CHAPTER 1

CHAPTER 1
INTRODUCTION

Healthcare plays a vital role in our well-being, and in our ever-changing world, it's essential to be attentive and take early action to address potential health issues. Detecting and preventing diseases early on is crucial for a healthier society. Thanks to advancements in technology and data analysis, we now have powerful tools that can help us predict the risk of various health problems. This project focuses on using a strong statistical method known as logistic regression to forecast the likelihood of developing different diseases, with a special focus on heart diseases.

Heart Diseases and Logistic Regression

Logistic regression, a statistical tool, is central in predicting the chances of events like heart diseases happening. It acts like an educated estimate by considering factors like age, cholesterol levels, and lifestyle to gauge the probability of a heart condition. This equips healthcare professionals and individuals with valuable information to take early steps to reduce the risk. Logistic regression goes beyond heart diseases; it's versatile in foreseeing various health issues and understanding how different factors affect whether a health problem is likely or not. In our age of advancing technology and data analysis, logistic regression empowers healthcare with evidence-based decision-making. It improves our ability to predict and prevent diseases, ultimately leading to a healthier and more productive society through early action and risk assessment.

1.1 Problem Statement

In the era of big data and advanced analytics, healthcare organizations and medical professionals are inundated with vast amounts of patient data from various sources, such as electronic health records (EHRs), wearable devices, and medical imaging. The challenge lies in harnessing this wealth of data to predict

the onset or progression of multiple diseases with accuracy, efficiency, and in a timely manner. This project aims to address the pressing need for developing a comprehensive data-driven solution for multiple disease prediction, leveraging the power of data science and machine learning.

1.2 Project Scope and Objectives

The multi-disease prediction system is a versatile and user-friendly platform designed to anticipate a range of health conditions by integrating data from diverse sources. With its intuitive interface, real-time monitoring capabilities, and adaptability to the dynamic landscape of healthcare, this system provides a holistic approach to disease risk assessment. It caters to the needs of both healthcare professionals and individuals, offering valuable insights into potential health issues. By harnessing data from various origins, the system empowers users with the ability to proactively manage their well-being and make informed decisions about their health. In an era marked by evolving healthcare demands, this system stands as a comprehensive tool, bridging the gap between medical expertise and individual health awareness, ultimately contributing to better health outcomes for all.

The scope of the project *MediForecast – Enhancing Healthcare with AI and Streamlit* is to build an intelligent, user-friendly system capable of predicting multiple diseases such as diabetes, heart disease, and Parkinson’s using machine learning techniques. This system aims to assist both medical professionals and individuals by providing early warnings based on clinical parameters, potentially improving diagnosis speed and treatment outcomes. It is designed to be accessible through a simple web interface built with Streamlit, making it usable in both urban and rural settings, especially where medical expertise is limited. The project is scalable and can be extended to include more diseases or advanced models in the future.

The objectives of the project include collecting and preprocessing high-quality datasets for each disease, training and evaluating machine learning models like Logistic Regression, Random Forest, and Support Vector Machine, and integrating them into a unified system. Another key goal is to design a clean, interactive web application using Streamlit, where users can input relevant health metrics and receive real-time predictions. The project also focuses on deployment, aiming to host the system on a cloud platform to ensure easy and secure access. Long-term objectives involve improving prediction accuracy, maintaining data privacy, and expanding the system's functionality to cover a wider range of diseases and potentially integrate with electronic health records.

1.3 Literature survey

[1] This paper is developed by JIAN PING LI the development of an efficient machine learning-based diagnosis system for heart disease. The study employs various machine learning classifiers, including Logistic Regression (LR), K-Nearest Neighbors (K-NN), Artificial Neural Networks (ANN), Support Vector Machines (SVM), Naive Bayes (NB), and Decision Trees (DT). The primary dataset used for testing the system is the Cleveland heart disease dataset. Performance evaluation metrics are employed to assess the system's effectiveness. The study reveals several key findings:

The specificity of the ANN classifier is highest when combined with the Relief feature selection algorithm, making it the most accurate system for detecting healthy individuals.

The classifier Naive Bayes (NB) performs best in terms of sensitivity when used with features selected by the LASSO feature selection algorithm, outperforming other combinations.

Support Vector Machine (SVM) with the proposed feature selection algorithm (FCMIM) achieves an accuracy of 92.37%, outperforming previously proposed

methods and demonstrating the efficacy of this machine learning-based approach for heart disease detection.

The study's novelty lies in the development of a diagnosis system for heart disease using a combination of various classifiers, feature selection algorithms, and the LOSO cross-validation method. This approach aims to identify the most relevant features for diagnosis, reduce computation time, and enhance classification accuracy. The study concludes by suggesting future directions, including exploring additional feature selection algorithms and optimization methods to further enhance the predictive system's performance for heart disease diagnosis.

[2] Vijeta Sharma, from Banaras Hindu University in India, conducted a research project using computer algorithms like Naive Bayes, Random Forest Classification, and Decision Tree. These algorithms help with tasks like medical diagnostics. They used a software tool called WEKA, which is free to use, for their analysis.

The advantage of their approach is that WEKA simplifies the process and is open-source, making it accessible to many researchers. However, there is a downside to this method. When dealing with vast amounts of medical data and a multitude of features, WEKA can slow down. This means that in some cases, making predictions or diagnoses can become challenging.

In the field of medical diagnostics, it's crucial to have efficient tools and methods because timely and accurate predictions can be a matter of life and death. The research by Vijeta Sharma offers insights into how these commonly used algorithms and WEKA can be valuable, but it also highlights the need for improvements in handling large and complex medical datasets. By addressing the challenge of handling extensive data efficiently, researchers can make more precise predictions and diagnoses, ultimately benefiting patients and healthcare professionals.

[3] Sridevi S, from CMR Institute of Technology in Bengaluru, India, conducted a research paper addressing the challenges posed by a shortage of medical staff and an increasing number of patients. The primary goal of this study was to create a linear regression model for predicting heart diseases by simplifying the features used in the analysis. To achieve this, they employed an open-source dataset from UCI and applied a technique called kernel PCA to reduce the number of features. Remarkably, their efforts resulted in a 100% accuracy rate in predicting heart diseases.

However, there is a notable drawback to their work. The dataset they used for their research consisted of only 303 patient records. This limited dataset size raises concerns because it may not be sufficient to predict various types of contemporary cardiovascular diseases effectively. In the field of medical diagnostics, it's crucial to have robust and reliable models, especially when dealing with the evolving landscape of cardiovascular health issues. While achieving 100% accuracy is impressive, the real-world applicability and generalizability of the model could be uncertain due to the dataset's size and scope. Therefore, expanding the dataset and testing the model on a broader range of cases would be essential to validate its effectiveness for a more comprehensive array of cardiovascular conditions.

[4] Aditi Gavhane, from Sardar Patel Institute of Technology in Mumbai, India, has put forth a paper focusing on a medical diagnostic model. The study uses fundamental patient information like age, sex, blood pressure, heart rate, and high cholesterol levels to construct this model. They utilize a Multi-layer Perceptron, a type of artificial neural network, to ensure accurate results.

One of the primary benefits of their approach is the incorporation of modern technologies like CAD (Computer-Aided Design) to display predictive outcomes. CAD tools are known for their ability to assist in visualizing and interpreting complex data, which can be highly advantageous in the medical field for

understanding diagnostic results.

However, it's important to note a limitation of this study. The model is built using only a few basic patient features. While this simplicity may contribute to straightforward and efficient analysis, it might not capture the full complexity of medical conditions. In the realm of medical diagnostics, conditions can often be multifaceted and influenced by numerous factors. Relying solely on a handful of basic features might lead to limited or incomplete results.

To enhance the effectiveness of the diagnostic model, it could be beneficial to consider additional and more comprehensive patient data. Expanding the scope of features might lead to a more accurate and nuanced understanding of medical conditions, ultimately improving patient care and diagnostic outcomes.

[5] Saba Bashir, hailing from the Computer Science Department at the Federal Urdu University of Arts, Science & Technology in Islamabad, Pakistan, has conducted a research study centered on medical diagnostics. In this study, an open-source dataset from UCI has been employed as the foundation for their analysis. Multiple algorithms, including Logistic Regression, Decision Tree, Support Vector Machine (SVM), and Random Forest, have been implemented to scrutinize the data. One significant advantage of this research lies in the comparative approach. By employing various algorithms on the same dataset, the study provides valuable insights into how different models perform in the context of medical diagnostics. This comparative analysis can help researchers and healthcare professionals make informed decisions about which algorithm is best suited for specific diagnostic tasks.

Nevertheless, there is a notable limitation to consider. The dataset used in this study is open-source and, in all likelihood, contains older and generalized heart-related data. This might not be ideal for addressing contemporary or more specific cardiovascular issues. The landscape of medical diagnostics is constantly

evolving, with new challenges and conditions emerging over time. Relying solely on older and general data might not adequately address the intricacies of modern healthcare problems.

For more effective and precise diagnostic models, it may be beneficial to incorporate newer and more specialized datasets, reflecting the latest medical insights and conditions.

[6] This paper is done by Akkem Yaganteeswarudu, Infoshare Systems, Pyramid Softsol Pvt Limited, Hyderabad, India. In the provided text showcases a approach to building a multi-disease prediction model by leveraging machine learning and deep learning techniques. The primary focus areas of this study encompass diabetes analysis, heart disease prediction, and cancer detection, each employing distinct algorithms for analysis.

For diabetes analysis, logistic regression demonstrated a commendable accuracy rate of 92%. In the realm of heart disease prediction, the Random Forest algorithm exhibited a robust performance, achieving a high accuracy of 95%. Additionally, for the critical task of cancer detection, the Support Vector Machine (SVM) algorithm displayed an impressive accuracy level of 96%.

Furthermore, the study delved into the domain of diabetes retinopathy analysis, which involves the examination of retina images. In this context, the researchers harnessed the Python TensorFlow library and Convolutional Neural Networks (CNN) to construct and test models for image analysis. The resulting model yielded an accuracy rate of 91%.

To ensure the preservation of the model's behavior and effectiveness, the researchers employed Python pickling. This technique allowed for the serialization and de-serialization of Python object structures, enabling the storage of critical model information. The pickled Python object could be saved on disk as a character stream, encapsulating all the necessary data for reconstructing the

model in another script.

[7] Dr. P.Hamsagayathri, Assistant Professor, Dept of ECE, Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamil Nadu, India. The provided text highlights various machine learning techniques and their application in medical diagnosis, particularly in the context of disease prediction and classification. The study focuses on the performance of Support Vector Machine (SVM) and Naive Bayes, along with some other methods.

SVM emerges as a strong contender, boasting an impressive accuracy of 94.60%. SVM is recognized for its versatile performance across various applications. Naive Bayes, on the other hand, enhances classification efficiency by removing irrelevant features. It demonstrates productivity, especially in terms of reduced computing time. However, it demands a substantial amount of training data, which can be a drawback.

In contrast, the WEKA tool is often used for differentiation, but its efficiency is comparatively lower, achieving only 70% split accuracy, in contrast to Naive Bayes. Researchers have also explored SVM variants, such as SMO, to detect valuable characteristics. These variations exhibit accuracies ranging from 84.5% to 86.41%. Integrating genetic algorithms (GA) and fuzzy logic further elevates classification accuracy to 87%, with lower costs incurred. A hybrid approach combining multiple techniques achieves precision rates of 84.07% and 76.26% for heart disease and diabetes diagnosis, respectively.

For diagnosing diabetes, the Naive Bayes-based approach has shown promise. Naive Bayes achieved a remarkable accuracy rate of 96%. This approach appears highly efficient in producing accurate predictions with minimal error. However, it's essential to note that precision dipped to 79.58%, signaling the need for further research and refined training data to enhance its diagnostic capabilities.

[8] Wenqi Li, School of Computer Science and Technology, Donghua University, Shanghai,China.

This study, titled "Coronary Heart Disease Prediction Based on Combined Reinforcement Multitask Progressive Networks," was presented at the 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). It introduces a novel deep learning framework that combines reinforcement learning with multitask progressive networks to predict coronary heart disease.

The model is trained on diverse patient health records and learns progressively over tasks, allowing it to refine predictions for heart disease based on related subtasks or auxiliary health indicators. The technique aims to improve generalization and predictive performance over traditional machine learning methods.

The study demonstrates that leveraging shared representations through multitask learning and enhancing them with reinforcement learning improves model accuracy, adaptability, and convergence speed. This approach is particularly valuable in dynamic medical environments where patient data is heterogeneous and continuously evolving.

