CHAPTER 8

IMPLEMENTATION

The implementation of the "Classification of Multiple Diseases Using Random Forest Technique" project involves the development and integration of several key modules. Each module plays a crucial role in the overall system, ensuring that data is accurately processed, predictions are reliably generated, and results are effectively communicated to the user.

8.1 Modules:

- Data Collection and Preprocessing Module
- Model Training and Evaluation Module
- User Interface Module
- Prediction Module

8.2 Module Description:

Data Collection and Preprocessing Module:

This module is responsible for gathering data from various reputable sources such as Kaggle, UC Irvine ML Repository, and Mendeley Data. The data includes medical records, symptom details, and vocal features pertinent to diseases like Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease. The preprocessing step involves cleaning the data to remove duplicates and handle missing values, normalizing and scaling numerical features, and transforming categorical variables into numerical formats where necessary.

Model Training and Evaluation Module:

The heart of the implementation lies in this module, where the Random Forest algorithm is applied. The dataset is split into training and testing sets using the Holdout Method (typically 80% for training and 20% for testing). The model is trained on the training data, learning patterns and correlations among features. The performance of the model is then evaluated using metrics such as accuracy, precision, recall, and F1-score on the testing set.

User Interface Module:

Developed using Django, the User Interface (UI) module provides a platform for users to interact with the system. Users can input their data through a web-based form, which includes fields for all required features. The interface is designed to be intuitive and user-friendly, ensuring that even those without technical expertise can easily use the system. Validation mechanisms are in place to ensure correct data entry.

Prediction Module:

The Prediction Module is the core component of the system where the actual disease prediction takes place. Once the user inputs their data through the UI, this data is sent to the Prediction Module. Here, the input data is processed and fed into a pre-trained Random Forest model, which has been saved in a pickle file. This model was trained on a comprehensive dataset and is capable of predicting the likelihood of various diseases based on the input features. The output of this module includes the predicted disease(s) and associated probability scores, indicating the confidence level of each prediction. The use of a pickle file allows for efficient loading and utilization of the model, ensuring quick and accurate predictions.

8.2.1 ALGORITHM:

Random Forest:

One of the most effective ensemble classification approaches is the random forest algorithm Many decision trees make up RF. Each decision tree provides a vote that indicates the object's class decision. In a random forest, there are three major tuning factors. The number of trees (n tree), the minimum node size, and the number of characteristics used to divide each node for each tree (m try). The benefits of the random forest algorithm are described below.

- Random forest algorithm is accurate ensemble learning algorithm.
- Random forest runs efficiently for large data sets.
- It can handle hundreds of input variables.
- Random forest estimates which variables are important in classification.
- It can handle missing data.
- Random forest has methods for balancing error for class unbalanced data sets.
- Generated forests in this method can be saved for future reference.
- Random forest overcomes the problem over fitting.
- In training data, RF is less sensitive to outlier.
- In RF, parameters can be set easily and eliminates the need for tree pruning.
- In RF accuracy and variable importance is automatically generated. Randomization is used to choose the optimal node to split on when building individual trees in random forest. This value is A, where A denotes the number 10 of characteristics in the data collection. However, RF will build a large number of noisy trees, lowering classification accuracy and leading to incorrect decisions for fresh samples.

Random forest Algorithm:

Step 1: Select random K data points from the training set.
Step 2: Build the decision trees associated with the selected data points
Step 3: Choose the number N for decision trees that you want to build.
Step 4: Repeat Step 1 and 2.

Step 5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

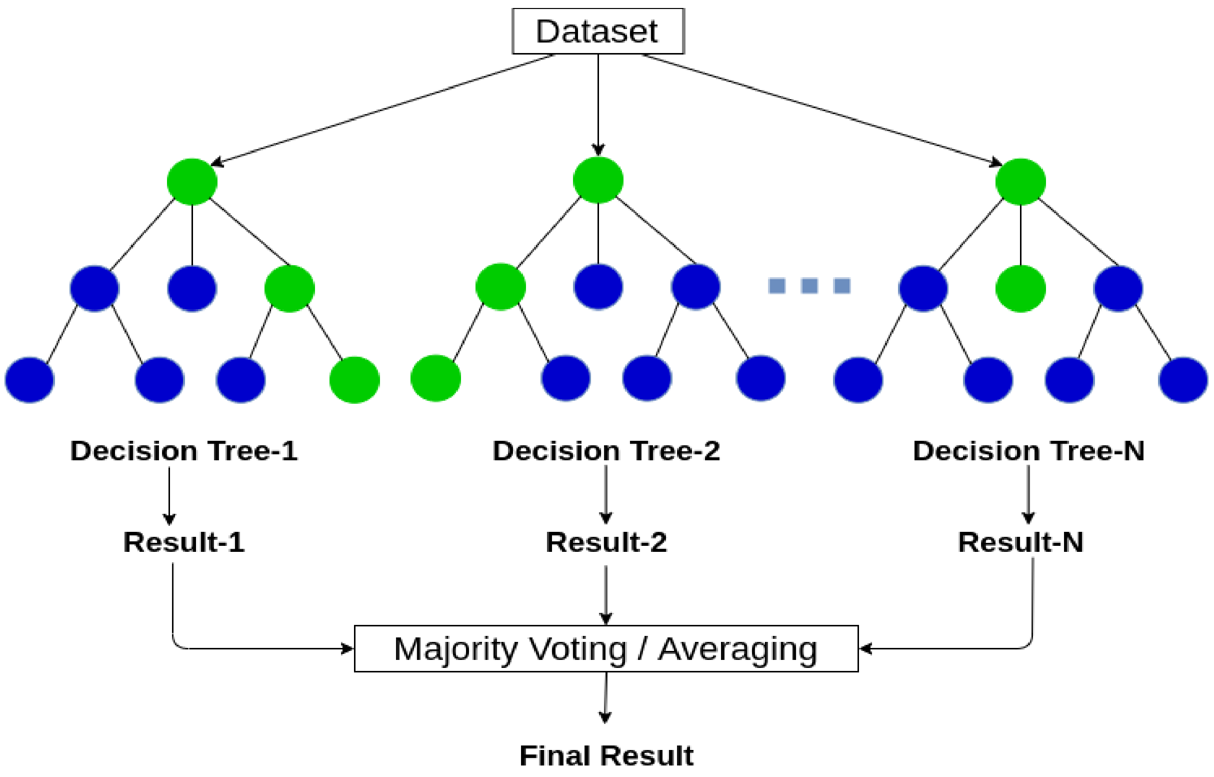


Figure 8.1 Random Forest Illustration.

How Does Random Forest Work?

The random Forest algorithm works in several steps which are discussed below–>

- Ensemble of Decision Trees: Random Forest leverages the power of ensemble learnind by constructing an army of decision trees. These trees are like individual experts, each specializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.
- Random Feature Selection: To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random Feature Selection. During the training of each tree,

a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.

- Bootstrap Aggregating or Bagging: The technique of bagging is a cornerstone of Random Forest's training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.
- Decision Making and Voting: When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the mode (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process.

Key Features of Random Forest:

1. High Predictive Accuracy: Imagine Random Forest as a team of decision-making wizards. Each wizard (decision tree) looks at a part of the problem, and together, they weave their insights into a powerful prediction tapestry. This teamwork often results in a more accurate model than what a single wizard could achieve.
2. Resistance to Overfitting: Random Forest is like a cool-headed mentor guiding its apprentices (decision trees). Instead of letting each apprentice memorize every detail of their training, it encourages a more well-rounded understanding. This approach helps prevent getting too caught up with the training data which makes the model less prone to overfitting.
3. Large Datasets Handling: Dealing with a mountain of data? Random Forest tackles it like a seasoned explorer with a team of helpers (decision trees). Each helper takes on a part of the dataset, ensuring that the expedition is not only thorough but also surprisingly quick.

4. Variable Importance Assessment: Think of Random Forest as a detective at a crime scene, figuring out which clues (features) matter the most. It assesses the importance of each clue in solving the case, helping you focus on the key elements that drive predictions.
5. Built-in Cross-Validation: Random Forest is like having a personal coach that keeps you in check. As it trains each decision tree, it also sets aside a secret group of cases (out-of-bag) for testing. This built-in validation ensures your model doesn't just ace the training but also performs well on new challenges.
6. Handling Missing Values: Life is full of uncertainties, just like datasets with missing values. Random Forest is the friend who adapts to the situation, making predictions using the information available. It doesn't get flustered by missing pieces; instead, it focuses on what it can confidently tell us.
7. Parallelization for Speed: Random Forest is your time-saving buddy. Picture each decision tree as a worker tackling a piece of a puzzle simultaneously. This parallel approach taps into the power of modern tech, making the whole process faster and more efficient for handling large-scale projects.

CHAPTER 9

SOFTWARE ENVIRONMENT

9.1 PYTHON:

What is Python? Executive Summary.

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

DJANGO:

What is Django?

Django is a Python-based web framework which allows you to quickly create web application without all of the installation or dependency problems that you normally will find with other frameworks.

When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use.

Why Django?

- It's very easy to switch database in Django framework.
- It has built-in admin interface which makes easy to work with it.
- Django is fully functional framework that requires nothing else.
- It has thousands of additional packages available.
- It is very scalable.

Popularity of Django:

Django is used in many popular sites like as: Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic etc. There are more than 5k online sites based on the Django framework. (Source)

Sites like Hot Frameworks assess the popularity of a framework by counting the number of GitHub projects and StackOverflow questions for each platform, here Django is in 6th position. Web frameworks often refer to themselves as "opinionated" or "un-opinionated" based on opinions about the right way to handle any particular task. Django is somewhat opinionated, hence delivers the in both worlds (opinionated & un-opinionated).

Features of Django:

- Versatility of Django

  Django can build almost any type of website. It can also work with any client-side framework and can deliver content in any format such as HTML, JSON, XML etc. Some sites which can be built using Django are wikis, social networks, new sites etc

- Security

  Since Django framework is made for making web development easy, it has been engineered in such a way that it automatically do the right things to protect the website. For example, In the Django framework instead of putting a password in cookies, the hashed password is stored in it so that it can't be fetched easily by hackers.

