MultiDiseasePrognosis

Submitted in partial fulfilment of the requirements for

the award for the degree of

Bachelor of Computer Applications

To

Guru Gobind Singh Indraprastha University

Supervisor
Mr. Meetender
Assistant Professor

Submitted by:
Rohit Kumar Kain
Enrolment No. 03911102021

Banarasidas Chandiwala Institute of Information Technology

New Delhi - 110019

Batch – 2021 - 2024

BCA – 307

Synopsis

Problem Definition

Brief Description: The project aims to develop a disease prediction system utilizing artificial intelligence to predict three specific diseases - diabetes, heart diseases, and Parkinson's disease. The prediction will be based on measurable quantities such as blood rate, sugar level, and other relevant features. Users will input their measurable health data, and the system will employ machine learning algorithms to provide a ranking of potential diseases.

Exact Outcome: The outcome of the project will be a robust disease prediction model capable of analysing measurable health data to predict the likelihood of diabetes, heart diseases, and Parkinson's disease. The output will be presented in a ranked order, providing users with valuable information about potential health risks.

Why the Particular Project is Chosen: The project is chosen due to the critical importance of health in people's lives. In the current era, where diseases are on the rise, early detection is crucial for preventive measures. This system will empower individuals to monitor their health using measurable quantities, enabling early detection and prevention of severe diseases. The focus on specific diseases adds depth to the project's relevance.

Objective and Scope of Project:

The main objective is to create an intelligent disease prediction model using measurable health data, contributing to early detection and prevention of diabetes, heart diseases, and Parkinson's disease. The project aims to enhance overall health awareness and promote proactive healthcare decisions.

Methodology:

Data Collection: Data will be collected from reliable health agencies, medical databases, and research publications to ensure credibility. Measurable health data, including blood rate and sugar levels, will be collected in various formats such as CSV, and APIs will be utilized for dynamic and speedy data access.

Machine Learning Algorithms: Appropriate machine learning algorithms will be employed to analyse measurable health data and predict diseases. Algorithms such as regression models and classification algorithms will be used to achieve accurate predictions based on the provided features.

Hardware and Software: The project will require a computer with sufficient processing power for data analysis and machine learning tasks. Software tools, including the Python programming language, machine learning libraries (e.g., scikit-learn, TensorFlow), and data visualization tools, will be utilized for efficient development.

Contribution/Value: The project's contribution lies in its potential to empower individuals to take control of their health by providing early insights into potential health issues. Early detection of diabetes, heart diseases, and Parkinson's disease can lead to informed medical consultations and proactive healthcare decisions, ultimately improving health outcomes.

Limitations: The accuracy of disease detection may vary, especially for individuals with unique or evolving symptoms and diseases. The quality and completeness of data will significantly influence the effectiveness of the system. Privacy concerns may arise due to the collection and analysis of user health data.

Nature of the Research: The research is primarily original, focusing on the development of a disease prediction model tailored to specific measurable health features. While the idea aligns with existing disease prediction systems, the project's execution and focus on particular diseases makes it distinct.

Chapter 1: Introduction

1.1 Background

In the fast-evolving landscape of healthcare, the incorporation of cutting-edge technologies is essential to meet the challenges posed by the increasing prevalence of chronic diseases and the demand for more efficient and proactive healthcare solutions. The MultiDiseasePrognosis project emerges as a response to this need, positioning itself at the forefront of the intersection between artificial intelligence and healthcare. By employing sophisticated machine learning algorithms, the project seeks to redefine the landscape of disease prediction and management.

1.2 Project Overview

MultiDiseasePrognosis stands as a pivotal machine learning initiative focused on predicting three major diseases: diabetes, heart diseases, and Parkinson's disease. Through the implementation of advanced algorithms, the project aims to offer early and precise predictions by conducting a comprehensive analysis of diverse medical data. The integration of machine learning techniques distinguishes this project as a groundbreaking effort to transform conventional disease management practices.

1.3 Need for the Project

1.3.1 Rising Healthcare Challenges

Contemporary healthcare faces an unprecedented rise in chronic diseases, necessitating a shift from reactive to proactive healthcare strategies. MultiDiseasePrognosis addresses this challenge by providing a tool for early detection and intervention, aligning with the emerging paradigm of predictive and preventative healthcare.

1.3.2 Limitations of Current Healthcare Practices

Current healthcare practices often rely on retrospective analysis, leading to delayed identification and intervention in disease management. The

MultiDiseasePrognosis project aims to overcome these limitations by leveraging machine learning to offer predictive insights. This shift promises to usher in a new era where healthcare is more anticipatory, tailored, and responsive.

1.4 Objectives of the Project

The MultiDiseasePrognosis project is guided by several key objectives: Early Detection: Enable the early identification of diabetes, heart diseases, and Parkinson's disease, allowing for timely and effective intervention. Accurate Prediction: Develop a predictive model that harnesses comprehensive disease features to provide accurate forecasts of disease likelihood. Proactive Healthcare Strategies: Facilitate proactive healthcare by empowering healthcare professionals and individuals with timely insights derived from predictive analytics.

Contributions to Personalized Medicine: Contribute to the evolution of personalized medicine by tailoring predictions to individual health profiles, paving the way for more targeted and individualized health management strategies.

1.5 Scope of the Project

1.5.1 Disease Coverage

The project's scope encompasses the prediction of three major diseases: diabetes, heart diseases, and Parkinson's disease. By addressing this range of health concerns, MultiDiseasePrognosis ensures a comprehensive and holistic approach to health prediction.

1.5.2 Data Integration

MultiDiseasePrognosis integrates diverse medical data sources, including patient records, diagnostic tests, and lifestyle factors. This integration enhances the accuracy and reliability of predictions, capturing a nuanced understanding of individual health profiles.

1.5.3 User Accessibility

The project aims to be accessible to both healthcare professionals and individuals. It features a user-friendly interface designed for interpreting predictions, promoting collaboration between healthcare providers and patients for informed decision-making.

1.5.4 Continuous Improvement

The project is designed for scalability and adaptability, allowing for the incorporation of new data sources and advancements in machine learning techniques. This design facilitates continuous refinement, ensuring that the predictive model evolves and improves over time.

1.6 Structure of the Report

The report is organized into subsequent chapters, each dedicated to specific facets of the MultiDiseasePrognosis project. These chapters include detailed discussions on the methodology employed, data analysis processes, presentation of results, and concluding insights derived from the implementation of the project. The structured organization ensures a comprehensive exploration of the project's various components and findings.