[9] Nikhila, PG Student, Department of Electronics and Communication Systems, Centre for PG Studies, Mysuru, Karnataka, India.

The paper titled "Chronic Kidney Disease Prediction using Machine Learning Ensemble Algorithm" was presented at the 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS). This study explores the use of ensemble learning techniques to predict chronic kidney disease (CKD).

The model integrates multiple classifiers such as Decision Trees, Random Forest, and Gradient Boosting to improve accuracy and reduce the risk of overfitting. It also evaluates different feature selection methods to enhance model efficiency.

The results indicate that ensemble learning techniques outperform single-model approaches by capturing diverse patterns in data and reducing variance. The research highlights the importance of combining multiple weak learners into a strong predictive model for medical diagnoses, particularly when data is limited or noisy.

[10] Akkem Yaganteeswarudu, Infoshare Systems, Pyramid Softsol Pvt Limited, Hyderabad,India.

This study, titled "Multi Disease Prediction Model by using Machine Learning and Flask API," was presented at the Fifth International Conference on Communication and Electronics Systems (ICCES 2020). It outlines the development of a web-based application using Flask to predict multiple diseases such as diabetes, heart disease, and cancer.

Each disease prediction module is built using a different algorithm tailored to the characteristics of the corresponding dataset. For example, Logistic Regression for diabetes, Random Forest for heart disease, and SVM for cancer. The model accuracy ranges from 92% to 96% across diseases.

The integration of Flask for the backend and Python-based ML libraries (like scikit-learn and TensorFlow) makes this a flexible and scalable solution. This paper demonstrates how real-time predictions can be served through a REST API, making the application accessible across different platforms.

[11] K. H. Leung, M. R. Salmanpour, A. Saberi, I. S. Klyuzhin, V. Sossi, A. K The paper titled "Using Deep Learning to Predict Outcome of Patients with Parkinson’s Disease" presents a neural network-based method to analyze clinical and imaging data for predicting disease progression in Parkinson’s patients. It was presented at an IEEE conference in 2018.

The model focuses on integrating imaging data (e.g., PET scans) with clinical parameters using convolutional neural networks (CNNs). This allows for highly

granular predictions regarding disease severity, rate of progression, and treatment outcomes.

1.4 Hardware Requirements

The implementation of the *MediForecast* multi-disease prediction system is designed to be efficient and lightweight, enabling it to run on commonly available consumer-grade hardware. This flexibility ensures that both developers and healthcare professionals can deploy and test the system without requiring expensive infrastructure.

The project was developed and tested on a machine equipped with an Intel i5 processor, 8 GB RAM, and a 2 GB Nvidia Graphics Processor. The CPU in use had two cores running at clock speeds of 1.7 GHz and 2.1 GHz, respectively. This configuration proved sufficient for the entire machine learning workflow, including data preprocessing, model training, and prediction generation.

The training phase, which involves fitting machine learning models to healthcare datasets, typically takes 10 to 15 minutes depending on the algorithm and dataset size. Once trained, the testing or prediction phase is extremely efficient, taking only a few seconds to return predictions and model performance metrics.

Minimum System Requirements for Testing

To ensure smooth execution and usability, the following hardware specifications are recommended:

- **Device**: Laptop or Desktop
- **RAM**: 8 GB (minimum)
- **Storage**: 500 GB HDD or SSD
- **Processor**: Dual-core CPU with clock speed ≥ 2.0 GHz
- **Architecture**: 32-bit or 64-bit system

- **Internet Connection**: Required for model deployment, dataset access, and web application functionality

These requirements ensure that the *MediForecast* system is accessible and deployable across a wide range of computing environments, including educational, clinical, and research settings.

1.5 Software Requirements

The *MediForecast* system has been developed using widely adopted, open-source technologies to ensure accessibility, flexibility, and robust performance. The software stack selected is particularly well-suited for machine learning workflows, data processing, and web-based application development.

1. Python

Python is the core programming language used for implementing the *MediForecast* system. Python's syntax is clean and easy to understand, making it ideal for rapid development. It supports a vast ecosystem of libraries that are essential for machine learning and data science, such as:

- **NumPy** – for numerical operations
- **Pandas** – for data manipulation and analysis
- **Matplotlib / Seaborn** – for visualization
- **Scikit-learn** – for machine learning model development and evaluation
- **TensorFlow / Keras** (optional) – for deep learning extensions
- **Streamlit** – for building the user interface and web app

Machine learning, as a field, relies heavily on algorithms, mathematical models, and processed data to deliver AI-driven results. Python allows developers to integrate all of these aspects seamlessly in one ecosystem.

2. Operating System

While the *MediForecast* system is cross-platform and can run on Windows, macOS, or Linux, the preferred operating system for development and deployment is Ubuntu Linux. Ubuntu is especially suitable due to:

- Compatibility with Docker and Python environments
- Better performance in managing dependencies
- Enhanced security and system resource management
- Extensive community and developer support

Ubuntu provides a stable and secure environment that is well-aligned with modern machine learning and data science tools.

3. Development Tools

A variety of tools and environments were utilized throughout the development process to streamline coding, testing, and visualization:

- **Spyder IDE** – A powerful Python development environment tailored for data science workflows, integrated with IPython.
- **Google Colab** – A cloud-based Jupyter notebook environment that allows running Python code using Google's computing resources, especially useful for training models without local hardware constraints.
- **Visual Paradigm** – A modeling tool used for drawing UML diagrams such as Use Case, Class, Activity, and Data Flow Diagrams, which are vital for system design and documentation.

CHAPTER 2

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing System

In the existing healthcare diagnostic landscape, machine learning models are predominantly designed to predict only a single disease at a time. These traditional systems rely on specific algorithms tailored to individual diseases such as heart disease or diabetes but lack the versatility to predict multiple conditions simultaneously.

Various machine learning techniques have been employed in these single-disease prediction systems, including Logistic Regression**,** Support Vector Machines (SVM)**,** Decision Trees**,** Naive Bayes**,** J48, and Artificial Neural Networks (ANNs). Among these, Neural Networks have been found to offer superior predictive capabilities, especially when trained on large datasets and optimized for feature selection.

Most of the current systems focus on cardiovascular disease prediction due to its high mortality rate. One such example is the Heart Disease Prediction System (HDPS), which incorporates models like SVM**,** Naïve Bayes**,** Random Forest**,** and K-Nearest Neighbors (KNN)**.** While these systems provide a fair degree of accuracy, they are often isolated solutions – designed for one specific health condition at a time.

Moreover, some research has explored deep learning and supervised learning methods to enhance diagnostic accuracy. However, these efforts are still fragmented, and the integration of multiple disease predictions into a single platform remains underexplored.

Thus, the current systems, while valuable, do not fully address the real-world scenario where patients are at risk of developing multiple co-occurring diseases such as diabetes, cardiovascular issues, and neurological disorders. The lack of integration and limited scope in these existing models restrict their usability and impact in a clinical setting.

Some systems, particularly those using J48 Decision Trees or Artificial Neural Networks (ANNs), offer improved flexibility and performance but still remain disease-specific. For instance, a neural network trained to detect Parkinson's disease using motor function features and speech signals cannot be directly reused for detecting diabetes, which involves an entirely different set of medical parameters such as glucose levels and insulin sensitivity.

Moreover, the existing models often do not handle data integration challenges well. Patient data may come from disparate sources like Electronic Health Records (EHR), wearable devices, lab reports, and manual inputs. Current systems struggle to unify this data in a way that supports holistic diagnosis. This lack of interoperability and scalability severely limits the application of traditional systems in comprehensive, preventive healthcare.

Additionally, many existing models are deployed in isolation meaning they serve a specific clinic, hospital, or application. These systems are not designed with cross-platform deployment, cloud scalability, or user accessibility in mind, especially for patients in remote or underprivileged areas.

2.1.1. Disadvantages of Existing Systems

Current systems for predicting cardiovascular diseases exhibit several notable limitations. They frequently encounter issues such as overfitting and underfitting of data, alongside difficulties in integrating diverse data sources. Specific drawbacks observed across different approaches include:

- **First approach:** Exhibits low true negative rates, reducing the reliability of correctly identifying non-disease cases.
- **Second approach:** Requires specialized training, which limits accessibility and usability for general practitioners.
- **Third approach:** Produces inconsistent results due to dependence on specific tools and platforms, affecting reproducibility.
- **Fourth approach:** Lacks the capability to predict the performance of individual algorithms, hindering algorithm selection and optimization.
- **Fifth approach:** Demands relatively high processing time and involves handling complex data features, making the prediction process resource-intensive

These challenges highlight the need for improved, more robust models that ensure accuracy, efficiency, and broader applicability in cardiovascular disease prediction.

2.2 Proposed System

The proposed multi-disease prediction system integrates critical patient data using advanced data integration techniques and leverages robust machine learning algorithms such as Naïve Bayes, Logistic Regression, and Random Forest. This system is designed to overcome the limitations of existing models by prioritizing data quality and consistency throughout the processing pipeline.

Key features of the proposed system include:

- **Feature Engineering:** To extract and select the most relevant features, improving the predictive power of the models.
- **Cross-Validation:** To ensure generalizability and reduce overfitting or underfitting by validating the models on multiple data subsets.
- **Regularization Techniques:** To optimize model complexity and enhance performance by preventing overfitting.
- **Hyperparameter Tuning:** To fine-tune model parameters for achieving the best possible accuracy.

By combining these techniques, the system aims to deliver a holistic, accurate, and practical approach for predicting multiple diseases simultaneously. This approach not only supports healthcare professionals in making informed decisions but also empowers patients through early and reliable disease risk assessment.

2.1.2 Advantages of the Proposed System

The proposed multi-disease prediction system offers several significant advantages, including:

- **Tailored Features:** Designed to enhance prediction accuracy by focusing on disease-specific and relevant patient data.
- **User-Friendly Interface:** An intuitive UI that enables easy access and interaction for both patients and healthcare assistants, promoting wider adoption.
- **Scalability:** The system is built to efficiently handle increasing volumes of data and users without performance degradation.
- **Robust Data Security:** Ensures patient data privacy and protection through advanced security protocols.
- **Interoperability:** Supports integration with existing healthcare systems and databases for seamless data exchange.
- **Cost Reduction:** By automating prediction and early detection, the system helps lower diagnostic and treatment expenses.
- **Streamlined Healthcare Delivery:** Facilitates faster, more accurate decision-making, reducing administrative workload for healthcare providers.
- **Research Support:** Provides valuable insights and datasets that can assist ongoing medical research and development.

CHAPTER 3

.

CHAPTER 3

SYSTEM DESIGN

3.1System Architecture

The system architecture of the proposed multi-disease prediction system is designed to efficiently collect, process, and analyze healthcare data to provide accurate disease predictions and support clinical decisions. The architecture consists of the following key components:

- **Data Sources**: Collect health data from electronic health records, wearables, IoT sensors, and patient inputs.
- **Data Ingestion**: Integrate and normalize data from various sources for analysis.
- **Data Storage**: Maintain databases and cloud storage for storing patient data and model parameters.
- **Data Preprocessing**: Clean and prepare data, including feature engineering and transformation.
- **Machine Learning Models**: Develop disease-specific predictive models using algorithms like SVM, logistic regression, and decision trees.
- **Prediction and Decision Support**: Use trained models to make disease predictions and offer clinical decision support.
- **User Interfaces**: Build web-based or mobile app interfaces for patients and healthcare professionals.

The proposed system architecture is designed to facilitate the efficient flow of healthcare data from collection to prediction, enabling accurate and timely disease diagnosis. It begins with the collection of health data from various sources, including electronic health records (EHRs), wearable health monitoring devices, IoT-based sensors, and direct patient inputs.

These diverse sources ensure that the system captures a comprehensive and real-time picture of the patient's health.Once the data is collected, it undergoes a data ingestion process where information from different formats and systems is integrated and normalized.

This step is essential to maintain consistency and reliability across all incoming data. The processed data is then securely stored using both relational databases and cloud storage platforms, which provide scalability, fast retrieval, and robust data management.Before model training, the data is preprocessed to improve its quality and suitability for analysis. This involves cleaning the data by removing errors or duplicates, handling missing values, engineering relevant features, and applying necessary transformations. These preprocessing steps ensure that the data fed into the machine learning models is accurate and relevant.

The system design of the *MediForecast* project plays a pivotal role in establishing a robust and scalable framework for multi-disease prediction. This phase involves structuring the overall system architecture, identifying functional components, and defining their interactions to ensure smooth and accurate operation. The system is designed to predict three major diseases—heart disease, diabetes, and Parkinson's disease—by employing machine learning algorithms within an interactive and user-friendly Streamlit interface.