- Scalability

  Django web nodes have no stored state, they scale horizontally – just fire up more of then when you need them. Being able to do this is the essence of good scalability. Instagram and Disqus are two Django based products that have millions of active users, this is taken as an example of the scalability of Django.

- Portability

  All the codes of the Django framework are written in Python, which runs on many platforms. Which leads to run Django too in many platforms such as Linux, Windows and Mac OS.

About SQLite:

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen().

Libraries:

Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Numpy:

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionality. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

- Using NumPy, a developer can perform the following operations −
- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Seaborn:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on top matplotlib library and is also closely integrated with the data structures from pandas. Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs so that we can switch between different visual representations for the same variables for a better understanding of the dataset.

Scikit – learn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

9.2 SOURCE CODE:

EDA for diabetes disease dataset:

D_EDA.ipynb:

```
#code for Diabetes Dataset cleaning and processing
#importing libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
D_data=pd.read_csv('datasets/ diabetes.csv')#loading dataset
D_data.info()
D_data.head()
#data processing
D_data['Gender']=D_data['Gender'].map({'Male': 1, 'Female': 0})
D_data['Polyuria']=D_data['Polyuria'].map({'Yes': 1, 'No': 0})
D_data['Polydipsia']=D_data['Polydipsia'].map({'Yes': 1, 'No': 0})
D_data['Sudden weight loss']=D_data['Sudden weight loss'].map({'Yes': 1, 'No': 0})
D_data['Weakness']=D_data['Weakness'].map({'Yes': 1, 'No': 0})
D_data['Polyphagia']=D_data['Polyphagia'].map({'Yes': 1, 'No': 0})
D_data['Genital thrush']=D_data['Genital thrush'].map({'Yes': 1, 'No': 0})
```

```
D_data['Visual blurring']=D_data['Visual blurring'].map({'Yes': 1, 'No':
0})
D_data['Itching']=D_data['Itching'].map({'Yes': 1, 'No': 0})
D_data['Irritability']=D_data['Irritability'].map({'Yes': 1, 'No': 0})
D_data['Delayed healing']=D_data['Delayed healing'].map({'Yes': 1,
'No': 0})
D_data['Partial paresis']=D_data['Partial paresis'].map({'Yes': 1, 'No':
0})
D_data['Muscle stiffness']=D_data['Muscle stiffness'].map({'Yes': 1,
'No': 0})
D_data['Alopecia']=D_data['Alopecia'].map({'Yes': 1, 'No': 0})
D_data['Obesity']=D_data['Obesity'].map({'Yes': 1, 'No': 0})
D_data['Class']=D_data['Class'].map({'Positive': 1, 'Negative': 0})
D_data.info()
D_data.shape
D_data.isnull().sum()#data cleaning
D_data.dtypes
D_data.describe()
print(D_data['Age'].unique())
D_data.drop_duplicates(inplace=True)#droping duplicates
# Histogram for numerical features
import matplotlib.pyplot as plt
import seaborn as sns
D_data['Age'].hist(bins=20)
plt.show()
D_data['Age'].value_counts().plot(kind='bar')
```

```
plt.show()
#Distribution plot for a numerical column
sns.histplot(D_data['Age'], kde=True)
plt.show()
#Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(D_data.corr(), annot=True, fmt='.2f')
plt.show()
D_data.plot.scatter(x='Age', y= 'Class')
plt.show()
sns.pairplot(D_data)
plt.show()
D_data.groupby('Age')['Class'].value_counts().unstack().plot(kind='bar'
, stacked=True)
plt.show()
D_data.groupby('Gender')['Class'].value_counts().unstack().plot(kind='
bar', stacked=True)
plt.show()
for i in D_data.columns:
    D_data[i].value_counts().plot(kind = "pie", figsize = (6,6), autopct =
"%1.1f%%")
    plt.xticks(rotation = 45)
    plt.show()
#storeing processed data to a csv file
D_data.to_csv('D_data.csv', index=False)
```

Diabetes.ipynb:

```
#code for Diabetes model creation and saving

#importing libraries

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.ensemble import RandomForestClassifier

data= pd.read_csv('datasets/D_data.csv') )#loading dataset

data.info()

data.head())

#Randomize the dataset

data = data.sample(frac=1).reset_index(drop=True

data.head()

X=data.drop(columns='Class',axis=1)

print(data.shape)

Y=data['Class']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,
stratify=Y, random_state=42)#Spliting dataset to training and testing

model_RFC = RandomForestClassifier(n_estimators=50)#model loading

model_RFC.fit(X_train, Y_train)#model training

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

y_pred = model_RFC.predict(X_test)#Testing model with testdata
```

```
accuracy = accuracy_score(Y_test, y_pred)#accuracy
precision = precision_score(Y_test, y_pred, average='weighted')
recall = recall_score(Y_test, y_pred, average='weighted')
f1 = f1_score(Y_test, y_pred, average='weighted')
cm = confusion_matrix(Y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
print(f'Confusion Matrix:\n{cm}')
import pickle
filename = "diabetes_model.pkl"
#Storing model in pickle fiie
pickle.dump(model_RFC, open(filename, "wb"))
```

LungCancer.ipynb:

```
#code for Diabetes model creation and saving
#importing libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
data= pd.read_csv('datasets/LC_data.csv') #loading dataset
data.info()
data.head()

X=data.drop(columns='LUNG_CANCER',axis=1)

Y=data['LUNG_CANCER']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,
stratify=Y, random_state=42) )#Spliting dataset to training and testing

model_RFC = RandomForestClassifier(n_estimators=100) #model loading

model_RFC.fit(X_train, Y_train)#model training

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

y_pred = model_RFC.predict(X_test)#Testing model with testdata

accuracy = accuracy_score(Y_test, y_pred)

precision = precision_score(Y_test, y_pred, average='weighted')

recall = recall_score(Y_test, y_pred, average='weighted')

f1 = f1_score(Y_test, y_pred, average='weighted')

cm = confusion_matrix(Y_test, y_pred)

print(f'Accuracy: {accuracy}')

print(f'Precision: {precision}')

print(f'Recall: {recall}')

print(f'F1 Score: {f1}')

print(f'Confusion Matrix:\n{cm}')

import pickle

filename = "LungCancer_model.pkl"

pickle.dump(model_RFC, open(filename, "wb"))
```

Home page:

Home.html:

```
<!-- home.html -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Home Page</title>

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/home.css' %}">

</head>

<body>

    <div class="container">

        <h1>Welcome to Multiple Disease Prediction</h1>

        <p>

                At HealthPredict, we leverage cutting-edge machine learning
technology to provide predictive insights into your health. Our platform is
designed to empower users with early detection and risk assessment for a
range of medical conditions, including heart disease, lung cancer, diabetes,
and Parkinson's disease. By inputting basic health and lifestyle information,
our advanced models analyze your data to provide personalized predictions,
helping you take proactive steps towards better health.

        </p>

        <p>
```

```
            Whether you're a health enthusiast, a patient looking for more
information, or simply curious about your well-being, HealthPredict offers
a user-friendly and secure platform for you to explore your health profile.
Our commitment to data privacy and security ensures that your information
is always protected.

        </p>

        <p>

           Join us on the journey to better health and peace of mind. Discover
what HealthPredict can do for you today!

        </p>

    </div>

    <form action="{% url 'home' %}">

    <ul>

        <li><a href="{% url 'diabetes' %}">Diabetes Prediction</a></li>

              <li><a href="{% url 'heart_prediction' %}">Heart Disease
Prediction</a></li>

                 <li><a href="{% url 'lung_cancer' %}">Lung Cancer
Prediction</a></li>

                    <li><a href="{% url 'parkinsons' %}">Parkinson's
Prediction</a></li>

    </ul>

</form>

</body>

</html>
```

heart_prediction.html

```
<!-- heartprediction.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Heart Disease Prediction</title>
    {% load static %}
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
    <style>
        .Input{
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .description{
            margin: 20px;
            padding: 10px;
            background-color: #fff;
            border-radius: 8px;
        }
    </style>
</head>
<body style="display: block;">
    <div class="Input">
        <h1>Heart Disease Prediction:</h1>
       <form method="post" action="{% url 'model_heart_prediction' %}">
           {% csrf_token %}
           <!-- Form fields for input -->
           <label for="Age">Age:</label>
           <input type="text" id="Age" name="Age" required><br><br>
               <label  for="Sex">Gender:(Enter  0  for  Female  and  1  for
male)</label>
           <input type="text" id="Sex" name="Sex" required><br><br>
           <label for="CP">CP:</label>
           <input type="text" id="CP" name="CP" required><br><br>
           <label for="Trestbps">Trestbps:</label>
                   <input   type="text"   id="Trestbps"   name="Trestbps"
required><br><br>
           <label for="Chol">Chol:</label>
           <input type="text" id="Chol" name="Chol" required><br><br>
           <label for="Fbs">Fbs:</label>
           <input type="text" id="Fbs" name="Fbs" required><br><br>
           <label for="Restecg">Restecg:</label>
```

```
                    <input type="text" id="Restecg" name="Restecg"
required><br><br>
            <label for="Thalach">Thalach:</label>
                    <input type="text" id="Thalach" name="Thalach"
required><br><br>
            <label for="Exang">Exang:</label>
                     <input type="text" id="Exang" name="Exang"
required><br><br>
            <label for="Oldpeak">Oldpeak:</label>
              <input type="text" step="0.1" id="Oldpeak" name="Oldpeak"
required><br><br>
            <label for="Slope">Slope:</label>
                    <input type="number" id="Slope" name="Slope"
required><br><br>
            <label for="Ca">Ca:</label>
            <input type="text" id="Ca" name="Ca" required><br><br>
            <label for="Thal">Thal:</label>
            <input type="text" id="Thal" name="Thal" required><br><br>
            <input type="submit" value="Predict">
        </form>
    </div>
    <div class="description" style="display: block;">
        NOTE: For input details:<br>
         ->cp (Chest Pain Type): The type of chest pain experienced, often
classified into several categories, :<br>
```

0: Typical angina<br>

1: Atypical angina<br>

2: Non-anginal pain<br>

->trestbps (Resting Blood Pressure): The resting blood pressure (in mm Hg) of the individual, typically measured at the hospital.