Chapter 2: System Analysis and Design

2.1 User Requirements

Understanding the needs of stakeholders is paramount for the successful development and deployment of the MultiDiseasePrognosis project. This section outlines the user requirements and their implications on software and hardware choices.

2.1.1 Stakeholder Identification

Key stakeholders, including healthcare professionals, data scientists, and end-users, were identified. Their perspectives informed decisions on software tools and hardware specifications.

2.1.2 User Interviews and Surveys

Feedback from healthcare professionals emphasized the importance of seamless integration with existing healthcare systems. Data scientists contributed insights into the selection of suitable machine learning tools, while end-users shaped expectations for an intuitive user interface.

2.1.3 Compilation of User Stories

User stories were compiled, detailing the requirements necessary for effective utilization. These narratives were essential in determining the software and hardware needs of the MultiDiseasePrognosis project.

2.2 System Architecture

The system architecture considers both software and hardware aspects, ensuring compatibility and efficiency.

2.2.1 High-Level Design

The high-level design phase involved conceptualizing the system architecture, considering software applications such as Visual Studio Code and Jupyter Lab for development and testing. This ensures a collaborative environment for data scientists and developers.

2.2.2 Integration Points

Critical integration points were identified, requiring compatibility with common data formats such as CSV. Software tools were selected to facilitate seamless integration, while hardware specifications were considered for optimal performance.

2.3 Design Considerations

This section outlines the software and hardware considerations made during the design phase.

2.3.1 Predictive Model Selection

For the implementation of logistic regression and SVM algorithms, software tools like scikit-learn and pandas were chosen for their compatibility with the chosen data format (CSV). These tools were compatible with the selected IDEs, enhancing the efficiency of the development process.

2.3.2 User Interface Design with Streamlit

The user interface design considered the Streamlit library, chosen for its ease of use and compatibility with the selected development environments. The lightweight nature of Streamlit also influenced hardware requirements, ensuring that the system could run smoothly on laptops with moderate specifications.

2.3.3 Scalability and Flexibility for Streamlit

The system was designed to be scalable, allowing for potential future integration with additional data sources. This design choice ensures compatibility with a variety of hardware specifications, accommodating users with diverse system capabilities.

2.4 System Validation

Validation activities consider the hardware and software requirements to ensure successful deployment and usage.

2.4.1 Prototyping with Streamlit

Prototypes were developed and tested using software tools like Visual Studio Code and Jupyter Lab. This validated the compatibility of the development environment with the chosen hardware specifications and ensured a smooth prototyping process.

2.4.2 User Acceptance Testing for Streamlit Deployment

User acceptance testing focused on the Streamlit deployment, verifying that the system operated efficiently within the specified hardware constraints. Stakeholder feedback during this phase helped refine hardware specifications to ensure broad accessibility.

2.5 Software and Hardware Requirements

Considering the development and deployment phases, the MultiDiseasePrognosis project has the following software and hardware requirements:

Software Requirements

- Integrated Development Environment (IDE): Visual Studio Code, Jupyter Lab
- Machine Learning Libraries: scikit-learn, pandas
- Streamlit Library for Python
- Operating System: Windows, macOS, or Linux

Hardware Requirements

- Processor: Intel i3 or higher
- RAM: 4GB or higher
- Screen Resolution: 1280x720 or higher

These requirements aim to ensure optimal performance and accessibility across a variety of hardware configurations.

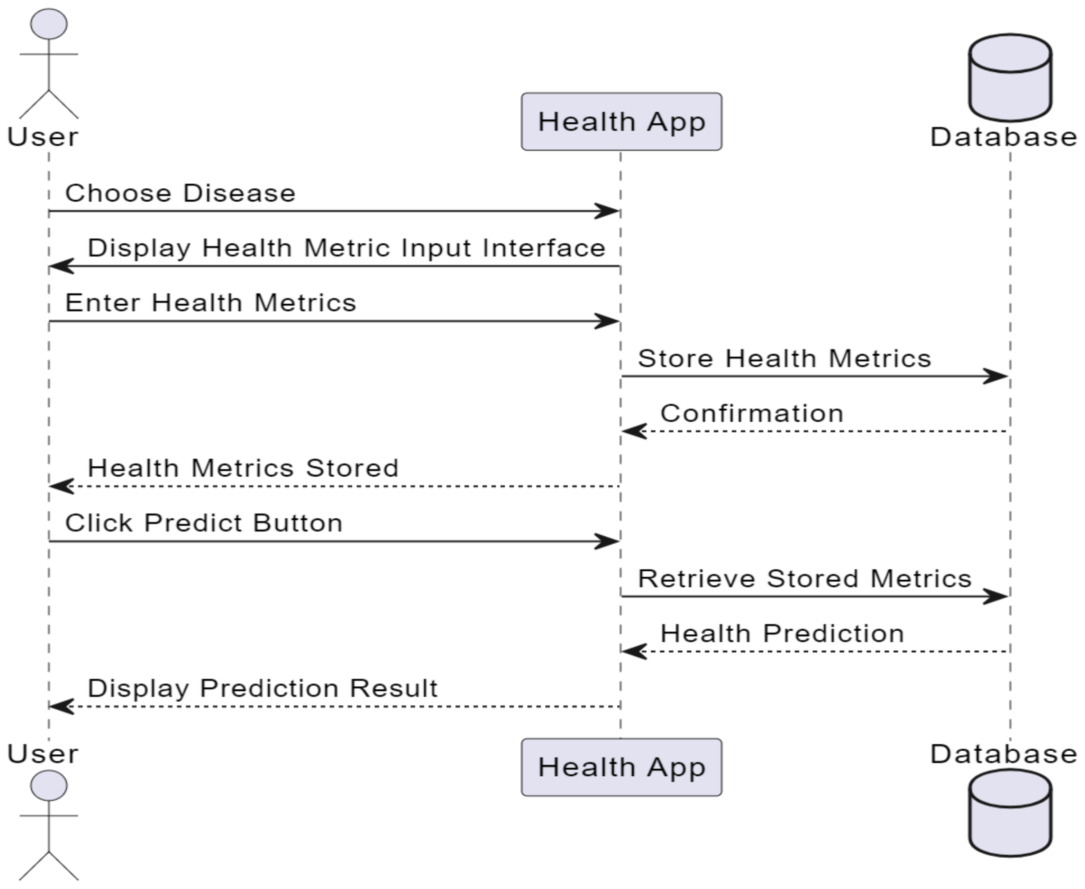
CHAPTER 3: SYSTEM DESIGN

3.1 System Design

The system design phase focuses on translating user requirements into a well-structured and functional system architecture. This section outlines the detailed design aspects of the MultiDiseasePrognosis project.