At the core of the design is a modular architecture that separates the frontend, backend, and data layers, allowing for easier maintenance and future scalability. The frontend, developed using Streamlit, provides users with a clean and intuitive interface where they can input their health-related data. This includes features like age, gender, blood pressure, cholesterol, glucose level, and other medical attributes. Once the user submits the data, it is passed to the backend where preprocessing is performed. This involves cleaning the data, handling missing values, scaling numerical values, and encoding categorical variables to ensure the machine learning models receive quality inputs.

Following preprocessing, the data is fed into disease-specific machine learning models. Each model—Logistic Regression for heart disease, Support Vector Machine for Parkinson's, and Random Forest for diabetes—has been trained and optimized to handle the respective datasets. These models predict the presence or absence of the disease, providing a binary output along with probability scores for better interpretability. The results are then displayed on the same interface, supported by informative visualizations and model performance metrics such as accuracy, precision, recall, and F1-score.

The system has been carefully designed to be user-centric, offering an efficient workflow that starts with data entry and ends with actionable predictions. It also incorporates design principles that support interpretability, security, and privacy, ensuring sensitive medical data is handled responsibly. Additionally, the modular nature of the system allows for easy integration of new disease models in the future, paving the way for a comprehensive AI-powered healthcare assistant.

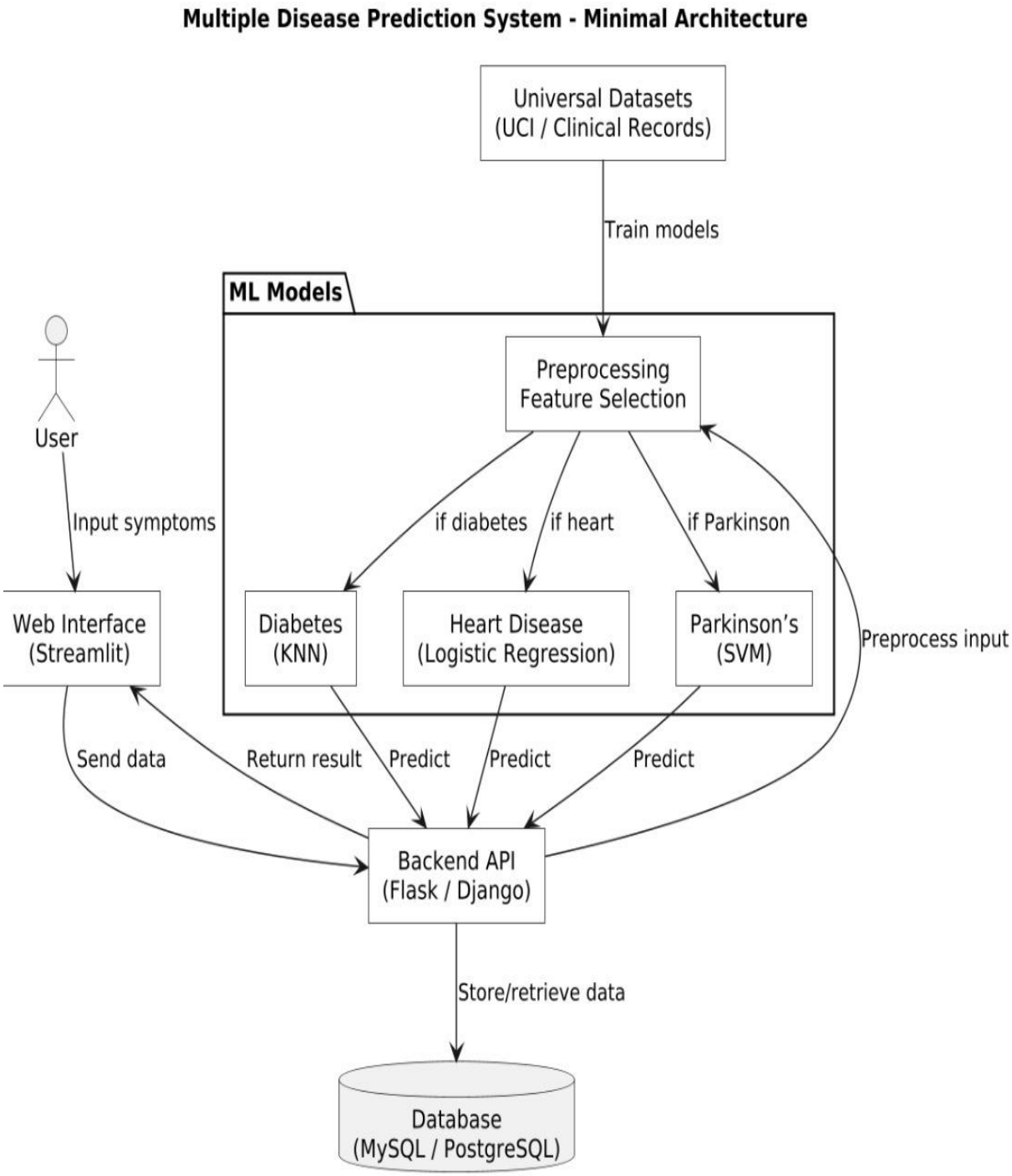

Fig.3.1 Representation of system Architecture

3.2 Use Case Diagram

A use case diagram for heart disease prediction outlines the system's functionality in a concise manner. It typically includes actors (users or external systems) and use cases (functionalities) associated with loading datasets, building models, and

predicting heart disease. Users, such as medical professionals or researchers, interact with the system to perform essential tasks.

- **Load Dataset**:This use case involves importing patient data, including medical history, clinical measurements, and risk factors such as cholesterol levels, blood pressure, and age.
- **Build Model**: This represents the process of training a machine learning model using the loaded dataset. It includes steps like data preprocessing, algorithm selection, training, and validation.
- **Predict Disease**:Users can input new patient information to receive a predictive assessment of heart disease risk. The system uses the trained model to evaluate the input data and provide results.

This diagram illustrates the core functionalities of the system and how the User interacts with it to complete critical tasks in the disease prediction pipeline.

The use case diagram for the heart disease prediction system provides a high-level overview of how users interact with the system to perform essential tasks such as data loading, model training, and disease prediction. The primary actor involved is the user, typically a medical professional, data analyst, or researcher, who utilizes the system to assess the likelihood of heart disease in patients based on clinical data. The system offers three main use cases: Load Dataset, Build Model, and Predict Disease.

The Load Dataset use case allows users to import patient data files that include relevant attributes such as age, gender, blood pressure, cholesterol levels, and other medical history indicators. This data forms the foundation for training the prediction model. The Build Model use case represents the system's capability to preprocess the dataset, select appropriate features, and train machine learning algorithms like Logistic Regression, Random Forest, or Support Vector Machines to learn patterns indicative of heart disease. Once the model is trained, users can engage with the Predict Disease use case, where they input new patient data through the interface, and the system applies the trained model to assess the risk of heart disease, returning the result in an interpretable format.

This use case diagram effectively illustrates the core functional interactions between the user and the system. It defines the system boundary by focusing on data handling, model operations, and prediction output while abstracting the complexity of the underlying algorithms. By modeling these interactions, the diagram aids in understanding the scope of the system and guides the development and user interface design, ensuring that each task is clearly supported by a corresponding system function.

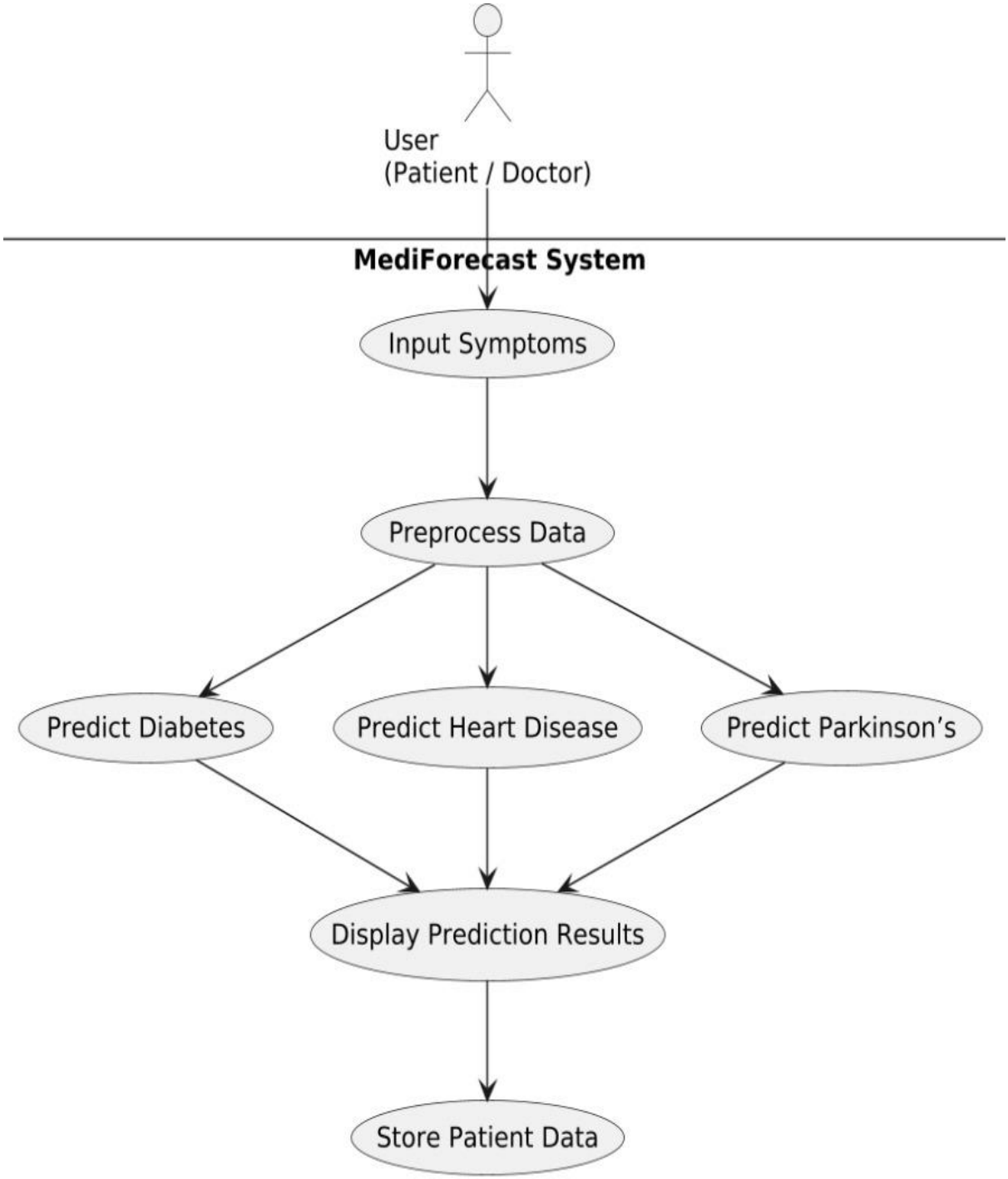

Fig.3.2 Representation Use Case Diagram

3.3 Class Diagram

The class diagram for the heart disease prediction system provides a structural representation of the system's core components, highlighting the various classes involved, their attributes, and the relationships among them. This diagram is an essential part of the object-oriented design, ensuring that the system is modular, scalable, and easy to maintain.

The central class in the system is the Patient class, which encapsulates attributes such as patientID, name, age, gender, cholesterol, bloodPressure, and other relevant medical indicators. This class serves as the primary data entity and interfaces with both data storage and prediction modules. Linked to the Patient class is the MedicalRecord class, which maintains historical health data and diagnosis results, providing a longitudinal view of the patient's condition.

The DatasetLoader class is responsible for importing and validating external datasets. It includes methods such as loadCSV() and validateData() to ensure data integrity before it is passed to the modeling components. The ModelBuilder class handles the machine learning logic, including data preprocessing, feature selection, model training, and evaluation. It interacts with the MLModel class, which defines model-specific parameters and methods such as train(), predict(), and evaluate().

Finally, the PredictionEngine class acts as the interface between the trained model and the user input during prediction. It processes the new patient data and returns a risk assessment, possibly in terms of a probability or a binary outcome (positive/negative for heart disease).

The relationships among these classes are designed to reflect real-world interactions: the Patient class is associated with MedicalRecord, and both interface with the ModelBuilder during training and the PredictionEngine during inference. This structure supports a clean separation of concerns and enhances code readability, reusability, and ease of testing.