<br>

->chol (Serum Cholesterol): The individual's serum cholesterol level in mg/dL.

<br>

->fbs (Fasting Blood Sugar): Indicates whether the fasting blood sugar is greater than 120 mg/dL (1 = true; 0 = false).

<br>

->restecg (Resting Electrocardiographic Results): Results of the resting electrocardiogram, which can include:

<br>

0: Normal<br>

1: Having ST-T wave abnormality (e.g., T wave inversions and/or ST elevation or depression of >0.05 mV)<br>

2: Showing probable or definite left ventricular hypertrophy according to Estes' criteria<br>

->thalach (Maximum Heart Rate Achieved): The maximum heart rate achieved during a stress test.<br>

->exang (Exercise Induced Angina): Indicates whether the individual experienced angina induced by exercise (1 = yes; 0 = no).

<br>

```
->oldpeak (ST Depression Induced by Exercise Relative to Rest): This
measures the amount of depression of the ST segment during exercise
relative to rest, <br>which can indicate the severity of coronary artery
disease.

<br>

->slope (Slope of the Peak Exercise ST Segment): The slope of the peak
exercise ST segment, with possible values:

<br>

0: Upsloping<br>

1: Flat<br>

2: Downsloping<br>

->ca (Number of Major Vessels Colored by Fluoroscopy): The number of
major vessels (0-3) colored by fluoroscopy.<br>

->thal (Thalassemia): A blood disorder with the following possible
values:<br>

0: Normal<br>

1: Fixed defect<br>

2: Reversible defect<br>

    <div>

</body>

</html>
```

prediction_result.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Heart Disease Prediction Result</title>
  {% load static %}
    <link rel="stylesheet" href="{% static 'css/styles.css' %}"> <!--
Assuming you have a CSS file in static folder -->
</head>
<body>
  <div class="container">
    <h1>Heart Disease Prediction Result</h1>
    <div class="result">
      {% if prediction == 1 %}
         <p>The model predicts that there is a risk of heart disease.</p>
      {% else %}
          <p>The model predicts that there is no significant risk of heart
disease.</p>
      {% endif %}
    </div>
    <div class="details">
      <h2>Input Details:</h2>
      <ul>
```

```
            <li>Age: {{ data.Age }}</li>

            <li>Sex: {{ data.Sex }}</li>

            <li>Chest Pain Type (CP): {{ data.CP }}</li>

           <li>Resting Blood Pressure (Trestbps): {{ data.Trestbps }}</li>

            <li>Cholesterol (Chol): {{ data.Chol }}</li>

            <li>Fasting Blood Sugar (Fbs): {{ data.Fbs }}</li>

                    <li>Resting  Electrocardiographic  Results  (Restecg):  {{
data.Restecg }}</li>

            <li>Maximum Heart Rate Achieved (Thalach): {{ data.Thalach
}}</li>

            <li>Exercise Induced Angina (Exang): {{ data.Exang }}</li>

                   <li>ST  Depression  Induced  by  Exercise  (Oldpeak):  {{
data.Oldpeak }}</li>

                 <li>Slope  of  the  Peak  Exercise  ST  Segment  (Slope):  {{
data.Slope }}</li>

            <li>Number of Major Vessels (Ca): {{ data.Ca }}</li>

            <li>Thalassemia (Thal): {{ data.Thal }}</li>

        </ul>

      </div>

      <a href="{% url 'heart_prediction' %}">Try Again</a>

  </div>

</body>

</html>
```

views.py:

```
from django.shortcuts import render

from django.views import View

from django.http import HttpResponse

from .models import  SignUpRecord ,LoginRecord ,Heart ,
LungCacerRecord , DiabetesRecord , ParkinsonsRecord

import pickle

import os

from django.contrib.auth.hashers import make_password

# Load the model once, outside the view class to avoid loading it multiple
times

MODEL_PATH1=os.path.join(os.path.dirname('D:/shiva/MDP_MajorProject
/'), 'heart_disease_model.pkl')

MODEL_PATH2=os.path.join(os.path.dirname('D:/shiva/MDP_MajorProject
/'), 'LungCancer_model.pkl')

MODEL_PATH3=os.path.join(os.path.dirname('D:/shiva/MDP_MajorProject
/'), 'diabetes_model.pkl')

MODEL_PATH4=os.path.join(os.path.dirname('D:/shiva/MDP_MajorProject
/'), 'parkinsons_model.pkl')

heart_disease_model = pickle.load(open(MODEL_PATH1, "rb"))

LungCancer_model = pickle.load(open(MODEL_PATH2, "rb"))

diabetes_model = pickle.load(open(MODEL_PATH3, "rb"))

parkinsons_model = pickle.load(open(MODEL_PATH4, "rb"))

class SignUp(View):

    def get(self,r):
        return render(r,template_name='index.html')
class Sign_Up(View):
    def post(self,r):
        email=r.POST['email']
        password=r.POST['password']
        cpassword=r.POST['cpassword']
        if SignUpRecord.objects.filter(email=email).exists():
            return HttpResponse("Account already exists please login by typing
'login' in place of 'sign=up' in url")
        if password != cpassword:
                return HttpResponse("Password and Confirm-Password are not
matching")
        hashed_password = make_password(password)
        s1 = SignUpRecord(email=email, password=hashed_password)
         s1=SignUpRecord(email=email,password=password,cpassword=cpa
ssword)
        s1.save()
          return HttpResponse("SignUp Successful. Pleasse login by typing
'login' in place of 'sign=up' in url")
class Login(View):
    def get(self,r):
        return render(r,template_name='login.html')
class Log_in(View):
    def post(self,r):
        email=r.POST['email']

        password=r.POST['password']

        try:

            if SignUpRecord.objects.get(email=email, password=password):

                return render(r, 'home.html')

        except SignUpRecord.DoesNotExist:

            return HttpResponse("Email or Password are Incorrect")

class Home(View):

    def get(self,r):

        return render(r,template_name='home.html')

class HeartPrediction(View):

    def get(self, request):

        return render(request, template_name='heart_prediction.html')

class ModelHeartPrediction(View):

    def post(self, request):

        try:

            age = int(request.POST['Age'])

            sex = int(request.POST['Sex'])

            cp = int(request.POST['CP'])

            trestbps = int(request.POST['Trestbps'])

            chol = int(request.POST['Chol'])

            fbs = int(request.POST['Fbs'])

            restecg = int(request.POST['Restecg'])
            thalach = int(request.POST['Thalach'])
            exang = int(request.POST['Exang'])
            oldpeak = float(request.POST['Oldpeak'])
            slope = int(request.POST['Slope'])
            ca = int(request.POST['Ca'])
            thal = int(request.POST['Thal'])
           h = Heart(age=age, sex=sex, cp=cp, trestbps=trestbps, chol=chol, fbs=fbs, restecg=restecg, thalach=thalach, exang=exang, oldpeak=oldpeak, slope=slope, ca=ca, thal=thal)
            h.save()
            # Making prediction
            prediction = heart_disease_model.predict([[age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]])
            # You can return the prediction result in the context if you render a template
            context = {
                'prediction': prediction[0],  # Assuming the model returns a list
                'data': request.POST,
            }
            return render(request, 'prediction_result1.html', context)
        except Exception as e:
            return HttpResponse(f"An error occurred: {str(e)}")
class LungCancer(View):
    def get(self,request):
        return render(request,template_name='lung_cancer.html')
class ModelLungCancer(View):
    def post(self,request):
        gender = int(request.POST['gender'])
        age = int(request.POST['age'])
        smoking = int(request.POST['smoking'])
        yellow_fingers = int(request.POST['yellow_fingers'])
        anxiety = int(request.POST['anxiety'])
        peer_pressure = int(request.POST['peer_pressure'])
        chronic_disease = int(request.POST['chronic_disease'])
        fatigue = int(request.POST['fatigue'])
        allergy = int(request.POST['allergy'])
        wheezing = int(request.POST['wheezing'])
        alcohol_consuming = int(request.POST['alcohol_consuming'])
        coughing = int(request.POST['coughing'])
        shortness_of_breath = int(request.POST['shortness_of_breath'])
        swallowing_difficulty = int(request.POST['swallowing_difficulty'])
        chest_pain = int(request.POST['chest_pain'])
        # Create HealthRecord object and save to database
        s2=LungCacerRecord(
            GENDER=gender,
            AGE=age,
            SMOKING=smoking,
            YELLOW_FINGERS=yellow_fingers,
            ANXIETY=anxiety,
            PEER_PRESSURE=peer_pressure,
            CHRONIC_DISEASE=chronic_disease,
            FATIGUE=fatigue,
            ALLERGY=allergy,
            WHEEZING=wheezing,
            ALCOHOL_CONSUMING=alcohol_consuming,
            COUGHING=coughing,
            SHORTNESS_OF_BREATH=shortness_of_breath,
            SWALLOWING_DIFFICULTY=swallowing_difficulty,
            CHEST_PAIN=chest_pain
        )
        s2.save()
        prediction =
LungCancer_model.predict([[gender,age,smoking,yellow_fingers,anxiety,p
eer_pressure,chronic_disease,fatigue,
        allergy,wheezing,alcohol_consuming,coughing,shortness_of_breath,
swallowing_difficulty,chest_pain]])
            # You can return the prediction result in the context if you render
a template
        context = {
        'prediction': prediction[0],  # Assuming the model returns a list
        'data': request.POST,
        }
        return render(request, 'prediction_result2.html', context)