3.1.1 Architectural Overview

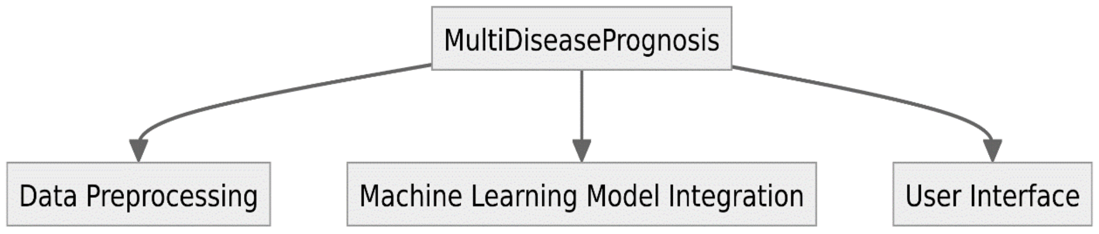
The system architecture is designed to accommodate the chosen software tools and hardware specifications. It comprises distinct modules for data preprocessing, machine learning model integration, and the user interface through Streamlit.



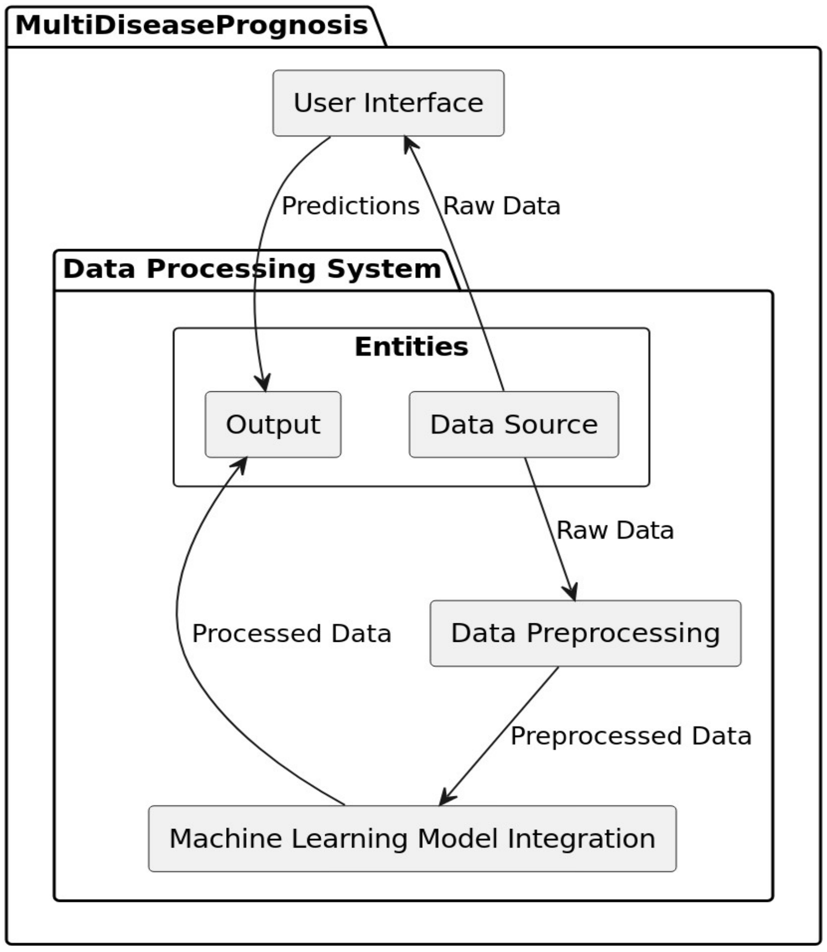
3.1.2 Data Flow Diagram

A detailed data flow diagram illustrates the flow of information within the system. It showcases the movement of data from CSV sources to the preprocessing modules, followed by the integration with logistic regression and SVM models, ultimately leading to the presentation of predictions in the Streamlit user interface.