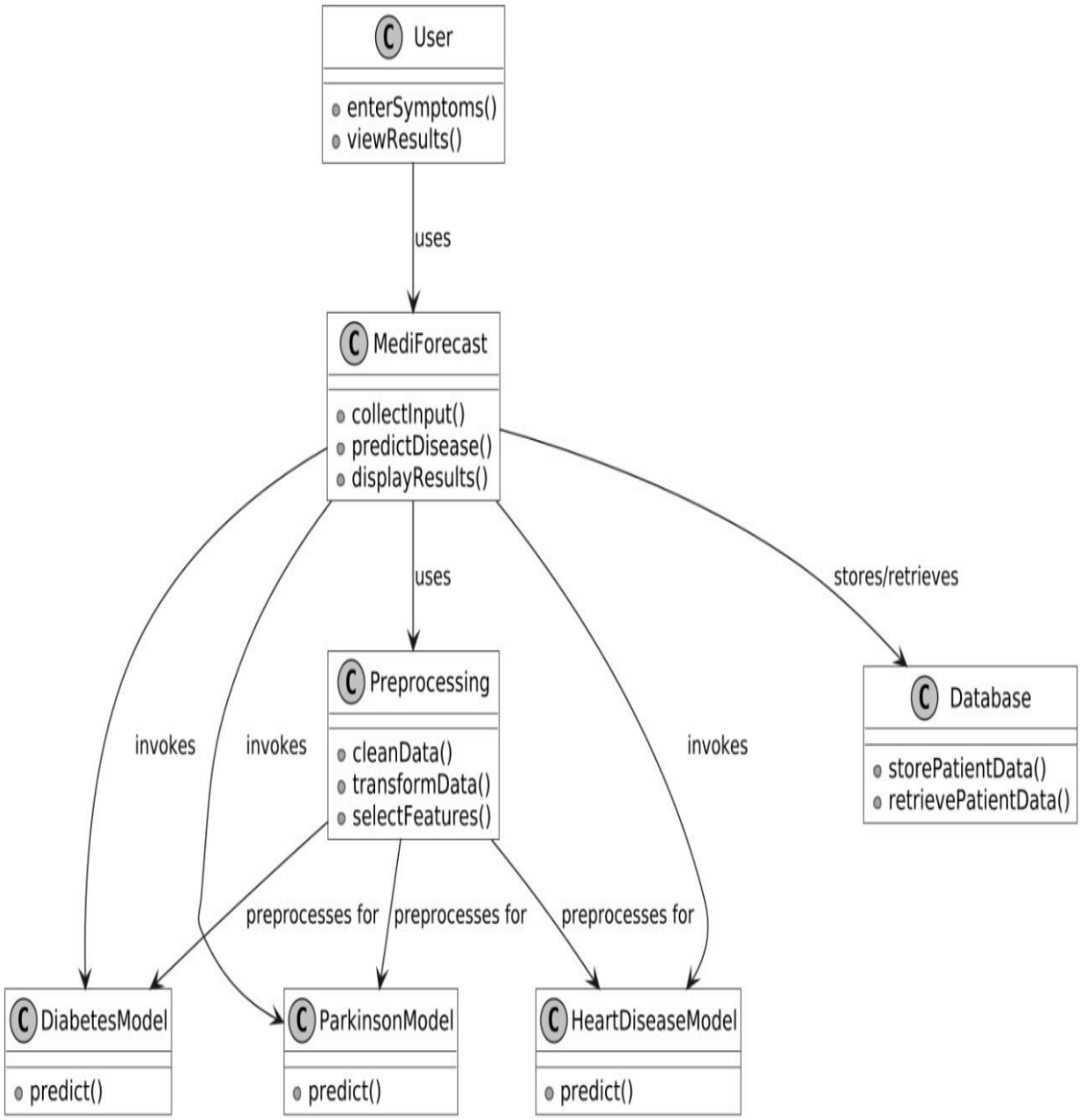

Fig.3.3 Representation of Class Diagram

3.4 Data Flow Diagram

The Data Flow Diagram (DFD) for the heart disease prediction system provides a clear and logical representation of how data moves through various components of the application. It outlines the transformation of raw input data into meaningful outputs, helping stakeholders understand the system's information flow, data

dependencies, and processing stages. The DFD is instrumental in identifying potential bottlenecks or redundancies and in ensuring that data handling is both accurate and efficient.

At the input level, the system receives patient information from external sources such as uploaded medical records or manual form entries. This data typically includes key health indicators such as age, gender, blood pressure, cholesterol, and other diagnostic features. The first processing step involves the Data Input Module, where the raw data is validated and converted into a suitable format for further processing.

Validated data is passed to the Data Preprocessing Module, which performs tasks such as handling missing values, normalization, and feature encoding. The clean dataset is then forwarded to the Model Training Module if the system is in training mode, or to the Prediction Module when new patient data is being analyzed. In the training pathway, the Model Training Module utilizes historical data to build or update the machine learning model. The trained model is stored in the Model Repository for reuse during predictions.

When a user inputs new patient data for prediction, the information is routed through the Prediction Module, where the trained model is applied to assess the risk of heart disease. The output, typically a risk score or binary classification, is sent to the Result Display Interface, where it is presented to the user through a graphical interface such as a probability bar, chart, or decision label (e.g., "High Risk", "Low Risk").

The DFD ensures that each step in the data lifecycle—input, preprocessing, modeling, prediction, and output—is well defined and interconnected. This facilitates an efficient and transparent flow of data, ultimately supporting the system’s goal of delivering reliable and timely disease risk assessments.

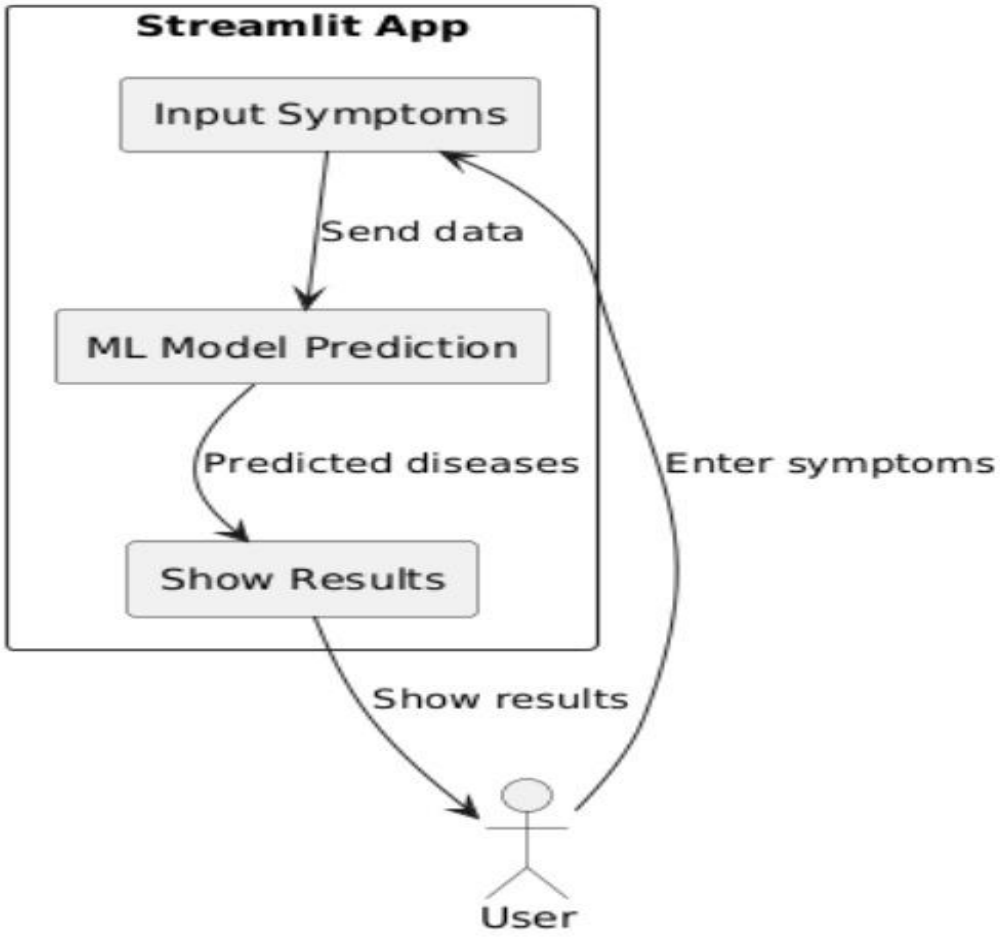

Fig.3.4 Representation of Data Flow Diagram

3.5 Activity Diagram

The activity diagram for the heart disease prediction system illustrates the sequential flow of activities involved in processing patient data and generating disease risk predictions. It is a dynamic behavioral diagram that highlights the control flow from one activity to another, making it easier for stakeholders to comprehend the system's logic, identify inefficiencies, and explore opportunities for automation.

The workflow begins with data collection, where patient medical information is either uploaded through files or manually entered via the user interface. This raw data then flows into the data validation and preprocessing stage, where it is checked for completeness, cleaned, and transformed into a format suitable for analysis. This step may include handling missing values, normalizing numerical features, and encoding categorical variables.

Once preprocessing is complete, the system checks the operational mode—model training or prediction. If the system is in training mode, the preprocessed data is passed to the model training activity, where machine learning algorithms such as Logistic Regression, Random Forest, or Support Vector Machines are used to train predictive models. The trained model is then evaluated and stored in the model repository for future use.
In prediction mode, newly entered patient data is sent to the prediction engine, which utilizes the trained model to assess the likelihood of heart disease. The result is then forwarded to the result generation stage, where the output is formatted into a readable and interpretable format, such as a probability score, risk level, or visual indicator.
The final activity involves displaying the prediction result to the user through a user-friendly interface, enabling medical professionals or researchers to take appropriate action based on the outcome. Throughout this process, decision nodes help control the flow based on conditions like "Is model available?" or "Is new data valid?", ensuring robustness and accuracy.
This activity diagram provides a comprehensive overview of the end-to-end system operations, facilitating clarity in design, development, and optimization of the heart disease prediction system.

The activity diagram of the heart disease prediction system depicts the detailed workflow involved in transforming raw patient data into actionable health insights. It captures all major steps, including data acquisition, preprocessing, model training, prediction, and result presentation, while also highlighting decision points and parallel processes. This comprehensive visualization aids developers, medical experts, and stakeholders in understanding the system's operation and facilitates identifying potential improvements or automation opportunities.
The process begins with data collection, where the user inputs patient information either by uploading datasets or entering individual records manually. Following this, the system performs data validation, ensuring completeness, correctness, and consistency of the input. Invalid or incomplete data triggers error handling routines, prompting the user for correction.

Once the data is validated, it moves to the preprocessing phase, which involves cleansing operations such as handling missing values, outlier detection, normalization, and feature extraction. This step is crucial to ensure that the machine learning algorithms receive high-quality input, thus improving model accuracy and reliability.
Subsequently, the workflow diverges based on the system's operational mode. In training mode, the system feeds the preprocessed data into the model training activity, where selected algorithms are applied to learn patterns indicative of heart disease. Model performance is evaluated using metrics like accuracy, precision,

and recall, and the best-performing model is saved for future predictions.