#----------------------------------------------------------------
class Diabetes(View):
    def get(self,request):
        return render(request,template_name='Diabetes.html')
class ModelDiabetes(View):
    def post(self, request):
        # Retrieve data from form
        age = int(request.POST['Age'])
        gender = int(request.POST['Gender'])
        polyuria = int(request.POST['Polyuria'])
        polydipsia = int(request.POST['Polydipsia'])
        sudden_weight_loss = int(request.POST['Sudden_weight_loss'])
        weakness = int(request.POST['Weakness'])
        polyphagia = int(request.POST['Polyphagia'])
        genital_thrush = int(request.POST['Genital_thrush'])
        visual_blurring = int(request.POST['Visual_blurring'])
        itching = int(request.POST['Itching'])
        irritability = int(request.POST['Irritability'])
        delayed_healing = int(request.POST['Delayed_healing'])
        partial_paresis = int(request.POST['Partial_paresis'])
        muscle_stiffness = int(request.POST['Muscle_stiffness'])
        alopecia = int(request.POST['Alopecia'])

        obesity = int(request.POST['Obesity'])

        # Create DiabetesRecord object and save to database

        diabetes_record = DiabetesRecord(

            age=age,

            gender=gender,

            polyuria=polyuria,

            polydipsia=polydipsia,

            sudden_weight_loss=sudden_weight_loss,

            weakness=weakness,

            polyphagia=polyphagia,

            genital_thrush=genital_thrush,

            visual_blurring=visual_blurring,

            itching=itching,

            irritability=irritability,

            delayed_healing=delayed_healing,

            partial_paresis=partial_paresis,

            muscle_stiffness=muscle_stiffness,

            alopecia=alopecia,

            obesity=obesity

        )

        diabetes_record.save()

                prediction = diabetes_model.predict([[age,gender, polyuria,
polydipsia, sudden_weight_loss, weakness,
                            polyphagia, genital_thrush, visual_blurring, itching,
irritability,
                            delayed_healing, partial_paresis, muscle_stiffness,
alopecia, obesity]])

            # You can return the prediction result in the context if you render
a template
        context = {
        'prediction': prediction[0],  # Assuming the model returns a list
        'data': request.POST,
        }
        return render(request, 'prediction_result3.html', context)
#--------------------------------------------------
class Parkinsons(View):
    def get(self,request):
        return render(request,template_name='Parkinsons.html')
class ModelParkinsons(View):
    def post(self,request):
        MDVP_Fo= float(request.POST['MDVP_Fo'])
        MDVP_Fhi= float(request.POST['MDVP_Fhi'])
        MDVP_Flo= float(request.POST['MDVP_Flo'])
        MDVP_Jitter_Percent= float(request.POST['MDVP_Jitter_Percent'])
        MDVP_Jitter_Abs= float(request.POST['MDVP_Jitter_Abs'])
        MDVP_RAP= float(request.POST['MDVP_RAP'])
        MDVP_PPQ= float(request.POST['MDVP_PPQ'])
        Jitter_DDP= float(request.POST['Jitter_DDP'])
        MDVP_Shim= float(request.POST['MDVP_Shim'])
        MDVP_Shim_dB= float(request.POST['MDVP_Shim_dB'])
        Shimmer_APQ3= float(request.POST['Shimmer_APQ3'])
        Shimmer_APQ5= float(request.POST['Shimmer_APQ5'])
        MDVP_APQ= float(request.POST['MDVP_APQ'])
        Shimmer_DDA= float(request.POST['Shimmer_DDA'])
        NHR= float(request.POST['NHR'])
        HNR= float(request.POST['HNR'])
        RPDE= float(request.POST['RPDE'])
        DFA= float(request.POST['DFA'])
        spread1= float(request.POST['spread1'])
        spread2= float(request.POST['spread2'])
        D2= float(request.POST['D2'])
        PPE= float(request.POST['PPE'])
        # Create HealthRecord object and save to database
         P_record=ParkinsonsRecord(MDVP_Fo=MDVP_Fo,MDVP_Fhi=MDVP_
Fhi,MDVP_Flo=MDVP_Flo,MDVP_Jitter_Percent=MDVP_Jitter_Percent,
            MDVP_Jitter_Abs=MDVP_Jitter_Abs,MDVP_RAP=MDVP_RAP,MDVP
_PPQ=MDVP_PPQ,Jitter_DDP=Jitter_DDP,MDVP_Shim=MDVP_Shim,
            MDVP_Shim_dB=MDVP_Shim_dB,Shimmer_APQ3=Shimmer_APQ
3,Shimmer_APQ5=Shimmer_APQ5,MDVP_APQ=MDVP_APQ,
            Shimmer_DDA=Shimmer_DDA,NHR=NHR,HNR=HNR,RPDE=RPDE
,DFA=DFA,spread1=spread1,
            spread2=spread2,D2=D2,PPE=PPE
        )
        P_record.save()
 prediction=parkinsons_model.predict([[MDVP_Fo,MDVP_Fhi,MDVP_Flo,M
DVP_Jitter_Percent,MDVP_Jitter_Abs,MDVP_RAP,MDVP_PPQ,Jitter_DDP,M
DVP_Shim, MDVP_Shim_dB,Shimmer_APQ3,Shimmer_APQ5,MDVP_APQ,
Shimmer_DDA,NHR,HNR,RPDE,DFA,spread1,spread2,D2,PPE]])
            # You can return the prediction result in the context if you render
a template
        context = {
        'prediction': prediction[0],  # Assuming the model returns a list
        'data': request.POST,
        }
        return render(request, 'prediction_result4.html', context)
```

CHAPTER 10

RESULT/DISCUSSIONS

10.1 SYSTEM TESTING:

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

TYPES OF TESTING :

There are many types of testing methods are available in that mainly used testing methods are as follows

Unit testing*:*

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

Integration testing:

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

Functional test:

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

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

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

System Test*:*

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

White Box Testing:

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

Black Box Testing:

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

10.1.1Test cases:

Tested	Test name	Inputs	Expected output	Actual Output	Result
1	load dataset	dataset	dataset loaded	successfully loaded	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
2	Splitting dataset into training and validation set	dataset	spitted to training and validation	successfully spitted to train data and validation data	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
3	Model creation	Random Forest from sklearn module	model created	successfully model created	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
4	training set evaluation	train data	training done	successfully trained	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
5	Validating test dataset	test data	test dataset validated	successfully validated	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
6	getting accuracy	test dataset	accuracy	successfully got accuracy	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
7	Save model to pickle file	model	Store in a pickle file	successfully loaded	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
8	Register/SignUp	Email, password , confirm password	SignUp	Successfully Register/SignUp	Pas success

Tested	Test name	Inputs	Expected output	Actual Output	Result
9	Login	Email, password	Login	Successfully Login	Pas success

Tested	Test name	Inputs	Expected output	Actual Output	Result
10	Predict disease	Related to disease	Having or not having disease	Successfully predicted	Pas success

Table 10.1: Test Cases

Accuracy Metrices of models:

Models	**SVM**	**Logistic Regression**	**Random forest**
Diabetes	80%	80%	80%
Heart Disease	78.3%	76.6%	88.3%
Lung Cancer	89%	90.9%	92.7%
Parkinson	89.4%	92.1%	94.7%

Table 10.2: Accuracy of models

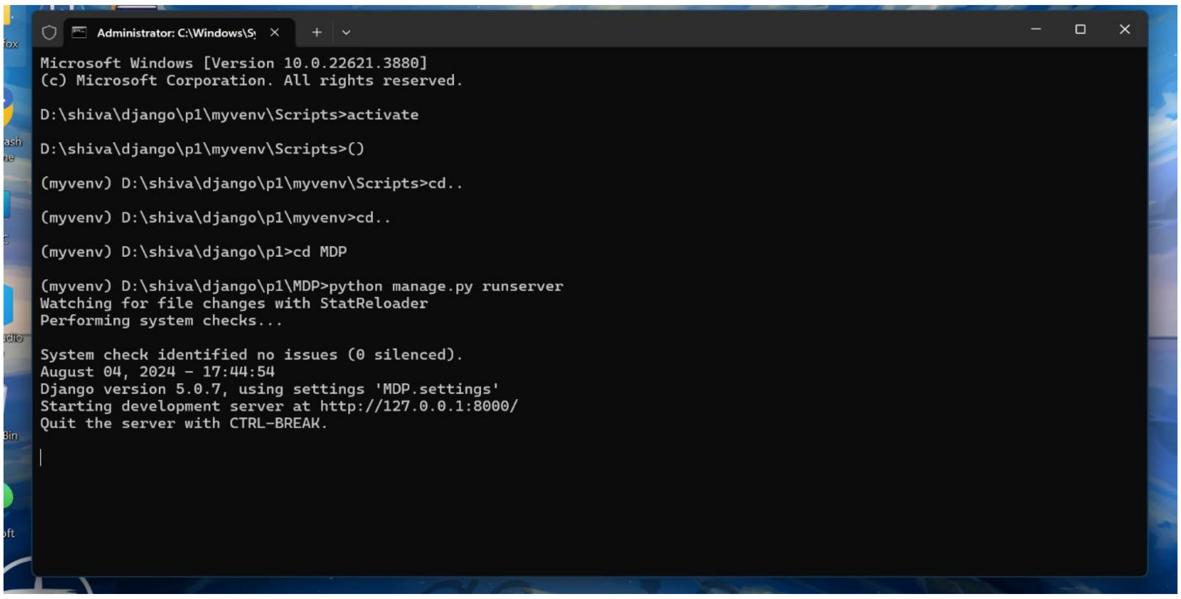
Accuracy Metrices of RFC model:

Models	**Accuracy**	**Precision**	**Recall**	**F1 score**
Diabetes	96%	96%	96%	95.9%
Heart Disease	88.3%	88.3%	88.3%	88.3%
Lung Cancer	92.7%	93.2%	92.7%	91.4%
Parkinson	94.7%	94.7%	94.7%	94.7%

Table 10.2: Accuracy Metrices of RFC

10.2 SCREENSHOTS:

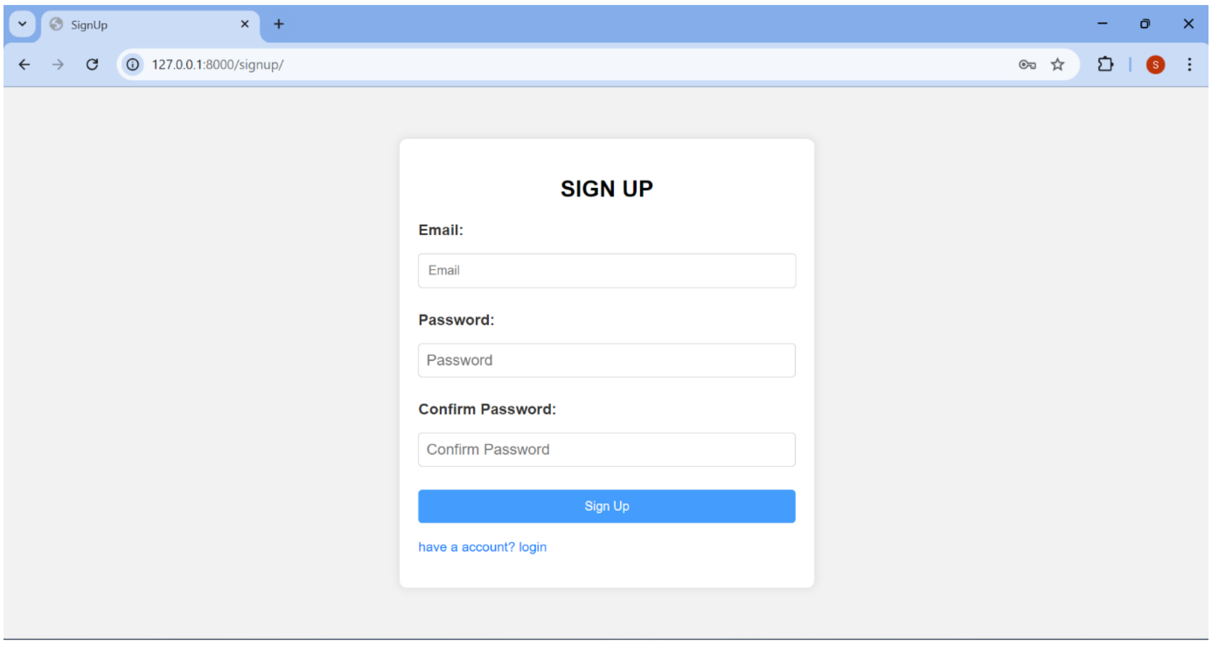
Runing Django local server.