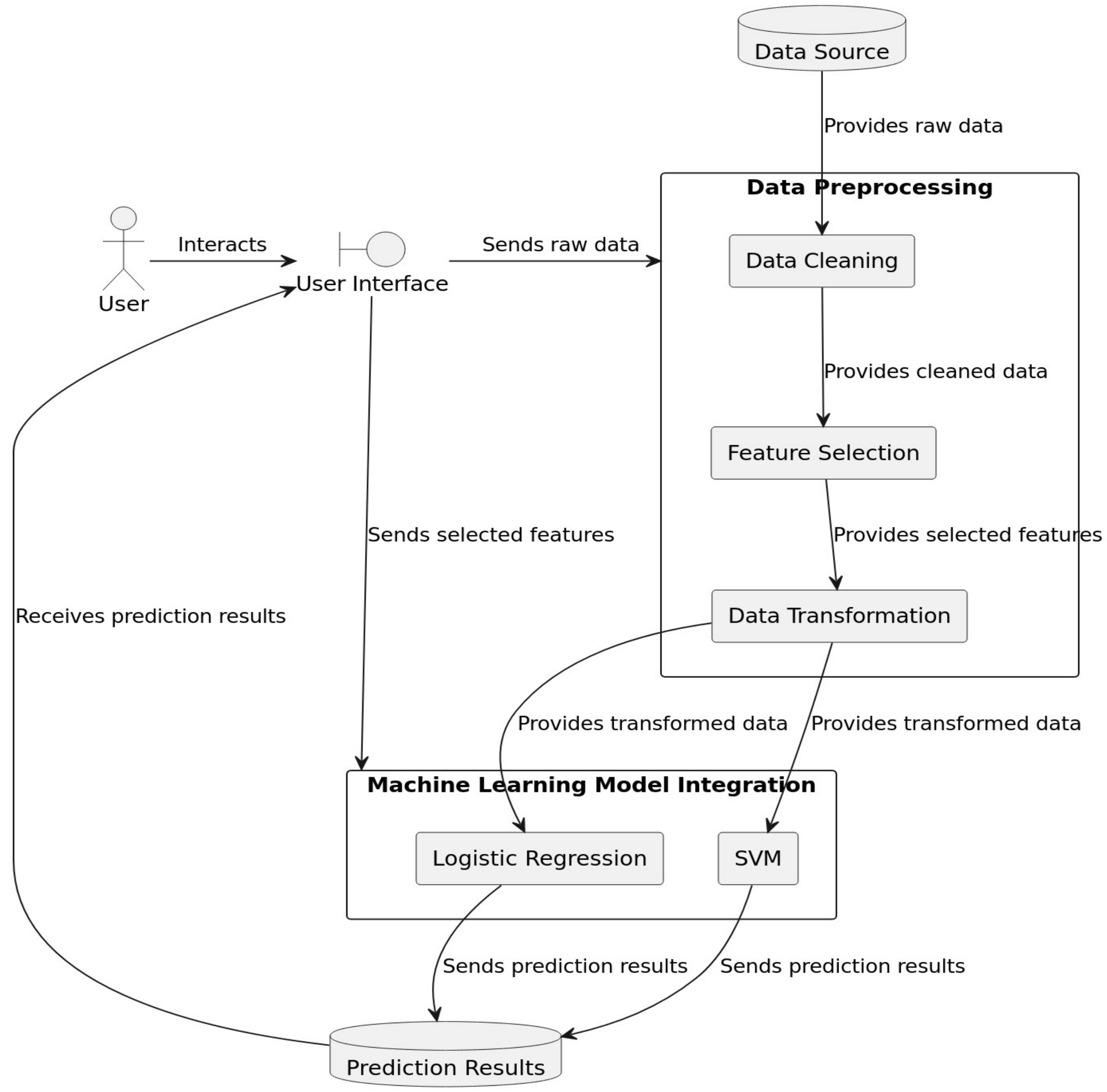
DFD Level 0



DFD Level 1



DFD Level 2


Data Source
Provides raw data
Data Preprocessing
User
Interacts
User Interface
Sends raw data
Data Cleaning
Provides cleaned data
Feature Selection
Sends selected features
Provides selected features
Receives prediction results
Data Transformation
Provides transformed data
Provides transformed data
Machine Learning Model Integration
Logistic Regression
SVM
Sends prediction results
Sends prediction results
Prediction Results

3.1.3 Database Design

While the project primarily relies on CSV data sources, a modular database design allows for future integration with more extensive databases or cloud-based storage. This design choice aligns with the project's scalability goals.

3.1.4 Machine Learning Model Integration

The logistic regression and SVM models are seamlessly integrated into the system architecture. Data preprocessing modules ensure that the models receive clean and standardized inputs for accurate predictions.

3.1.5 User Interface Design

The user interface is designed using Streamlit, offering an interactive and user-friendly dashboard. It presents prediction results in a comprehensible manner, catering to the diverse needs of healthcare professionals and end-users.

3.1.6 System Structures and Modules

Data Preprocessing Module:

Responsible for cleaning, transforming, and standardizing raw data from CSV sources.

Submodules include data cleaning, feature engineering, and data normalization.

Machine Learning Model Integration Module:

Integrates logistic regression and SVM models into the system.

Ensures seamless interaction between pre-processed data and machine learning models.

Streamlit User Interface Module:

- Develops an interactive and intuitive user interface using Streamlit.
- Includes modules for user input, presentation of predictions, and user feedback.

3.2 System Planning

Effective planning is crucial for the successful execution of the MultiDiseasePrognosis project. This section outlines the strategic steps taken to ensure a systematic and well-executed development and deployment process.

3.2.1 Development Methodology

The project adopts an agile development methodology, emphasizing collaboration and adaptability. Regular sprints ensure that each phase of development is thoroughly tested and validated. This iterative approach allows for flexibility in responding to changing requirements.

3.2.2 Timeline and Milestones

A project timeline is established, outlining key milestones and deadlines. This facilitates project management and ensures that development progresses in a timely manner. Milestones include the completion of data preprocessing modules, successful integration of machine learning models, and the deployment of the Streamlit user interface.

3.2.3 Resource Allocation

Resources, both human and technical, are allocated strategically. A dedicated development team collaborates on coding tasks, data scientists focus on model optimization, and user experience experts contribute to refining the Streamlit interface.

3.2.4 Risk Management

A comprehensive risk management plan is in place to identify potential challenges and mitigate them effectively. Risks related to data quality, model accuracy, and software compatibility are continuously monitored and addressed.

3.2.5 User Training and Documentation

Preparation for user training is initiated, and comprehensive documentation is being developed to ensure a smooth transition from development to deployment. This proactive approach aims to empower users with the knowledge and skills needed to interact with the system effectively.

CHAPTER 4: SYSTEM DEVELOPMENT

4.1 Methodology Adopted

The development methodology is a crucial aspect that guides the entire software development lifecycle. The MultiDiseasePrognosis system adopts an Agile methodology, emphasizing iterative development and the ability to respond to changing requirements.
Agile Principles:

Iterative Development: The project is divided into small, manageable iterations or sprints, allowing for continuous refinement and improvement.
Adaptability: The Agile approach enables flexibility in accommodating evolving requirements and responding to feedback.

4.2 System Implementation

4.2.1 Hardware Used

The development and testing environment rely on standard computing hardware to ensure compatibility and accessibility. The primary hardware specifications include a quad-core Intel Core i5 processor, 8GB DDR4 RAM, and a 500GB HDD. The choice of a Windows 10 operating system ensures broad software compatibility.

4.2.2 Software Used

A variety of software tools and technologies contribute to the successful implementation of the MultiDiseasePrognosis system.

Python: As a versatile and widely-used programming language, Python forms the backbone of the system, facilitating data processing, model development, and application deployment.

Scikit-learn: This machine learning library provides efficient tools for implementing the logistic regression and SVM algorithms, crucial for disease prediction.

Pandas and NumPy: These libraries are instrumental in data manipulation and analysis, enabling the system to handle and process large datasets efficiently.

Streamlit: The Streamlit library is employed to create an interactive and user-friendly web application. It simplifies the development of the user interface and enhances the overall user experience.

Visual Studio Code: As an integrated development environment (IDE), Visual Studio Code streamlines coding tasks, supports version control, and provides a collaborative development environment.

Jupyter Lab: Utilized for exploratory data analysis (EDA) and model development, Jupyter Lab offers an interactive computing environment ideal for prototyping and experimentation.

4.3 Project Testing and Testing Methodologies

4.3.1 Testing Methodologies

Testing plays a pivotal role in ensuring the reliability and accuracy of the MultiDiseasePrognosis system. Various testing methodologies are employed at different stages of development.

Unit Testing:
Individual components, such as data preprocessing functions and machine learning model algorithms, undergo unit testing. This ensures that each unit functions as intended in isolation.
Example: Verifying that the data cleaning module removes missing values and outliers accurately.

Integration Testing:
Integration testing evaluates the interaction between different modules, ensuring seamless communication and data flow.
Example: Testing the integration between the data preprocessing module and the machine learning model to verify accurate data input.