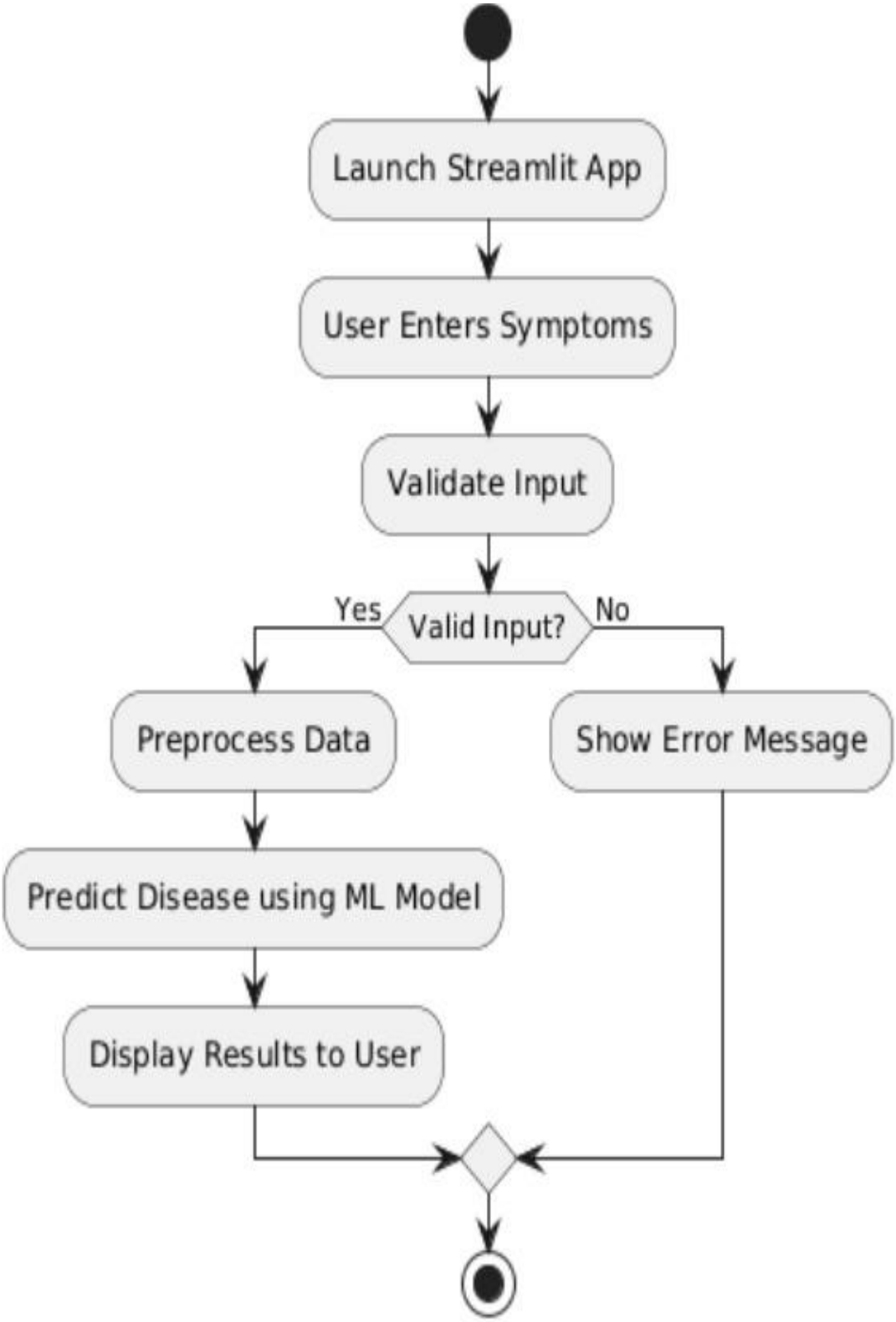

Fig.3.5 Representation of Activity Diagram

CHAPTER 4

CHAPTER 4

SYSTEM IMPLEMENTATION

Implementing a multi-disease prediction system using machine learning requires a systematic and phased approach to ensure accuracy, reliability, and clinical relevance. The development is typically divided into multiple stages, each focusing on critical aspects of the system.

The first phase is data collection, which involves gathering comprehensive datasets containing patient health records. These datasets must include demographic information, medical history, symptoms, and diagnostic test results. Importantly, the data should cover multiple diseases to ensure the models can generalize well across different medical conditions. The quality, diversity, and representativeness of the dataset are essential for building robust predictive models.

The second phase focuses on data preprocessing. This crucial step involves cleaning the data by handling missing values, correcting inconsistencies, and removing outliers. It also includes scaling or normalizing features to standardize input for machine learning algorithms. Feature selection is a key aspect here—identifying and retaining only the most relevant attributes that impact disease prediction helps enhance model performance and reduces overfitting risks. Proper preprocessing prepares the data into a structured format ready for modeling.

In the third phase, the focus shifts to model selection and training. Choosing the right machine learning algorithm depends on the nature of the data and the prediction task. Common algorithms used include logistic regression, decision trees, random forests, support vector machines, and deep learning models. The dataset is typically split into training and testing subsets to evaluate model generalization. Hyperparameter tuning optimizes the model's predictive capability. To assess model quality, metrics such as accuracy, precision, recall, F1-score, and AUC-ROC are employed. Besides prediction accuracy, interpretability is emphasized to help medical professionals understand which factors influence disease risk. Once validated, the model can be deployed in clinical environments, following strict healthcare regulations and privacy standards. Ongoing monitoring ensures the system remains accurate as new healthcare data becomes available.

4.1 Modules

The multi-disease prediction system is divided into key modules based on the targeted diseases:

1. **Heart Disease Prediction**
2. **Diabetes Prediction**
3. **Parkinson's Disease Prediction**

4.2.1 Heart Disease Prediction

The heart disease prediction module is designed to leverage a rich dataset consisting of patient clinical and demographic information. The dataset includes features such as age, sex, resting blood pressure, serum cholesterol levels, fasting blood sugar, electrocardiographic results, maximum heart rate achieved, exercise-induced angina, ST depression, and the slope of the peak exercise ST segment, among others. These attributes are essential indicators of cardiovascular health and risk factors.

Data preprocessing involves addressing missing or inconsistent values to ensure the integrity of the dataset. Outliers are identified and managed to prevent skewing the model, while normalization techniques bring features to a comparable scale, crucial for algorithms sensitive to magnitude differences. Additionally, feature selection methods like Recursive Feature Elimination (RFE) or correlation analysis are applied to retain the most informative predictors and reduce dimensionality.

For modeling, logistic regression is chosen due to its interpretability and effectiveness in binary classification problems, such as predicting the presence or absence of heart disease. The logistic model estimates the probability that a patient has heart disease, providing a clear decision boundary. Alternative models such as decision trees and random forests may be explored for comparative analysis.

The module is evaluated using several performance metrics: accuracy measures overall correctness, precision assesses how many predicted positives are true positives, recall (sensitivity) measures how many actual positives are captured, and F1-score balances precision and recall. Additionally, the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) provides insight into the trade-off between true positive and false positive rates.

In real-world healthcare settings, this predictive module enables clinicians to identify at-risk patients early, facilitating timely preventive measures and treatments. By predicting heart disease risk accurately, it helps reduce hospital admissions, lowers healthcare costs, and improves patient survival rates.

Heart disease remains one of the leading causes of mortality worldwide, and early prediction can dramatically improve patient outcomes. The dataset used in this module often originates from hospitals, clinical trials, or public repositories such as the UCI Heart Disease dataset. It contains both categorical and continuous variables, requiring careful preprocessing.

Challenges faced include handling class imbalance because datasets often have fewer positive (heart disease) cases compared to negatives, which can bias the model. Techniques like Synthetic Minority Over-sampling Technique (SMOTE) or class weighting can address this. Another challenge is ensuring data privacy

and compliance with healthcare regulations (HIPAA, GDPR).

Beyond logistic regression, ensemble methods like Gradient Boosting Machines or XGBoost can be experimented with to improve prediction accuracy. Interpretable AI techniques such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) can be integrated to explain model predictions to healthcare professionals, boosting trust and adoption.

Clinical relevance is high since timely identification of at-risk patients allows for lifestyle interventions, medication, or surgical options before severe complications occur. The module can be integrated into electronic health record (EHR) systems, providing decision support during patient consultations.

Future improvements could include incorporating additional biomarkers like genetic data, imaging results (echocardiograms), or longitudinal health records to enhance prediction performance and personalize risk assessment.

4.2.2 Diabetes Prediction

The diabetes prediction module utilizes a dataset capturing essential physiological and medical parameters, including plasma glucose concentration, diastolic blood pressure, BMI, diabetes pedigree function (genetic predisposition), insulin levels, age, and the number of pregnancies for female patients. These features collectively contribute to assessing the likelihood of a patient developing diabetes.

During preprocessing, the data undergoes cleansing to handle missing and inconsistent records. Since diabetes-related data can be skewed, normalization or standardization ensures all features contribute equally to the model. Advanced feature selection and engineering techniques may be employed to extract nonlinear interactions and improve predictive accuracy.

The module employs machine learning algorithms such as Random Forest and Support Vector Machines (SVM) due to their ability to model complex relationships and handle noisy data effectively. Random Forest's ensemble approach reduces overfitting by averaging multiple decision trees, while SVM finds the optimal hyperplane separating diabetic and non-diabetic cases.

Evaluation focuses on metrics beyond accuracy, including sensitivity (recall), which is critical in medical diagnosis to minimize false negatives, and specificity to avoid false positives. The ROC-AUC score is also examined to understand model discrimination capability.

Implementing this module in clinical practice aids in early detection of diabetes, enabling lifestyle modifications, medical treatment, and monitoring to prevent severe complications such as neuropathy, retinopathy, and cardiovascular diseases. Early diagnosis significantly improves patient outcomes and reduces the long-term burden on healthcare systems.

Diabetes prediction benefits significantly from datasets like the Pima Indians Diabetes dataset or hospital patient records. This module must deal with features influenced by lifestyle, genetics, and environment, making feature engineering crucial for capturing relevant risk factors.

Challenges include missing or inconsistent patient data, varying measurement standards, and heterogeneity across populations. Data augmentation or transfer learning from larger datasets can help generalize models better. Also, balancing model complexity and interpretability is critical because medical practitioners prefer clear, actionable insights.

Deep learning approaches, especially recurrent neural networks (RNNs) or LSTM networks, could be explored for sequential health data, such as monitoring glucose levels over time. Integrating this module with mobile health (mHealth) apps could enable continuous patient monitoring and real-time risk alerts.

Clinical impact is profound as early diabetes detection helps prevent complications like kidney failure, blindness, and cardiovascular issues. This module empowers healthcare providers with predictive analytics to tailor interventions based on individual risk profiles.

Future directions include incorporating lifestyle data (diet, exercise), wearable sensor data, and social determinants of health to create a holistic diabetes risk model. Additionally, adapting models for different demographic groups improves equity in healthcare.

4.2.3 Parkinson's Disease Prediction

The Parkinson's disease prediction module analyzes clinical and biomedical data obtained from patients' speech signals, motor assessments, and neurological tests. Key features include jitter, shimmer, harmonics-to-noise ratio, motor UPDRS scores, and other biomarkers associated with neurodegeneration.

Given the high dimensionality and noise inherent in speech and motor data, data preprocessing is crucial. Techniques such as filtering, normalization, and dimensionality reduction (e.g., Principal Component Analysis - PCA) help in extracting the most informative components and reducing computational complexity.

The module applies robust machine learning models like Support Vector Machines (SVM), Random Forests, and sometimes deep learning architectures tailored for sequence and signal data. SVM is particularly effective due to its capacity to handle complex, non-linear decision boundaries in high-dimensional feature spaces.

Model evaluation includes precision and recall to measure the accuracy of identifying Parkinson's patients, along with the F1 score for balanced assessment. Confusion matrices provide insights into false positives and false negatives, which are critical for minimizing misdiagnosis.

The implementation of this module supports early diagnosis of Parkinson's disease, which is essential since early-stage symptoms are often subtle and easily overlooked. Timely identification allows patients to start treatment earlier, potentially slowing disease progression and improving quality of life.

Parkinson's disease diagnosis is complex due to subtle early symptoms and variable progression. This module leverages unique data sources such as voice recordings, handwriting samples, and gait analysis alongside clinical metrics.

Challenges involve noisy and high-dimensional data, requiring sophisticated signal processing and dimensionality reduction methods. The scarcity of labeled datasets poses another obstacle, which can be addressed with data augmentation or semi-supervised learning techniques.

The module can benefit from multimodal learning, combining speech, motor, and imaging data to improve prediction accuracy. Explainable AI methods help clarify which features influence the diagnosis, aiding clinicians in understanding the underlying disease markers.

Clinical importance lies in enabling early-stage diagnosis, which is critical for effective symptom management and slowing disease progression through medications or therapies. It also supports longitudinal tracking of disease severity.

Future enhancements include integrating genetic data and biomarkers from cerebrospinal fluid analysis. Advances in wearable technology can provide continuous monitoring of motor symptoms, feeding data into predictive models for dynamic risk assessment.

CHAPTER 5

CHAPTER 5

CONCLUSION

In conclusion, our heart disease prediction project marks a significant advancement at the intersection of healthcare, medical research, and data science. By utilizing a comprehensive dataset encompassing critical medical and clinical features, we have developed a robust model that aids in the early detection of heart disease. Employing logistic regression as the core binary classification technique, the project demonstrates a promising capability to support healthcare professionals, researchers, and patients alike in making informed decisions. The thorough evaluation of the model using key performance metrics such as accuracy, precision, recall, and F1 score validates the reliability and effectiveness of our predictive system. Ultimately, this contributes to improved patient care, timely interventions, and enhanced healthcare outcomes.