```
Microsoft Windows [Version 10.0.22621.3880]
(c) Microsoft Corporation. All rights reserved.

D:\shiva\django\p1\myvenv\Scripts>activate

D:\shiva\django\p1\myvenv\Scripts>()

(myvenv) D:\shiva\django\p1\myvenv\Scripts>cd..

(myvenv) D:\shiva\django\p1\myvenv>cd..

(myvenv) D:\shiva\django\p1>cd MDP

(myvenv) D:\shiva\django\p1\MDP>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 04, 2024 - 17:44:54
Django version 5.0.7, using settings 'MDP.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

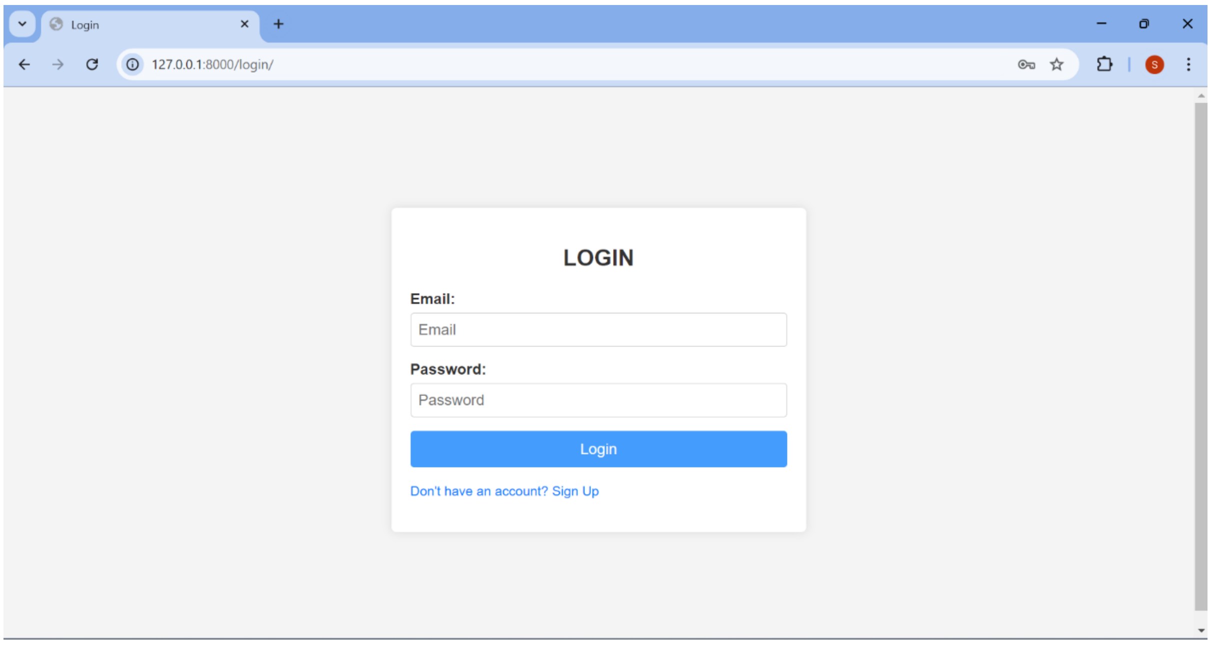
SignUp page:

Here user has to register/signup with entering email Id and password.



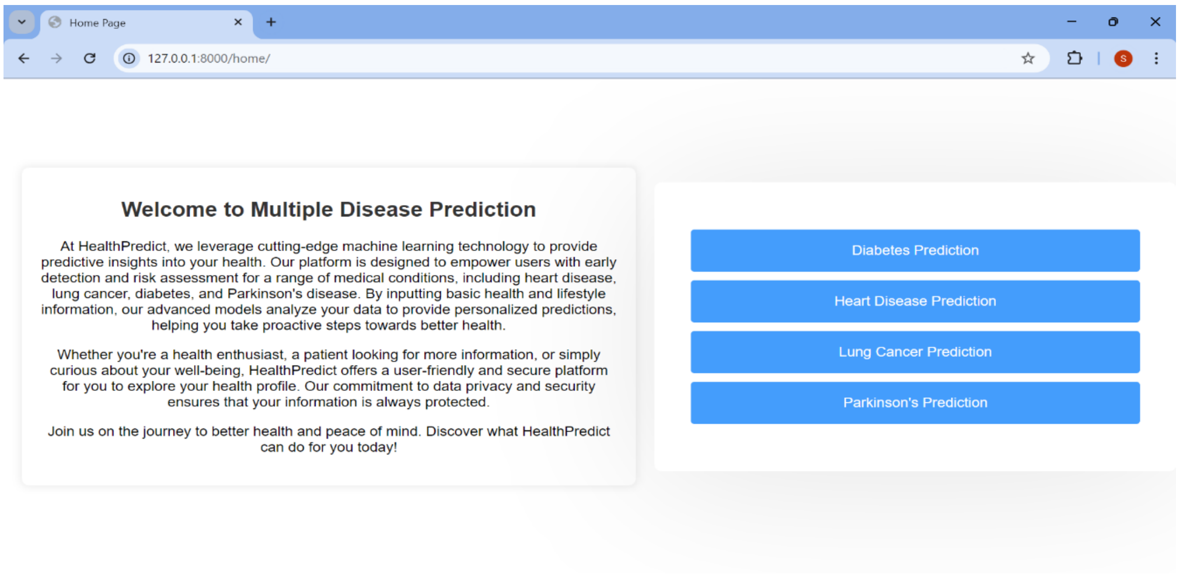
Login page:

Here user can login with email and password.



Home page:

Here user can read about the Site and will select the disease he want to predict.



Welcome to Multiple Disease Prediction

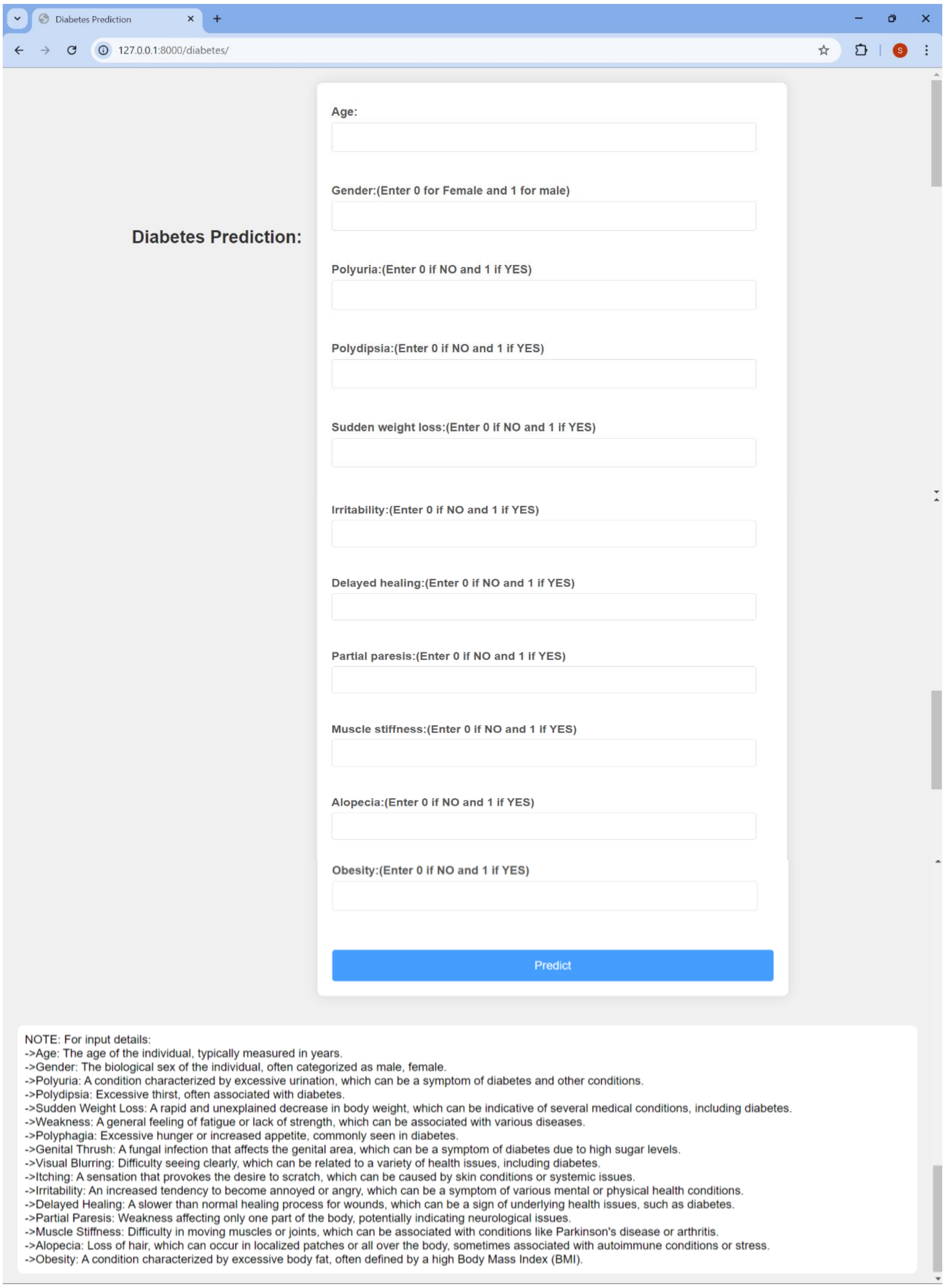
At HealthPredict, we leverage cutting-edge machine learning technology to provide predictive insights into your health. Our platform is designed to empower users with early detection and risk assessment for a range of medical conditions, including heart disease, lung cancer, diabetes, and Parkinson's disease. By inputting basic health and lifestyle information, our advanced models analyze your data to provide personalized predictions, helping you take proactive steps towards better health.