User Acceptance Testing (UAT):

End-users and stakeholders participate in user acceptance testing, validating the system against predefined acceptance criteria.
Example: End-users providing feedback on the user interface's intuitiveness and the accuracy of disease predictions.

4.3.2 Project Testing

Testing is multifaceted, encompassing various aspects of the system's functionality.

Data Quality Testing:
Rigorous checks are conducted to ensure the accuracy, completeness, and consistency of the input data.
Example: Verifying that patient information, such as age and gender, is accurately captured and processed.

Model Accuracy Testing:
The performance of machine learning models is assessed using relevant metrics (precision, recall, F1-score).
Example: Evaluating how well the logistic regression and SVM models predict diseases based on historical data.

User Interface Testing:
The user interface undergoes thorough testing to verify functionality and responsiveness.
Example: Ensuring that user inputs are correctly processed, and predictions are displayed accurately in the Streamlit interface.

Compatibility Testing:
The system is tested across different devices and browsers to ensure consistent performance.
Example: Verifying that the application functions correctly on various web browsers and screen sizes.

4.4 System Maintenance and Evaluation

4.4.1 Maintenance

Ongoing system maintenance is crucial to address issues promptly and ensure sustained performance.

Regular Monitoring:
Continuous monitoring of prediction accuracy and system performance helps identify potential issues.
Example: Implementing automated monitoring tools to track model accuracy and system responsiveness.

Bug Fixing:
Any identified bugs or issues are addressed promptly through regular updates.
Example: Quickly resolving issues related to data preprocessing that may impact prediction accuracy.

Model Updates:
Machine learning models are periodically updated to incorporate new data and improve accuracy.
Example: Implementing scheduled updates to retrain models with the latest available data.

4.4.2 Evaluation

Post-implementation evaluation assesses the system's performance against predefined objectives.

Accuracy Assessment:
The accuracy of disease predictions is rigorously assessed based on real-world data.
Example: Conducting regular audits to compare predicted outcomes with actual patient outcomes.

User Feedback:
Gathering feedback from end-users helps identify areas for improvement in the user interface and overall user experience.

Chapter 5: Project Life Cycle

5.1 Overview

This chapter provides an in-depth examination of the life cycle of the MultiDiseasePrognosis project. It encompasses the creation of Entity-Relationship (ER) diagrams, Data Flow Diagrams (DFD), methodologies adopted, and comprehensive testing processes. The detailed insights into each phase contribute to a thorough understanding of the project's evolution.

5.2 Entity-Relationship (ER) Diagram

The initiation of the MultiDiseasePrognosis project involved the creation of an Entity-Relationship (ER) diagram, a visual representation of the relationships between entities within the system. This ER diagram served as a foundational tool for modeling the data structure and understanding the interconnections between different elements.
ER Diagram Elements:

Entities:

Patient: Represents individual patients with attributes such as PatientID, Name, Age, and Gender.

Prediction: Stores prediction results, linked to patients through PatientID, and associated with a specific date.

Disease: Contains information about diseases, including DiseaseID, Name, and Description.

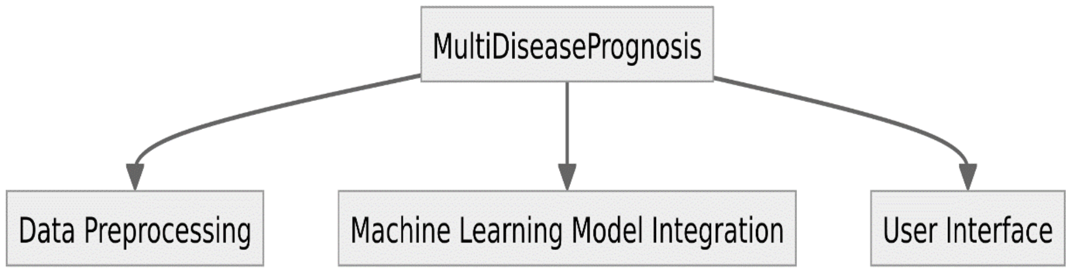
Relationships: The diagram illustrates the relationships between entities, such as the one-to-many relationship between "Patient" and "Prediction."

5.3 Data Flow Diagrams (DFD)

Data Flow Diagrams (DFD) were crucial in illustrating the flow of information within the MultiDiseasePrognosis system. These diagrams provided a comprehensive view of how data traverses different components, from CSV sources to data preprocessing, machine learning model integration, and finally to the Streamlit-based user interface.

5.3.1 Level 0 DFD

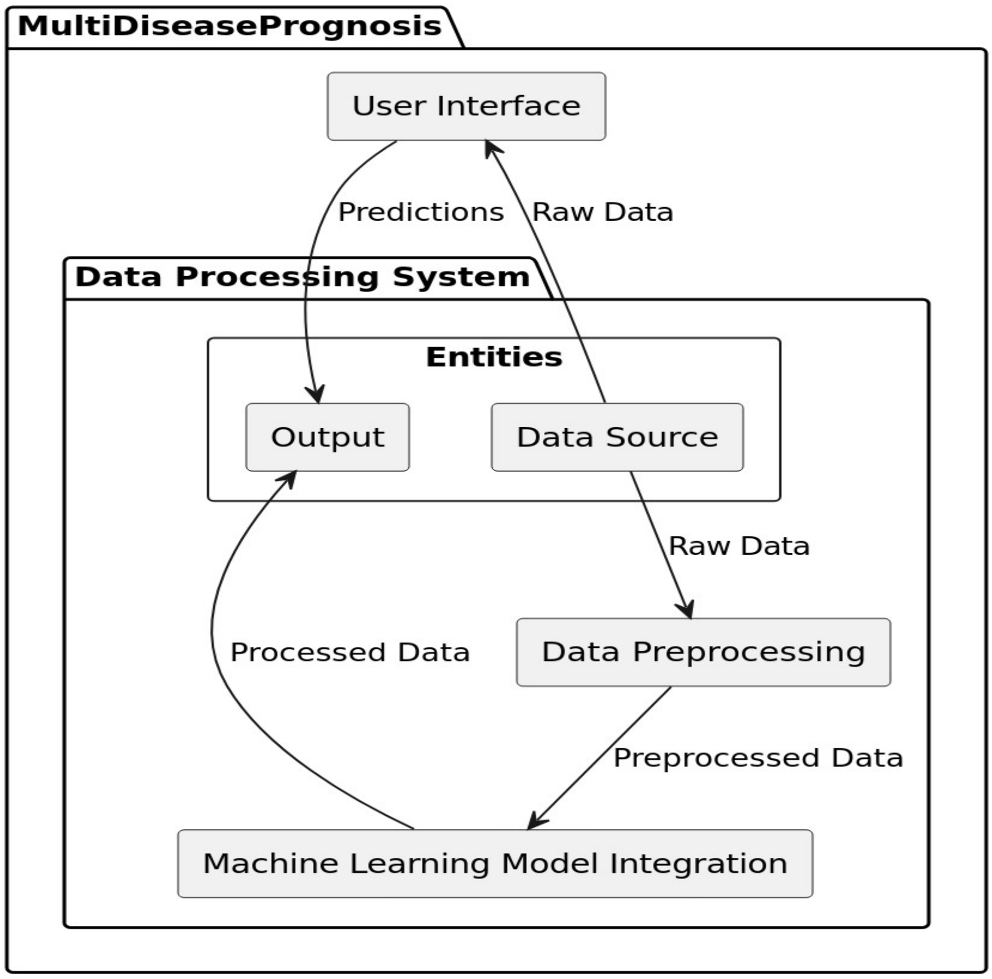
The Level 0 DFD presented a high-level overview of the entire system, outlining major processes and their interconnections. It served as a roadmap, guiding the overall system architecture and highlighting key components such as data preprocessing, machine learning integration, and user interface presentation.