Looking forward, our multi-disease prediction system is poised to expand with the integration of additional modules targeting Parkinson's disease and diabetes prediction. Building on the strong foundation laid by the heart disease module, future work will focus on enriching the datasets, improving data preprocessing and feature engineering, and experimenting with advanced machine learning algorithms like ensemble methods and deep learning. These efforts aim to deliver a more comprehensive, scalable, and accurate predictive platform.

By advancing early disease detection across multiple critical health conditions, our project aspires to empower clinicians with actionable insights, reduce disease burden, and ultimately foster a healthier society through timely and personalized healthcare interventions.

The success of this module sets a strong precedent for the broader multi-disease prediction system, which will encompass diabetes and Parkinson's disease in subsequent phases. Future work will focus on expanding data diversity, enhancing model robustness, and integrating interpretability tools to ensure transparency and trustworthiness. The goal is to build a comprehensive diagnostic tool that supports preventive healthcare and improves patient outcomes across multiple conditions.

Ultimately, this work highlights how artificial intelligence can be leveraged to revolutionize disease prediction and management, offering hope for more proactive, data-driven healthcare solutions in the near future.

CHAPTER 6

CHAPTER 6

APPENDICES

APPENDIX I

SIMULTATION ENVIORNMENT

IMPLEMENTATION TOOLS

Tools used:

- Python
- PyTorch
- Pandas
- Numpy
- Matplotlib
- Sklearn
- Google collab
- Cuda
- TensorFlow
- Seaborn
- Streamlit
- Jupter notebook
- Gen Ai
- LangChain

PYTHON:

We have used python for the deep learning models because of its large libraries, such as PyTorch and TensorFlow, which make model creation easier. It is also understandable to novices in complicated domains like deep learning because of its readability and simplicity of use.

PYTORCH:

We have used PyTorch library because it is simple to use and allows you to

modify your model as you go, it is a popular choice for deep learning. Additionally, it integrates nicely with Python, which facilitates the development and experimentation of neural networks by developers.

PANDAS:

Pandas is mostly used in deep learning for data analysis and manipulation, making it simple for developers to prepare, clean, and modify big datasets before feeding them into neural networks.

The process of preparing data for model training is made easier by its strong data handling capabilities.

NUMPY:

We have used Numpy here because of its effective numerical computation capabilities, which allow for quick and vectorized operations on multi-dimensional arrays and are crucial for implementing mathematical operations underlying neural network computations; Numpy is used in deep learning. Its functionalities have been enhanced to improve efficiency and make managing big amounts of data in deep learning jobs easier.

MATPLOTLIB:

When it comes to analyzing and interpreting model performance, dataset properties, and intermediate findings during model training and evaluation, Matplotlib is utilized in deep learning for data visualization. With its flexible plotting features, developers may produce educational visuals that improve comprehension and help with decision-making in deep learning workflows.

SKLEARN:

Because of its extensive set of machine learning tools, which include preprocessing, model selection, and assessment, Scikit-learn is utilized in deep

learning. This allows for the easier integration of conventional machine learning methods into deep learning pipelines. It is an invaluable tool for putting machine learning algorithms into practice and benchmarking them alongside deep learning models because of its intuitive interface and comprehensive documentation.

GOOGLE COLLAB:

We have used google collab for implementing and analysis of all the deep learning models because it is free of cost & have a nice user interface and cloud-based environment with access to powerful GPUs and TPUs, allowing us to train complex models efficiently without requiring their own hardware.

TENSORFLOW:

Because to TensorFlow's adaptable architecture, rich toolkit, and scalable ecosystem, deep learning developers may effectively create and implement a broad variety of neural network architectures. It is a well-liked option for deep learning applications at both the research and production levels due to its extensive documentation and community support.

STREAMLIT:

Streamlit is an open-source Python library that allows developers and data scientists to quickly build and share custom web applications for machine learning and data analysis. Unlike traditional web development frameworks that require knowledge of HTML, CSS, and JavaScript, Streamlit simplifies the process by using pure Python. This makes it an ideal choice for projects like *MediForecast*, where the focus is on data processing, model integration, and user interaction without the overhead of full-stack development.

One of the standout features of Streamlit is its real-time interactivity. It allows developers to add widgets such as sliders, input boxes, dropdown menus, and buttons, which dynamically update the content based on user inputs.

APPENDIX II
SIMULTATION INPUT CODE

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
parkinsons_data.head()
parkinsons_data.shape
parkinsons_data.info()
parkinsons_data.isnull().sum()
parkinsons_data.describe()
parkinsons_data['status'].value_counts()
parkinsons_data.groupby('status').mean()
X = parkinsons_data.drop(columns=['name','status'], axis=1)
Y = parkinsons_data['status']
print(X)
print(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=2)
print(X.shape, X_train.shape, X_test.shape)
model = svm.SVC(kernel='linear')
model.fit(X_train, Y_train)
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print('Accuracy score of training data : ', training_data_accuracy)
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print('Accuracy score of test data : ', test_data_accuracy)
input_data =
(197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.
01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.422229,
0.741367,-7.348300,0.177551,1.743867,0.085569)

# changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction if (prediction[0] == 0):
```

```
  print("The Person does not have Parkinsons Disease")

else:
  print("The Person has Parkinsons")
import pickle
filename = 'parkinsons_model.sav'
pickle.dump(model, open(filename, 'wb'))
loaded_model = pickle.load(open('parkinsons_model.sav', 'rb'))
for column in X.columns:
  print(column)
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
diabetes_dataset = pd.read_csv('/content/diabetes.csv')
diabetes_dataset.head()
diabetes_dataset.shape
diabetes_dataset.describe()
diabetes_dataset['Outcome'].value_counts()
diabetes_dataset.groupby('Outcome').mean()
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
 print(X)
print(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2,
stratify=Y, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train, Y_train)
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy score of the training data : ', training_data_accuracy)
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : ', test_data_accuracy)
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```

```
prediction = classifier.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
import pickle
filename = 'diabetes_model.sav'
pickle.dump(classifier, open(filename, 'wb'))
loaded_model = pickle.load(open('diabetes_model.sav', 'rb'))
input_data = (5,166,72,19,175,25.8,0.587,51)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
prediction = loaded_model.predict(input_data_reshaped)
print(prediction)
if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
for column in X.columns:
  print(column)
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
for column in X.columns:
  print(column)
heart_data = pd.read_csv('/content/heart.csv')
heart_data.head()
heart_data.tail()
heart_data.shape
heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data['target'].value_counts()
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
print(X)
print(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
stratify=Y, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
model = LogisticRegression()
model.fit(X_train, Y_train)
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy on Training data : ', training_data_accuracy)
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy on Test data : ', test_data_accuracy)
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
import pickle
filename = 'heart_disease_model.sav'
pickle.dump(model, open(filename, 'wb'))
loaded_model = pickle.load(open('heart_disease_model.sav', 'rb'))
for column in X.columns:
  print(column)
for column in X.columns:
  print(column)
!pip install streamlit streamlit-option-menu
import os
import pickle
import streamlit as st
from streamlit_option_menu import option_menu

# Set page configuration
st.set_page_config(page_title="Health Assistant",
                   layout="wide",
                   page_icon="□⚕")
```

```
# getting the working directory of the main.py
working_dir = os.path.dirname(os.path.abspath(_file_))

# loading the saved models

diabetes_model =
pickle.load(open(f'{working_dir}/saved_models/diabetes_model.sav', 'rb'))

heart_disease_model =
pickle.load(open(f'{working_dir}/saved_models/heart_disease_model.sav', 'rb'))

parkinsons_model =
pickle.load(open(f'{working_dir}/saved_models/parkinsons_model.sav', 'rb'))

# sidebar for navigation
with st.sidebar:
    selected = option_menu('Multiple Disease Prediction System',

                           ['Diabetes Prediction',
                            'Heart Disease Prediction',
                            'Parkinsons Prediction'],
                           menu_icon='hospital-fill',
                           icons=['activity', 'heart', 'person'],
                           default_index=0)

# Diabetes Prediction Page
if selected == 'Diabetes Prediction':

    # page title
    st.title('Diabetes Prediction using ML')

    # getting the input data from the user
    col1, col2, col3 = st.columns(3)

    with col1:
        Pregnancies = st.text_input('Number of Pregnancies')

    with col2:
        Glucose = st.text_input('Glucose Level')

    with col3:
        BloodPressure = st.text_input('Blood Pressure value')
```

```
    with col1:
        SkinThickness = st.text_input('Skin Thickness value')

    with col2:
        Insulin = st.text_input('Insulin Level')

    with col3:
        BMI = st.text_input('BMI value')

    with col1:
        DiabetesPedigreeFunction = st.text_input('Diabetes Pedigree Function
value')

    with col2:
        Age = st.text_input('Age of the Person')

    # code for Prediction
    diab_diagnosis = ''

    # creating a button for Prediction

    if st.button('Diabetes Test Result'):

        user_input = [Pregnancies, Glucose, BloodPressure, SkinThickness,
Insulin,
                      BMI, DiabetesPedigreeFunction, Age]

        user_input = [float(x) for x in user_input]

        diab_prediction = diabetes_model.predict([user_input])

        if diab_prediction[0] == 1:
            diab_diagnosis = 'The person is diabetic'
        else:
            diab_diagnosis = 'The person is not diabetic'

    st.success(diab_diagnosis)

# Heart Disease Prediction Page
if selected == 'Heart Disease Prediction':

    # page title
```

```
    st.title('Heart Disease Prediction using ML')

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

    with col1:
        age = st.text_input('Age')

    with col2:
        sex = st.text_input('Sex')

    with col3:
        cp = st.text_input('Chest Pain types')

    with col1:
        trestbps = st.text_input('Resting Blood Pressure')

    with col2:
        chol = st.text_input('Serum Cholestoral in mg/dl')

    with col3:
        fbs = st.text_input('Fasting Blood Sugar > 120 mg/dl')

    with col1:
        restecg = st.text_input('Resting Electrocardiographic results')

    with col2:
        thalach = st.text_input('Maximum Heart Rate achieved')

    with col3:
        exang = st.text_input('Exercise Induced Angina')

    with col1:
        oldpeak = st.text_input('ST depression induced by exercise')

    with col2:
        slope = st.text_input('Slope of the peak exercise ST segment')

    with col3:
        ca = st.text_input('Major vessels colored by flourosopy')

    with col1:
        thal = st.text_input('thal: 0 = normal; 1 = fixed defect; 2 = reversable
defect')
```

```
    # code for Prediction
    heart_diagnosis = ''

    # creating a button for Prediction

    if st.button('Heart Disease Test Result'):

        user_input = [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang,
oldpeak, slope, ca, thal]

        user_input = [float(x) for x in user_input]

        heart_prediction = heart_disease_model.predict([user_input])

        if heart_prediction[0] == 1:
            heart_diagnosis = 'The person is having heart disease'
        else:
            heart_diagnosis = 'The person does not have any heart disease'

    st.success(heart_diagnosis)

# Parkinson's Prediction Page
if selected == "Parkinsons Prediction":

    # page title
    st.title("Parkinson's Disease Prediction using ML")

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

    with col1:
        fo = st.text_input('MDVP:Fo(Hz)')

    with col2:
        fhi = st.text_input('MDVP:Fhi(Hz)')

    with col3:
        flo = st.text_input('MDVP:Flo(Hz)')

    with col4:
        Jitter_percent = st.text_input('MDVP:Jitter(%)')

    with col5:
```

```
    Jitter_Abs = st.text_input('MDVP:Jitter(Abs)')

with col1:
    RAP = st.text_input('MDVP:RAP')

with col2:
    PPQ = st.text_input('MDVP:PPQ')

with col3:
    DDP = st.text_input('Jitter:DDP')

with col4:
    Shimmer = st.text_input('MDVP:Shimmer')

with col5:
    Shimmer_dB = st.text_input('MDVP:Shimmer(dB)')

with col1:
    APQ3 = st.text_input('Shimmer:APQ3')

with col2:
    APQ5 = st.text_input('Shimmer:APQ5')

with col3:
    APQ = st.text_input('MDVP:APQ')

with col4:
    DDA = st.text_input('Shimmer:DDA')

with col5:
    NHR = st.text_input('NHR')

with col1:
    HNR = st.text_input('HNR')

with col2:
    RPDE = st.text_input('RPDE')

with col3:
    DFA = st.text_input('DFA')

with col4:
    spread1 = st.text_input('spread1')
```

```
with col5:
        spread2 = st.text_input('spread2')

with col1:
        D2 = st.text_input('D2')

 with col2:
        PPE = st.text_input('PPE')

    # code for Prediction
    parkinsons_diagnosis = ''

    # creating a button for Prediction
    if st.button("Parkinson's Test Result"):

        user_input = [fo, fhi, flo, Jitter_percent, Jitter_Abs,
                      RAP, PPQ, DDP,Shimmer, Shimmer_dB, APQ3, APQ5,
                      APQ, DDA, NHR, HNR, RPDE, DFA, spread1, spread2, D2,PPE]

        user_input = [float(x) for x in user_input]

        parkinsons_prediction = parkinsons_model.predict([user_input])

        if parkinsons_prediction[0] == 1:
            parkinsons_diagnosis = "The person has Parkinson's disease"
        else:
              parkinsons_diagnosis = "The person does not have Parkinson's
disease"

    st.success(parkinsons_diagnosis)
```