Whether you're a health enthusiast, a patient looking for more information, or simply curious about your well-being, HealthPredict offers a user-friendly and secure platform for you to explore your health profile. Our commitment to data privacy and security ensures that your information is always protected.

Join us on the journey to better health and peace of mind. Discover what HealthPredict can do for you today!

Diabetes Prediction Page:

Here User will enter the details required for predicting the disease.



Diabetes Prediction:

Age:

Gender:(Enter 0 for Female and 1 for male)

Polyuria:(Enter 0 if NO and 1 if YES)

Polydipsia:(Enter 0 if NO and 1 if YES)

Sudden weight loss:(Enter 0 if NO and 1 if YES)

Irritability:(Enter 0 if NO and 1 if YES)

Delayed healing:(Enter 0 if NO and 1 if YES)

Partial paresis:(Enter 0 if NO and 1 if YES)

Muscle stiffness:(Enter 0 if NO and 1 if YES)

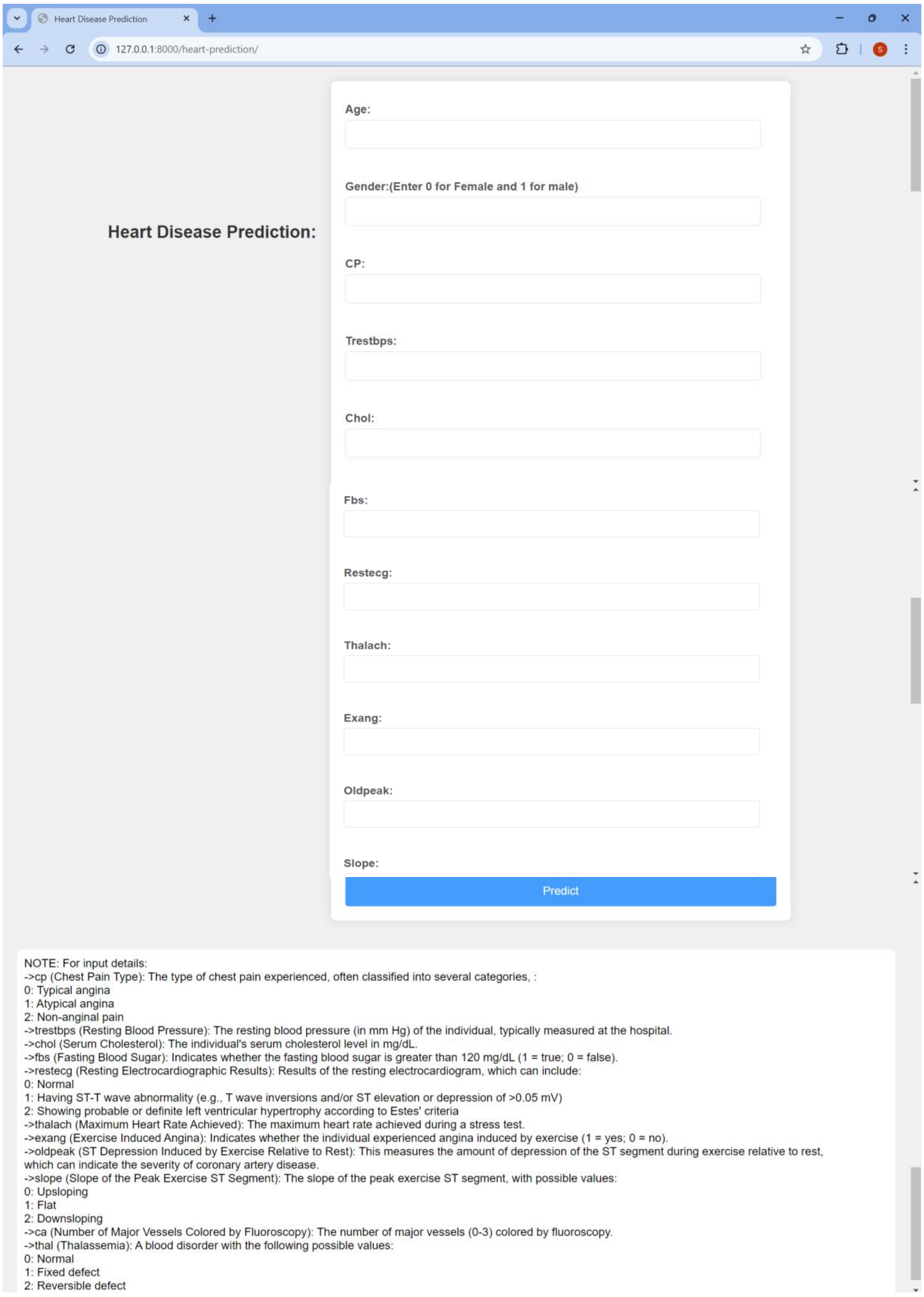
Alopecia:(Enter 0 if NO and 1 if YES)

Obesity:(Enter 0 if NO and 1 if YES)

NOTE: For input details:
->Age: The age of the individual, typically measured in years.
->Gender: The biological sex of the individual, often categorized as male, female.
->Polyuria: A condition characterized by excessive urination, which can be a symptom of diabetes and other conditions.
->Polydipsia: Excessive thirst, often associated with diabetes.
->Sudden Weight Loss: A rapid and unexplained decrease in body weight, which can be indicative of several medical conditions, including diabetes.
->Weakness: A general feeling of fatigue or lack of strength, which can be associated with various diseases.
->Polyphagia: Excessive hunger or increased appetite, commonly seen in diabetes.
->Genital Thrush: A fungal infection that affects the genital area, which can be a symptom of diabetes due to high sugar levels.
->Visual Blurring: Difficulty seeing clearly, which can be related to a variety of health issues, including diabetes.
->Itching: A sensation that provokes the desire to scratch, which can be caused by skin conditions or systemic issues.
->Irritability: An increased tendency to become annoyed or angry, which can be a symptom of various mental or physical health conditions.
->Delayed Healing: A slower than normal healing process for wounds, which can be a sign of underlying health issues, such as diabetes.
->Partial Paresis: Weakness affecting only one part of the body, potentially indicating neurological issues.
->Muscle Stiffness: Difficulty in moving muscles or joints, which can be associated with conditions like Parkinson's disease or arthritis.
->Alopecia: Loss of hair, which can occur in localized patches or all over the body, sometimes associated with autoimmune conditions or stress.
->Obesity: A condition characterized by excessive body fat, often defined by a high Body Mass Index (BMI).

Heart Prediction Page:

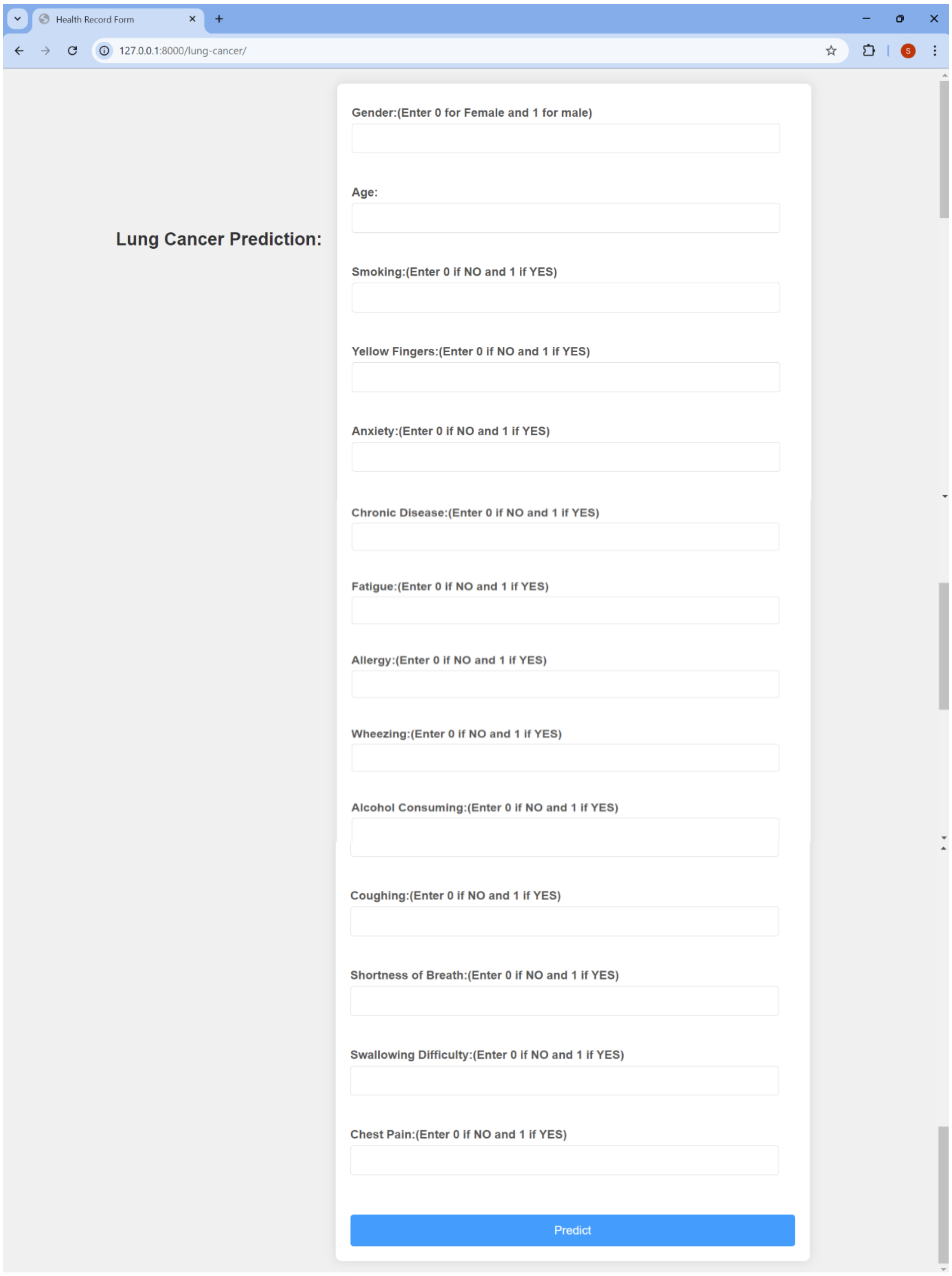
Here User will enter the details required for predicting the disease.


Heart Disease Prediction
127.0.0.1:8000/heart-prediction/
Heart Disease Prediction:
Age:
Gender:(Enter 0 for Female and 1 for male)
CP:
Trestbps:
Chol:
Fbs:
Restecg:
Thalach:
Exang:
Oldpeak:
Slope:
Predict

NOTE: For input details:
->cp (Chest Pain Type): The type of chest pain experienced, often classified into several categories, :
0: Typical angina
1: Atypical angina
2: Non-anginal pain
->trestbps (Resting Blood Pressure): The resting blood pressure (in mm Hg) of the individual, typically measured at the hospital.
->chol (Serum Cholesterol): The individual's serum cholesterol level in mg/dL.
->fbs (Fasting Blood Sugar): Indicates whether the fasting blood sugar is greater than 120 mg/dL (1 = true; 0 = false).
->restecg (Resting Electrocardiographic Results): Results of the resting electrocardiogram, which can include:
0: Normal
1: Having ST-T wave abnormality (e.g., T wave inversions and/or ST elevation or depression of >0.05 mV)
2: Showing probable or definite left ventricular hypertrophy according to Estes' criteria
->thalach (Maximum Heart Rate Achieved): The maximum heart rate achieved during a stress test.
->exang (Exercise Induced Angina): Indicates whether the individual experienced angina induced by exercise (1 = yes; 0 = no).
->oldpeak (ST Depression Induced by Exercise Relative to Rest): This measures the amount of depression of the ST segment during exercise relative to rest, which can indicate the severity of coronary artery disease.
->slope (Slope of the Peak Exercise ST Segment): The slope of the peak exercise ST segment, with possible values:
0: Upsloping
1: Flat
2: Downsloping
->ca (Number of Major Vessels Colored by Fluoroscopy): The number of major vessels (0-3) colored by fluoroscopy.
->thal (Thalassemia): A blood disorder with the following possible values:
0: Normal
1: Fixed defect
2: Reversible defect

Lung Cancer Prediction Page:

Here User will enter the details required for predicting the disease.



Health Record Form

127.0.0.1:8000/lung-cancer/

Lung Cancer Prediction:

Gender:(Enter 0 for Female and 1 for male)

Age:

Smoking:(Enter 0 if NO and 1 if YES)

Yellow Fingers:(Enter 0 if NO and 1 if YES)

Anxiety:(Enter 0 if NO and 1 if YES)

Chronic Disease:(Enter 0 if NO and 1 if YES)

Fatigue:(Enter 0 if NO and 1 if YES)

Allergy:(Enter 0 if NO and 1 if YES)

Wheezing:(Enter 0 if NO and 1 if YES)

Alcohol Consuming:(Enter 0 if NO and 1 if YES)

Coughing:(Enter 0 if NO and 1 if YES)

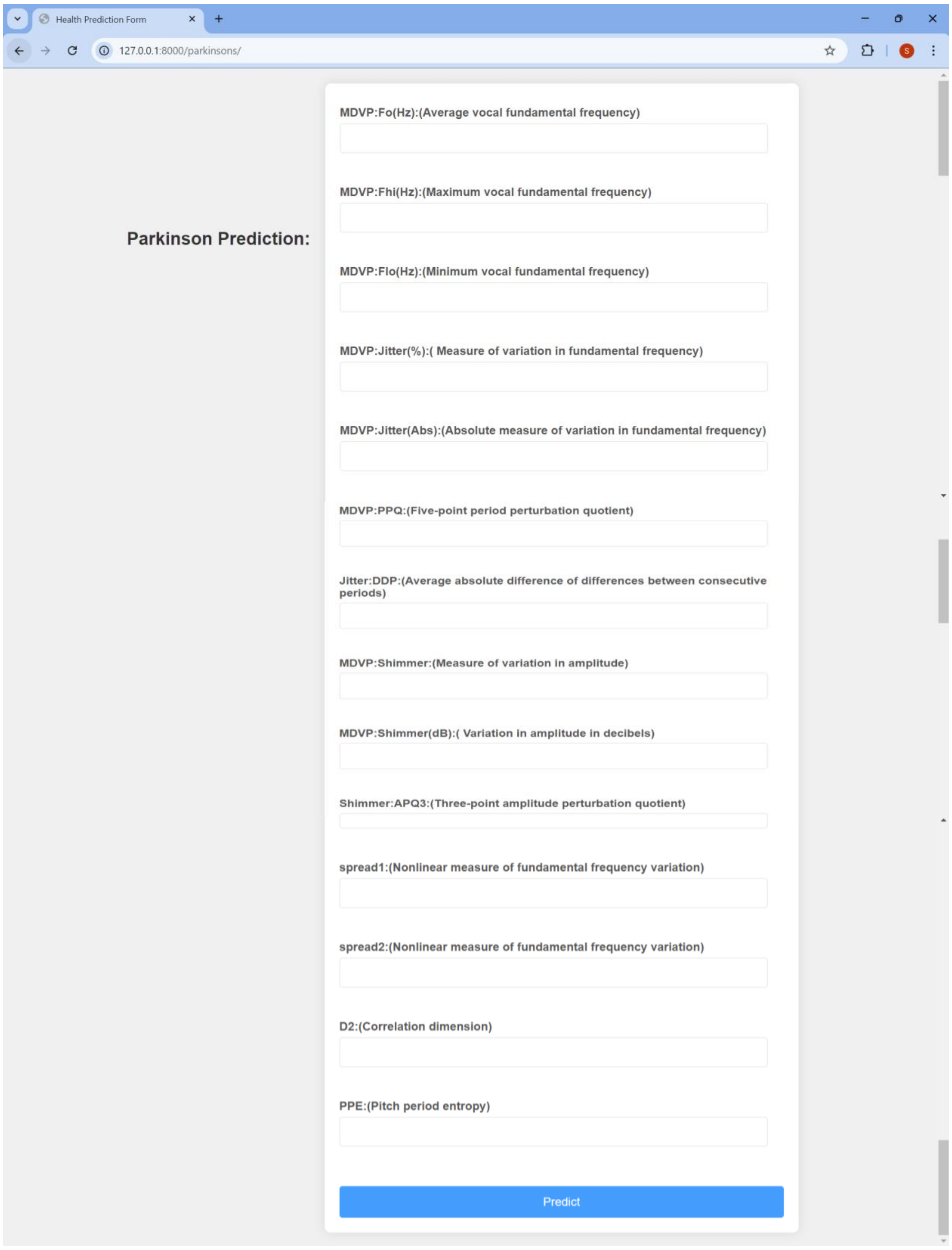
Shortness of Breath:(Enter 0 if NO and 1 if YES)

Swallowing Difficulty:(Enter 0 if NO and 1 if YES)

Chest Pain:(Enter 0 if NO and 1 if YES)

Parkinson Prediction Page:

Here User will enter the details required for predicting the disease.