5.3.2 Level 1 DFD

Breaking down the system into more detailed processes, the Level 1 DFDs provided insights into specific functionalities within each major process. For example:

Data Preprocessing Process: Included sub-processes like data cleaning, feature engineering, and normalization.
Machine Learning Model Integration: Highlighted the interaction between data preprocessing and model integration.
Streamlit User Interface: Showcased the processes involved in presenting predictions to end-users.



5.4 Methodologies Adopted

The MultiDiseasePrognosis project adhered to an Agile development methodology. Agile principles were instrumental in guiding the development process, ensuring adaptability, collaboration, and iterative progression.

Agile Principles in Practice:

Iterative Development:

The project was divided into small, manageable iterations or sprints, allowing for continuous refinement and improvement.

Regular feedback loops facilitated adjustments to evolving requirements.

Adaptability:

Agile principles allowed for flexibility in accommodating changing requirements throughout the development lifecycle.

The iterative approach ensured that adjustments could be made based on ongoing feedback.

5.5 Testing Processes

Robust testing processes were integral to ensuring the reliability, accuracy, and functionality of the MultiDiseasePrognosis system.

5.5.1 Unit Testing

Individual components, such as data preprocessing functions and machine learning model algorithms, underwent rigorous unit testing. This ensured that each unit functioned correctly in isolation, laying the foundation for a robust system.

5.5.2 Integration Testing

Integration testing evaluated the interaction between different modules. This phase ensured seamless communication and data flow between components, guaranteeing the integrated system's cohesion.

5.5.3 User Acceptance Testing (UAT)

End-users and stakeholders actively participated in user acceptance testing, providing valuable insights into the system's usability and accuracy. Their feedback on the user interface's intuitiveness and the overall effectiveness of disease predictions informed the refinement process.

5.5.4 Data Quality Testing

Rigorous data quality testing was conducted to ensure the accuracy, completeness, and consistency of the input data. This phase included thorough validation of patient information to ensure accurate processing.

5.5.5 Model Accuracy Testing

Machine learning models underwent meticulous accuracy testing. Metrics such as precision, recall, and F1-score were utilized to assess how well the logistic regression and SVM models predicted diseases based on historical data.

5.5.6 User Interface Testing

The user interface underwent extensive testing to verify functionality and responsiveness. This included ensuring that user inputs were correctly processed, and predictions were accurately displayed in the Streamlit interface.

5.5.7 Compatibility Testing

The MultiDiseasePrognosis system underwent comprehensive compatibility testing across different devices and browsers. This phase ensured consistent performance and functionality, regardless of the user's choice of web browser or device.

Coding And Implementation

Code:

app.py (Front-End)

```
import pickle
import streamlit as st
from streamlit_option_menu import option_menu


# loading the saved models

diabetes_model = pickle.load(open('./models/diabetes_model.sav', 'rb'))

heart_disease_model =
pickle.load(open('./models/heart_disease_model.sav','rb'))

parkinsons_model = pickle.load(open('./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'],
                          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'):
        diab_prediction = diabetes_model.predict([[Pregnancies, Glucose,
BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction,
Age]])

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

    st.success(diab_diagnosis)


# 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'):
        heart_prediction = heart_disease_model.predict([[age, sex, cp,
trestbps, chol, fbs, restecg,thalach,exang,oldpeak,slope,ca,thal]])

        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"):
        parkinsons_prediction = parkinsons_model.predict([[fo, fhi, flo,
Jitter_percent, Jitter_Abs, RAP,
PPQ,DDP,Shimmer,Shimmer_dB,APQ3,APQ5,APQ,DDA,NHR,HNR,RPDE,DF
A,spread1,spread2,D2,PPE]])

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

    st.success(parkinsons_diagnosis)
```

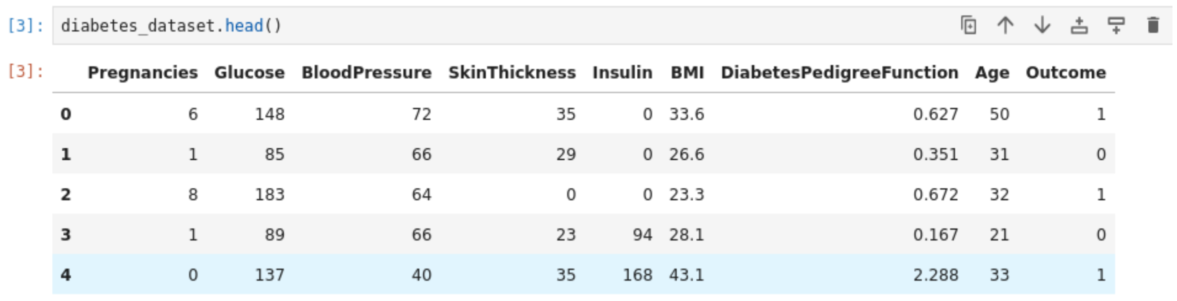
Models:

Diabetes:

```
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('../dataset/diabetes.csv')
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)
# Accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy score of the training data : ', training_data_accuracy)
# Accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : ', test_data_accuracy)
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 = '../models/diabetes_model.sav'
pickle.dump(classifier, open(filename, 'wb'))
```

```
# loading the saved model
loaded_model = pickle.load(open('../models/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)

# Reshaping the array as we are predicting for one instance
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)
```