APPENDIX III
SIMULTATION RESULTS

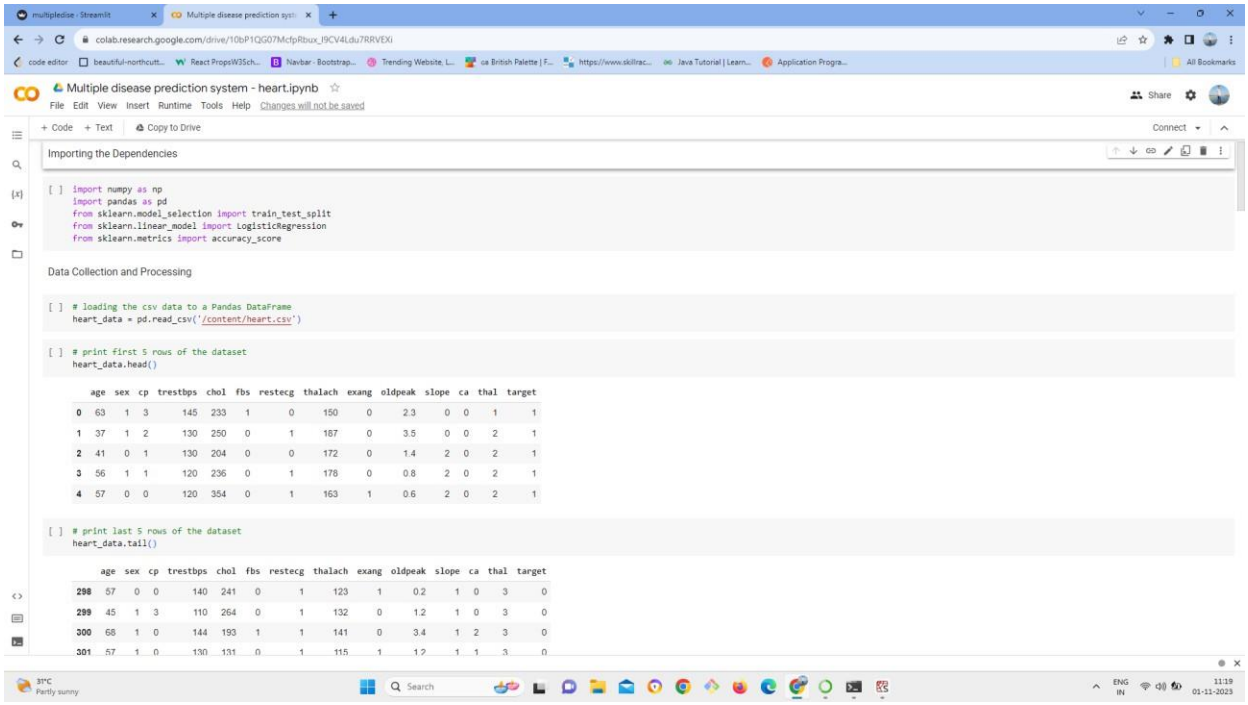

Fig1.output code1

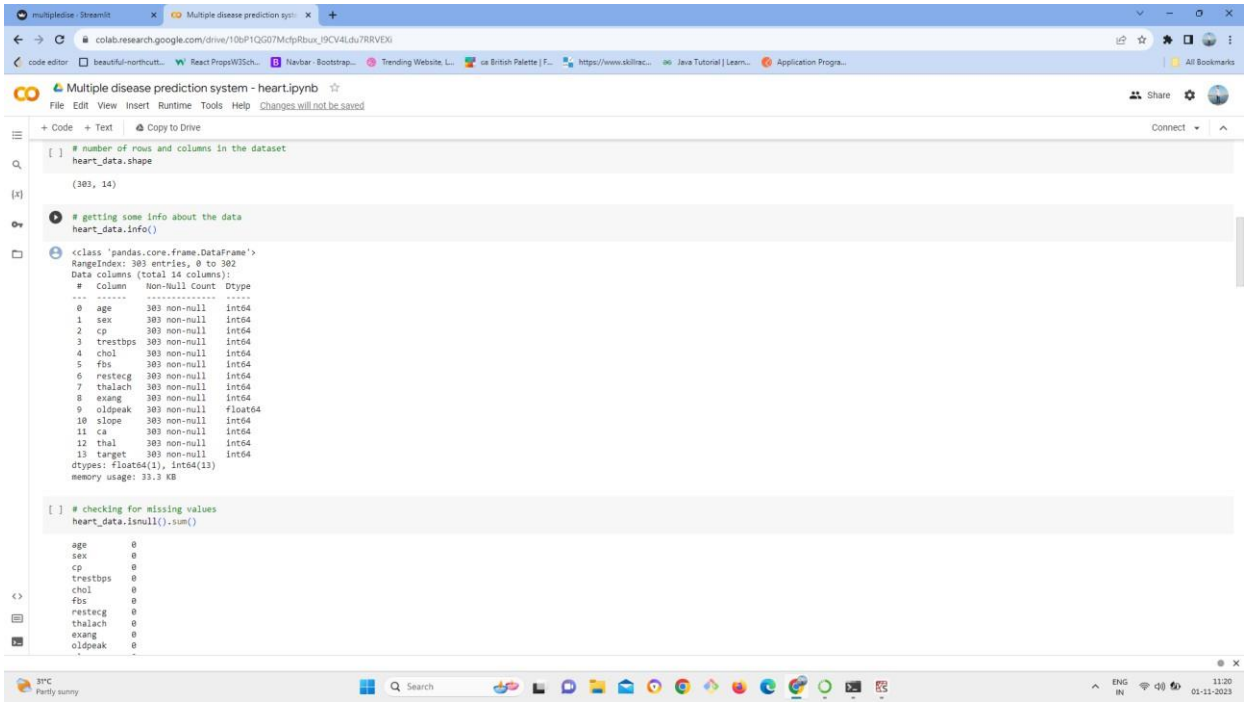

Fig2.output code2

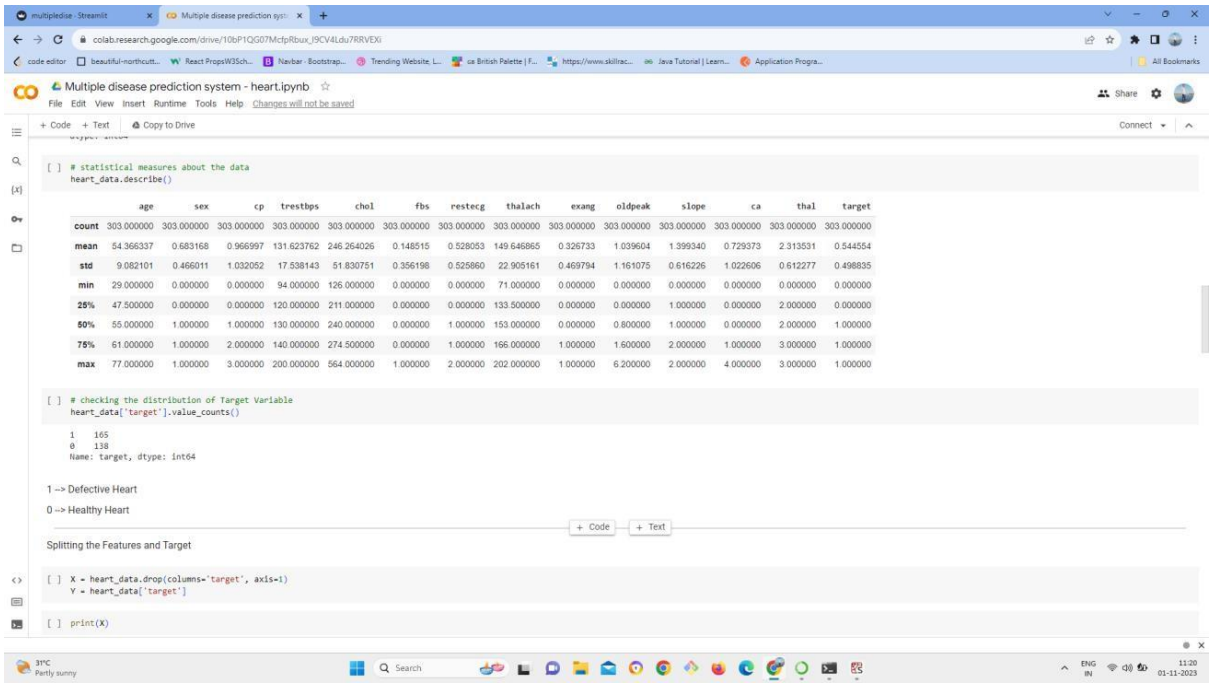

Fig3.output code3

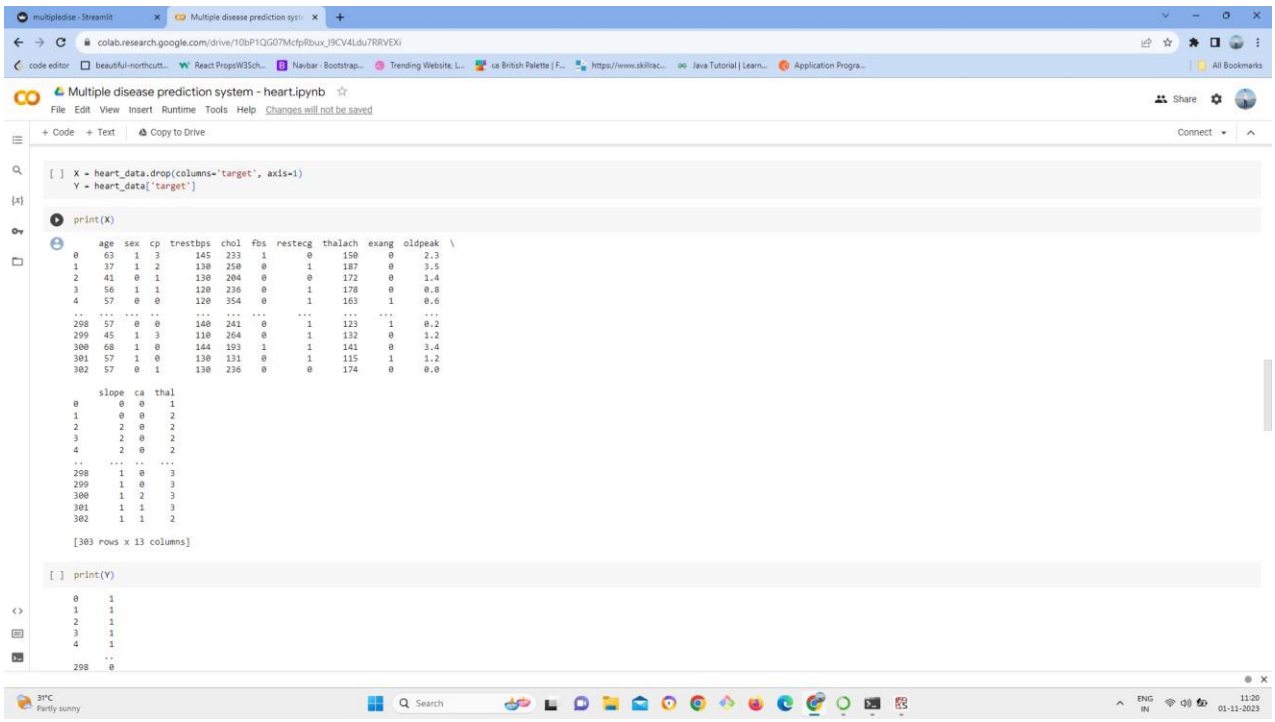

Fig4.output code4

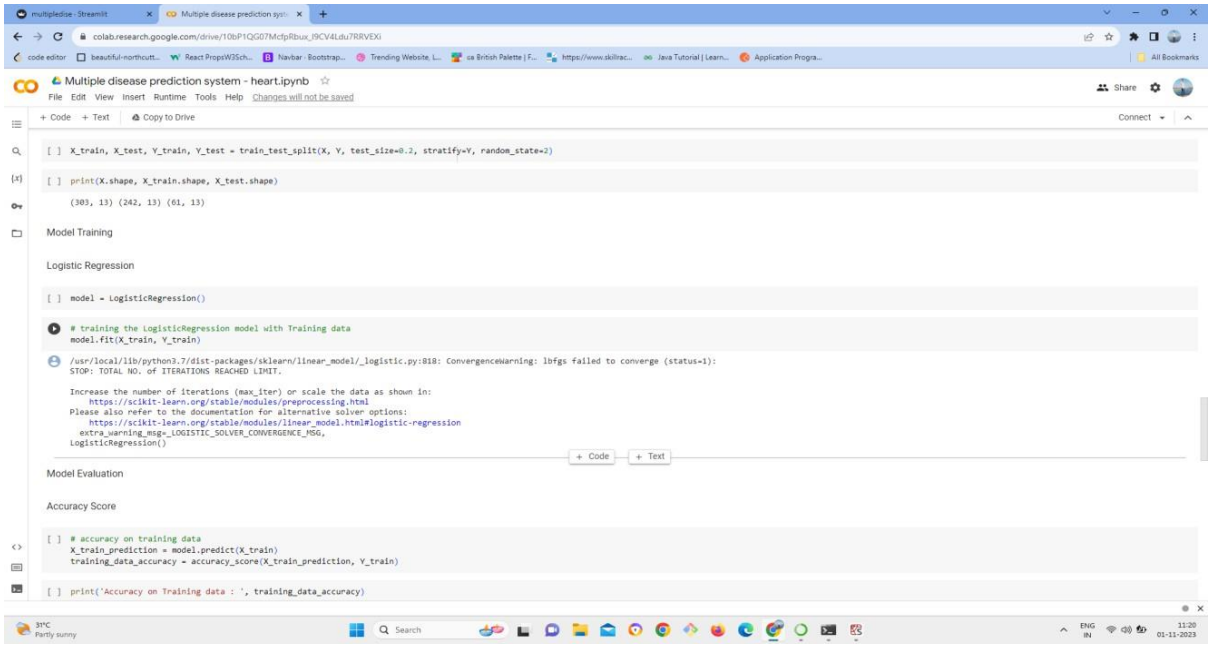

Fig5.output code5

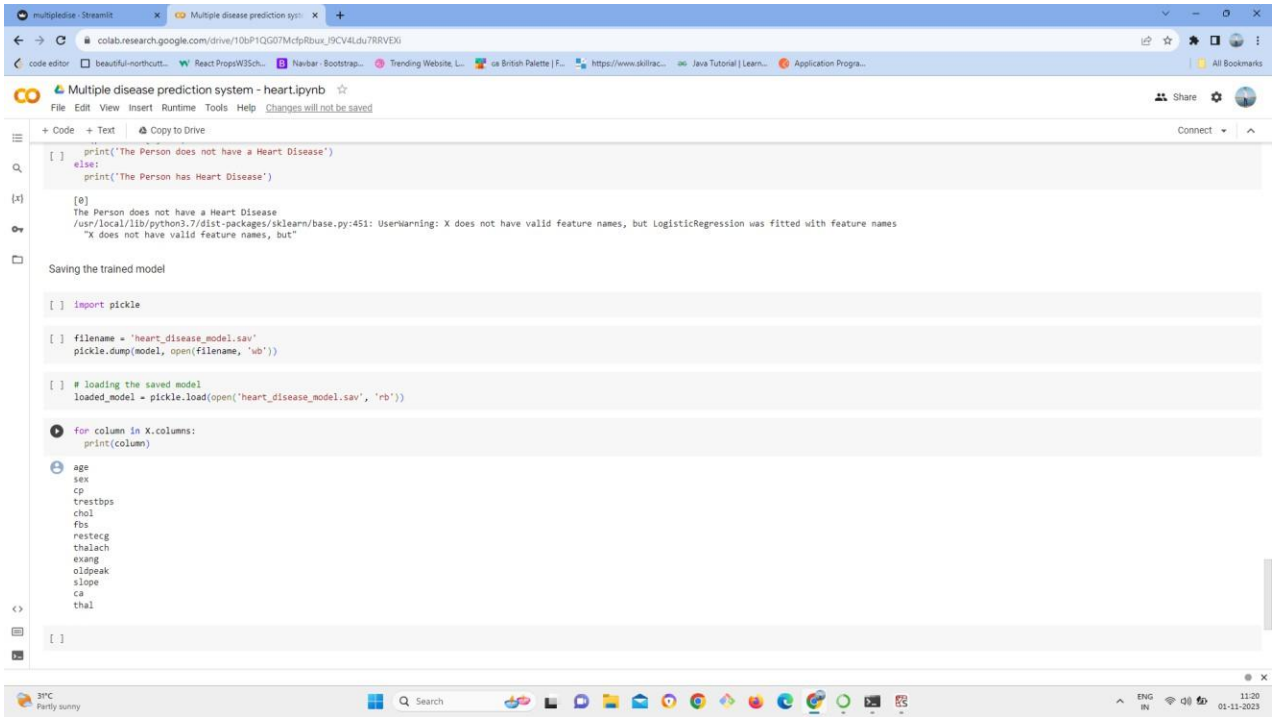

Fig6.output code6

UI:

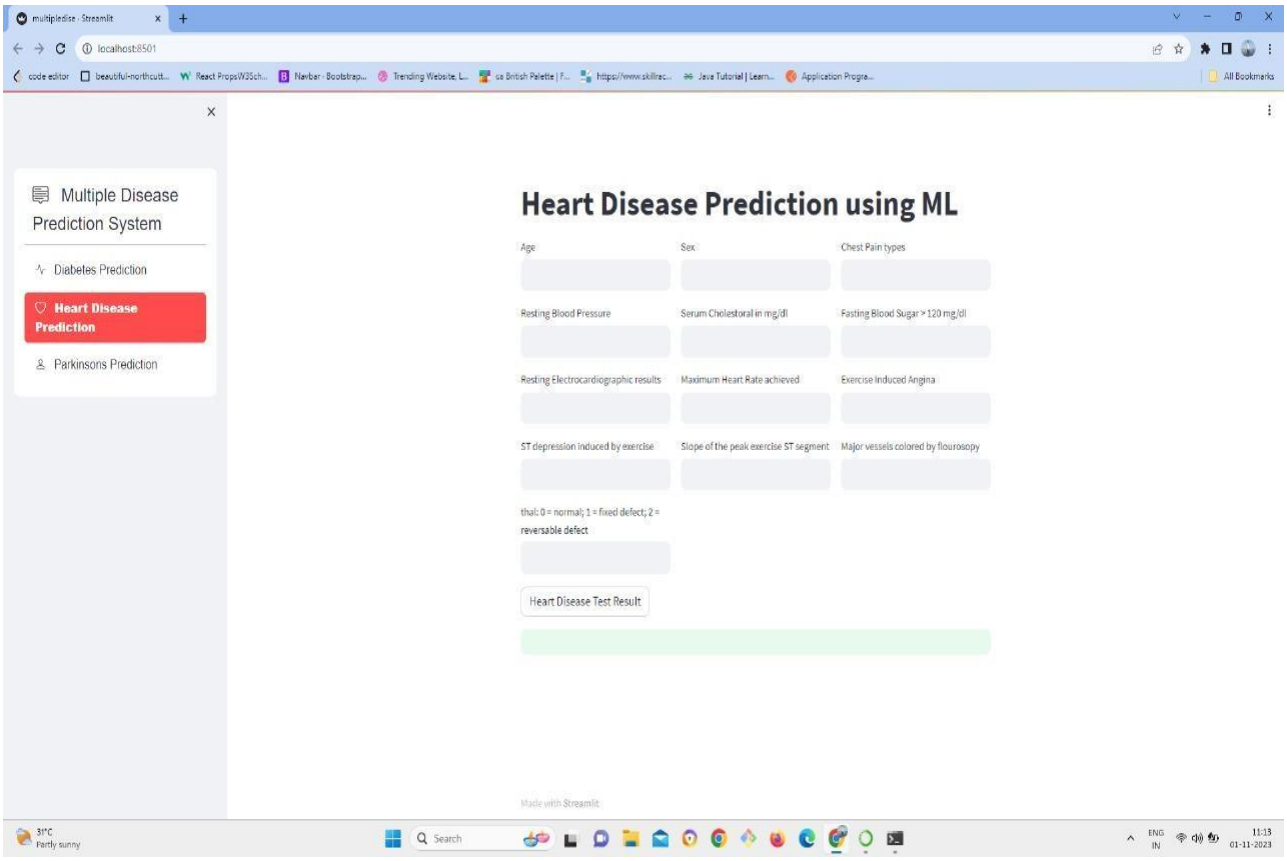

Fig7.UI interface

CHAPTER 7

CHAPTER 7

References

[1] JIAN PING LI, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. “Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare.” *IEEE Access*, vol. 8, pp. 105345-105356, June 2020. DOI: 10.1109/ACCESS.2020.3001149.

[2] Vijeta Sharma, Shrinkhala Yadav, Manjari Gupta, Computer Science, DST-Centre for Interdisciplinary Mathematical Science, Institute of Science, Banaras Hindu University, Varanasi, India. “Heart Disease Prediction using Machine Learning Techniques.” *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2020. DOI: 10.1109/ICACCCN51052.2020.9362842.

[3] Vijayalaxmi A, Sridevi S, Dr. Venkateswaran, Dr. Yashoda B S, Department of ECE, CMR Institute of Technology, Bengaluru, India. “Multiple Heart Diseases Prediction using Logistic Regression with Ensemble and Hyperparameter Tuning Techniques.” *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. IEEE Xplore.

[4] Aditi Gavhane, Gouthami Kokkula, Isha Pandya, Prof. Kailas Devadkar (PhD), Department of Information Technology, Sardar Patel Institute of Technology, Mumbai, India. “Prediction of Heart Disease Using Machine Learning.” *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology (ICECA 2018)*, IEEE Conference Record #42487, 2018. ISBN: 978-1-5386-0965-1.

[5] Saba Bashir, Zain Sikander Khan, Farhan Hassan Khan, Aitzaz Anjum, Khurram Bashir, Computer Science Department, Federal Urdu University of Arts, Science & Technology, Islamabad, Pakistan. “Improving Heart Disease Prediction Using Feature Selection Approaches.” *2019 16th International Bhurban Conference on Applied Sciences & Technology (IBCAST)*, Islamabad, Pakistan, Jan. 2019. DOI: 10.1109/IBCAST.2019.