Parkinson Prediction:

MDVP:Fo(Hz):(Average vocal fundamental frequency)

MDVP:Fhi(Hz):(Maximum vocal fundamental frequency)

MDVP:Flo(Hz):(Minimum vocal fundamental frequency)

MDVP:Jitter(%):(Measure of variation in fundamental frequency)

MDVP:Jitter(Abs):(Absolute measure of variation in fundamental frequency)

MDVP:PPQ:(Five-point period perturbation quotient)

Jitter:DDP:(Average absolute difference of differences between consecutive periods)

MDVP:Shimmer:(Measure of variation in amplitude)

MDVP:Shimmer(dB):(Variation in amplitude in decibels)

Shimmer:APQ3:(Three-point amplitude perturbation quotient)

spread1:(Nonlinear measure of fundamental frequency variation)

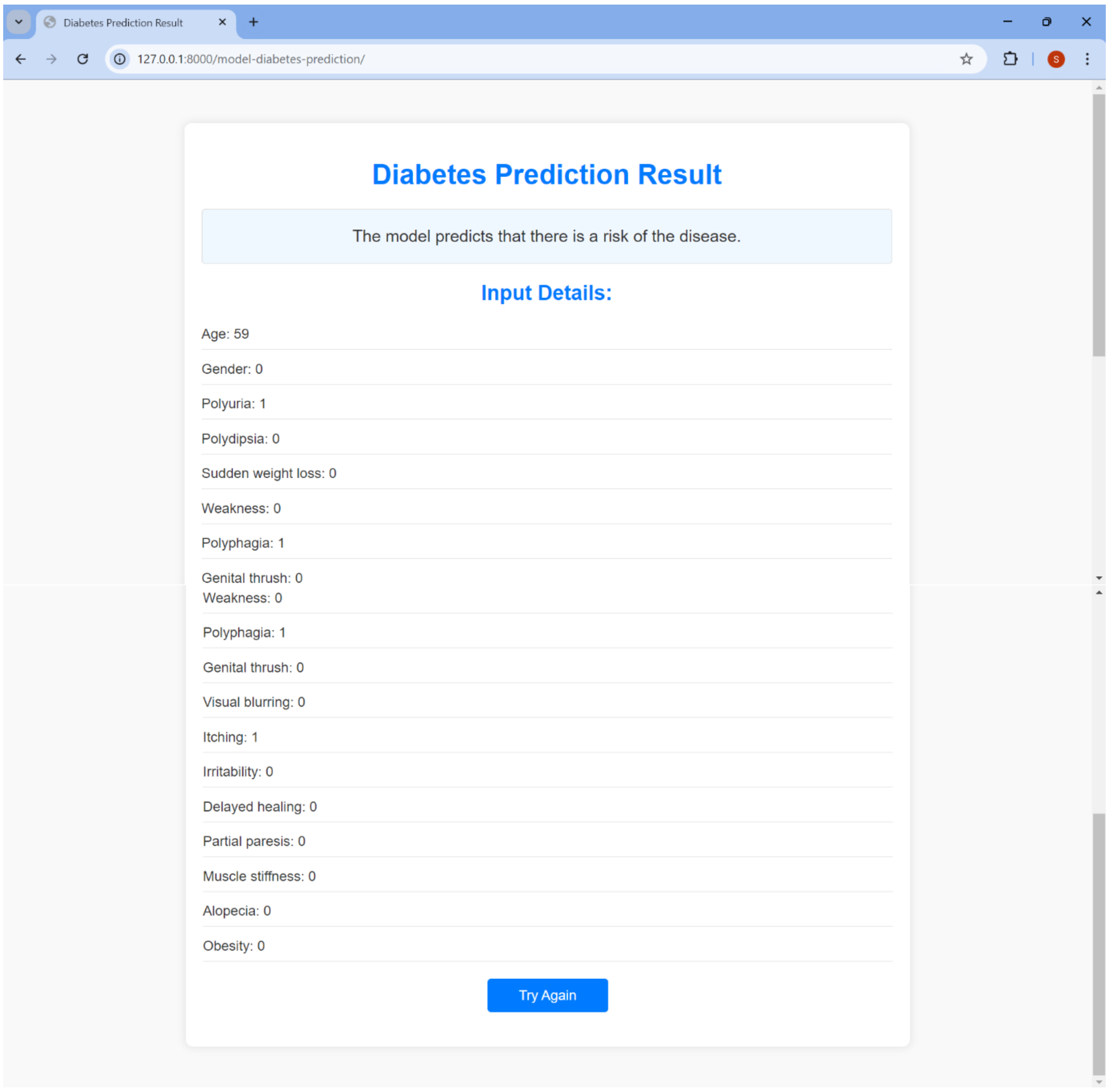
spread2:(Nonlinear measure of fundamental frequency variation)

D2:(Correlation dimension)

PPE:(Pitch period entropy)

Diabetes Prediction Result Page:

Here User will enter the details required for predicting the disease.



Diabetes Prediction Result
127.0.0.1:8000/model-diabetes-prediction/

Diabetes Prediction Result

The model predicts that there is a risk of the disease.

Input Details:

Age: 59

Gender: 0

Polyuria: 1

Polydipsia: 0

Sudden weight loss: 0

Weakness: 0

Polyphagia: 1

Genital thrush: 0

Weakness: 0

Polyphagia: 1

Genital thrush: 0

Visual blurring: 0

Itching: 1

Irritability: 0

Delayed healing: 0

Partial paresis: 0

Muscle stiffness: 0

Alopecia: 0

Obesity: 0

CHAPTER 11

Conclusion

The "Classification of Multiple Diseases by applying technique of Random Forest" project has demonstrated the potential of machine learning to revolutionize healthcare diagnostics. By utilizing the Random Forest algorithm, the project successfully classifies multiple diseases, including Diabetes, Heart Disease, Lung Cancer, and Parkinson's Disease, based on an individual's medical data. The system's accuracy and reliability in predicting these conditions provide a valuable tool for early diagnosis, which is crucial for timely medical intervention and improved patient outcomes. The project integrates a user-friendly web interface developed using Django, allowing easy data entry and immediate access to prediction results. The system's ability to present results in an understandable format, including visualizations and detailed reports, further enhances its usability and effectiveness. This integration ensures that both healthcare professionals and patients can leverage the system's capabilities without requiring advanced technical knowledge.Overall, the project showcases a practical application of machine learning in healthcare, highlighting its ability to process complex datasets, uncover hidden patterns, and provide actionable insights. It marks a significant step towards personalized medicine, where diagnostic tools can be tailored to an individual's unique health profile.

Future Enhancements:

1. **Expansion of Training Data**: To further improve the model's accuracy and generalizability, it is crucial to expand the training dataset. Including a more extensive and diverse range of data can help the model learn from a broader spectrum of medical conditions and patient demographics. This can also aid in reducing biases and improving the model's performance across different population groups.

2. **Addition of More Diseases**: Future iterations of the project can include the prediction of additional diseases. Expanding the scope to cover a broader range of medical conditions would increase the system's value as a comprehensive diagnostic tool. This could involve incorporating data for conditions like Alzheimer's disease, various cancers, autoimmune disorders, and infectious diseases, among others.
3. **Enhancement of the User Interface**: Improving the user interface can significantly enhance user experience. This includes optimizing the design for mobile devices, making the interface more intuitive, and incorporating interactive elements that guide users through the data entry process. Additionally, providing detailed explanations and guidance on interpreting the results can help users better understand the predictions and their implications.
4. **Improving Model Accuracy and Robustness**: Efforts can be made to enhance the accuracy and robustness of the model. This could involve exploring advanced machine learning techniques, such as deep learning, ensemble learning, or even hybrid models that combine multiple algorithms. Fine-tuning hyperparameters and utilizing techniques like cross-validation can also improve model performance.
5. **Integration with Electronic Health Records (EHR)**: Integrating the system with EHRs can streamline data input and provide real-time updates to patient records. This integration would enable seamless data flow between the diagnostic tool and healthcare providers, facilitating more informed clinical decisions and personalized treatment plans.

6. **Real-Time Monitoring and Alerts**: For conditions that require continuous monitoring, implementing real-time data collection and alert systems could be highly beneficial. For instance, the system could monitor patients with chronic conditions and send alerts to healthcare providers if there are significant changes in the patient's condition, allowing for timely intervention.
7. **Data Privacy and Security Enhancements**: As the system deals with sensitive medical data, ensuring robust data privacy and security measures is paramount. Future enhancements could include implementing advanced encryption methods, secure data storage solutions, and compliance with relevant healthcare regulations and standards, such as HIPAA.
8. **Research and Development of Predictive Models for Rare Diseases**: Exploring the development of predictive models for rare diseases can fill an existing gap in healthcare diagnostics. These conditions often lack extensive datasets, and developing models to predict them could provide invaluable support to healthcare providers and patients.
9. **User Personalization and Adaptation**: The system could be enhanced to offer personalized recommendations and insights based on a user's unique medical history and risk factors. This could include lifestyle recommendations, preventive measures, and personalized monitoring plans.

CHAPTER 12:REFERENCES

[1] Umair Muneer Butt,Sukumar Letchmunan, Mubashir Ali, Fadratul Hafinaz Hassan, Anees Baqir, and Hafiz Husnain Raza Sherazi,”Machine Learning Based Diabetes Classification and Prediction for Healthcare Applications”.

[2] Ahmad Ayid Ahmad,Huseyin Polat, Yanwu Xu. ”Prediction of Heart Disease Based on Machine Learning Using Jellyfish Optimization Algorithm “.

[3] Tehnan I. A. Mohamed, Absalom El-Shamir Ezugwu ,”Enhancing Lung Cancer Classification And Prediction With Deep Learning And Multi-Omics Data”.

[4] Sura Mahmood Abdullah, Thekra Abbas, Munzir Hubiba Bashir, Ishfaq Ahmad Khaja, Musheer Ahmad, Naglaa F.Soliman, Walid El-Shafai, “Deep Transfer Learning Based Parkinson’s Disease Detection Using Optimized Feature Selection”.

[5] Gayathri Devi Nagalapuram, Varshashree Dasuratha,Vansika Singh,

” Literature Survey for Lung Cancer Analysis and Prediction”.

[6] H. Al-Mallah, et al., "Multiple Disease Prediction Using Hybrid Deep Learning Architecture," 2016.

[7] Y. Gamo, et al., "Machine Learning Based Clinical Decision Support Systems for Multi-Disease Prediction: A Review," 2020.

[8] W. Li, et al., "Towards Multi-Disease Prediction Using Graph Neural Networks," 2020. [8] R. Ribeiro, et al., "A Survey on Explainable AI Techniques for Diagnosis and Prognosis in Healthcare," 2020.

[9] E. Char, et al., "Ethical Considerations in AI-Driven Healthcare," 2020. M. Wild, et al., "Streamlit for Machine Learning: Creating Interactive Web Apps in Python," 2022