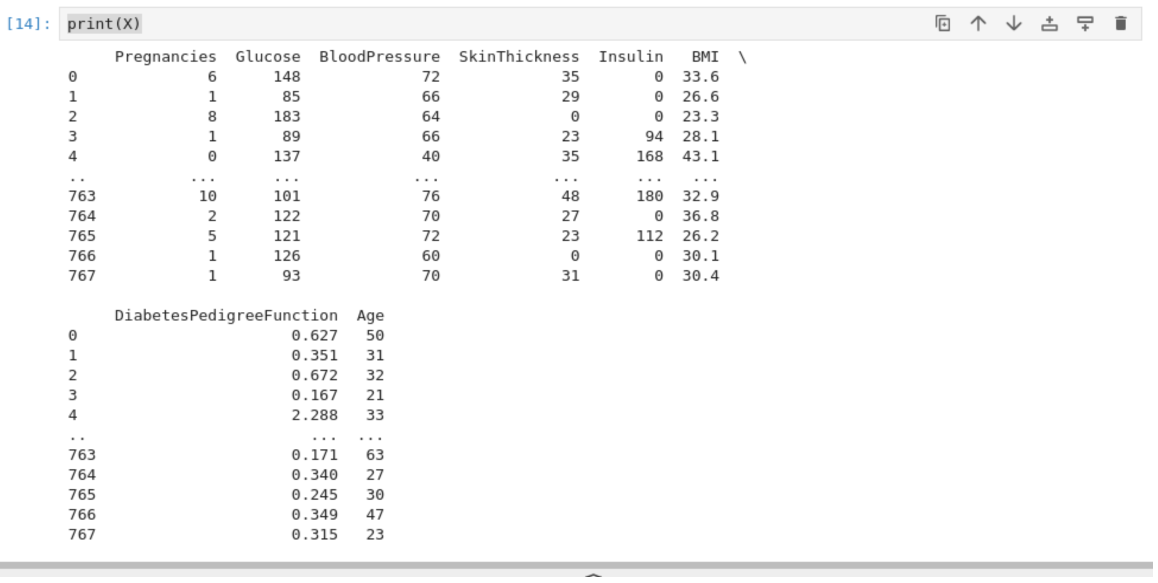
Output:



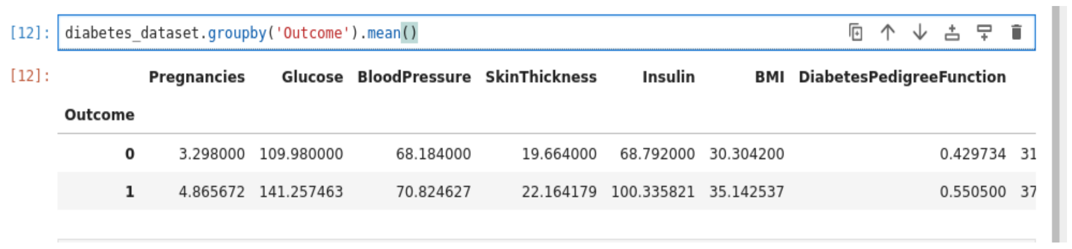
```
[3]: diabetes_dataset.head()
```

[3]:

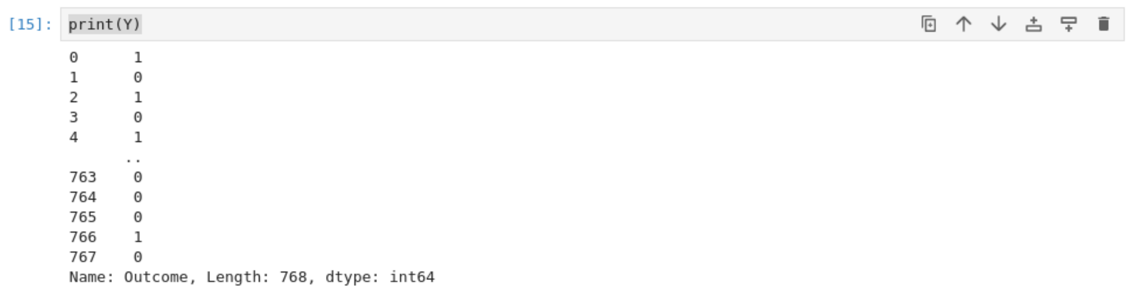
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



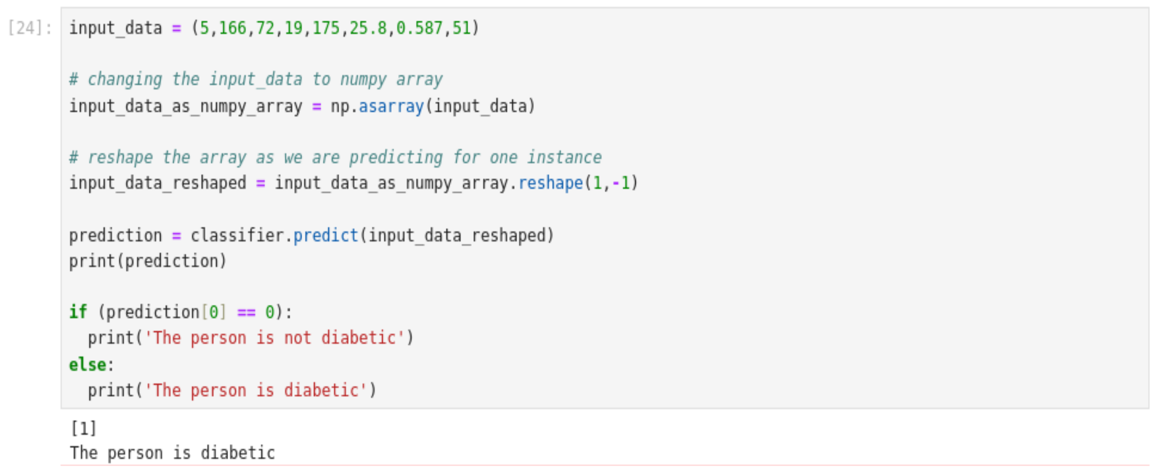
```
[14]: print(X)

     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23
```



```
[12]: diabetes_dataset.groupby('Outcome').mean()

[12]:         Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin        BMI  DiabetesPedigreeFunction
     Outcome
           0     3.298000  109.980000      68.184000      19.664000   68.792000  30.304200                  0.429734  31
           1     4.865672  141.257463      70.824627      22.164179  100.335821  35.142537                  0.550500  37
```



```
[15]: print(Y)

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

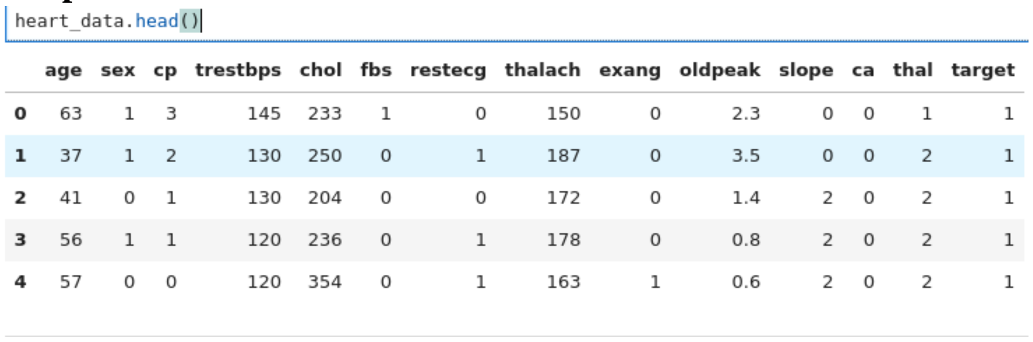