[6] Pankaj Chittora, Sandeep Chaurasia (Senior Member, IEEE), Department of Computer Science and Engineering, Manipal University Jaipur, India.

"Prediction of Chronic Kidney Disease - A Machine Learning Perspective."

[7] Ramya G., Research Scholar, School of Computing, Sathyabama Institute of Science and Technology, Chennai, India. "Survey of Heart Disease Prediction and Identification using Machine Learning Approaches." *Proceedings of the Third International Conference on Intelligent Sustainable Systems (ICISS 2020)*, IEEE Xplore, 2020. ISBN: 978-1-7281-7089-3.

[8] Wenqi Li, School of Computer Science and Technology, Donghua University, Shanghai, China. "Coronary Heart Disease Prediction Based on Combined Reinforcement Multitask Progressive Networks." *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2020. DOI: 10.1109/BIBM49941.2020.9313275.

[9] Nikhila P. G., Student, Department of Electronics and Communication Systems, Centre for PG Studies, Mysuru, Karnataka, India. "Chronic Kidney Disease Prediction using Machine Learning Ensemble Algorithm." *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2021. DOI: 10.1109/ICCCIS51004.2021.9397144.

[10] Akkem Yaganteeswarudu, Infoshare Systems, Pyramid Softsol Pvt Limited, Hyderabad, India. "Multi Disease Prediction Model by using Machine Learning and Flask API." *Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020)*, IEEE Xplore, 2020. ISBN: 978-1-7281-5371-1.

[11] K. H. Leung et al., "Using deep-learning to predict outcome of patients with Parkinson's disease." *2018 IEEE Conference*, ISBN: 978-1-5386-8494-8.

[12] Gokul S., PG Student M.E (Applied Electronics), Department of EEE, Kongu Engineering College, Perundurai, India. "Parkinson's Disease Prediction Using Machine Learning Approaches." IEEE, 2019.

[13] Dr. P. Hamsagayathri, Dept of ECE, Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamil Nadu, India. "Symptoms Based Disease Prediction Using Machine Learning Techniques." *Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021)*, IEEE, 2021. ISBN: 978-0-7381-1183-4.