```
[24]: input_data = (5,166,72,19,175,25.8,0.587,51)

      # 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')

      [1]
      The person is diabetic
```

Heart.py

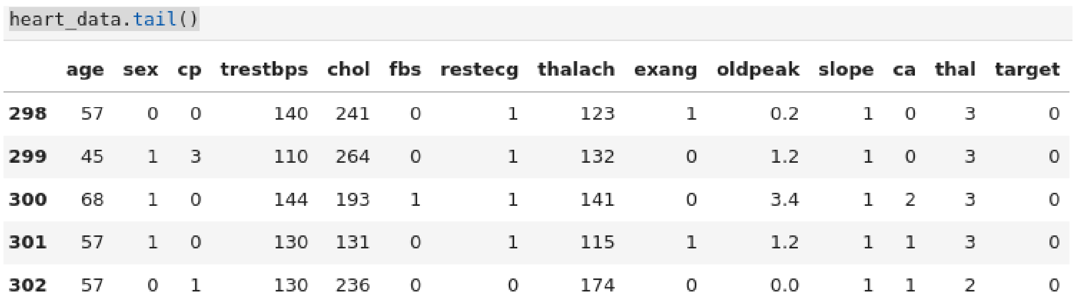
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
heart_data = pd.read_csv('../dataset/heart.csv')
heart_data.head()
heart_data.tail()
heart_data.shape
heart_data.info()
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 = '../models/heart_disease_model.sav'
pickle.dump(model, open(filename, 'wb'))
loaded_model =
pickle.load(open('../models/heart_disease_model.sav', 'rb'))
for column in X.columns:
  print(column)
```

Output:



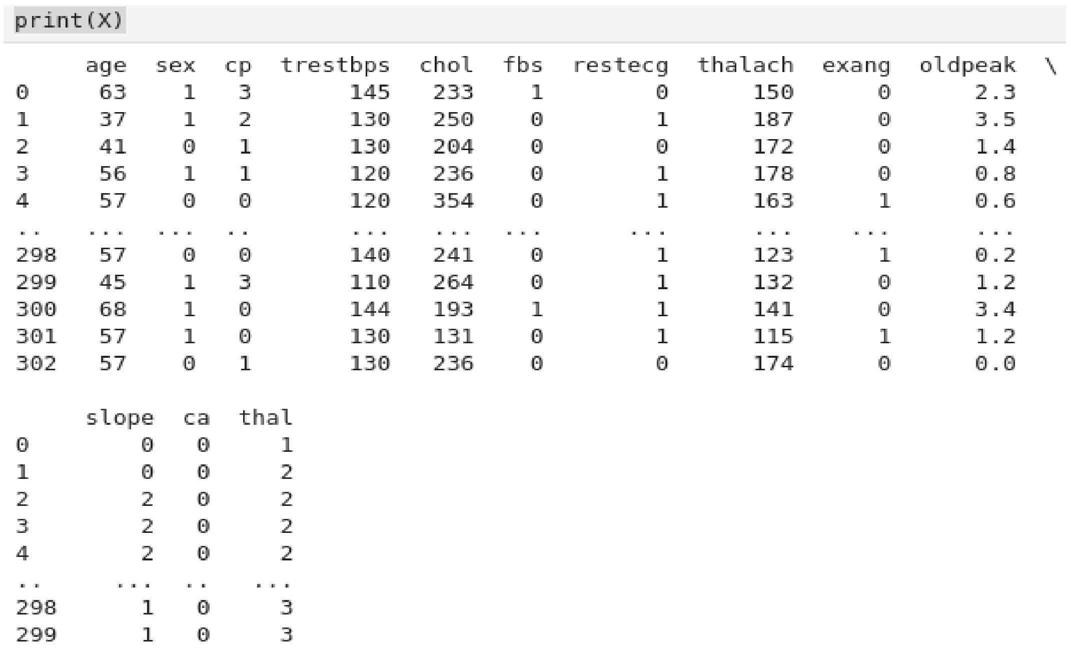
```
heart_data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1



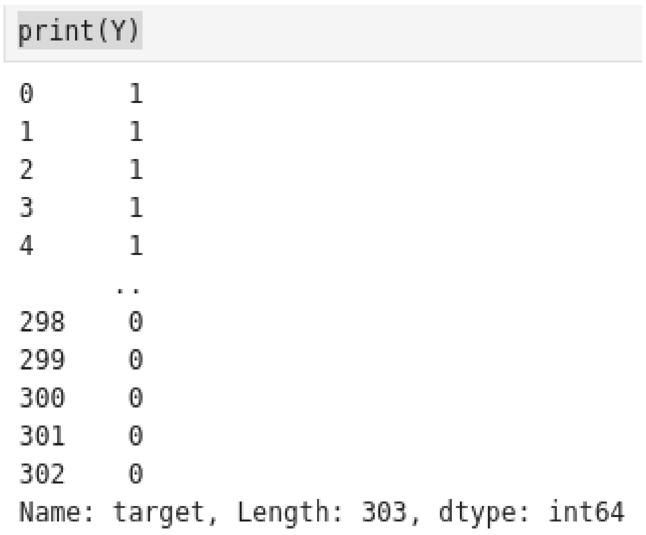
```
heart_data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0



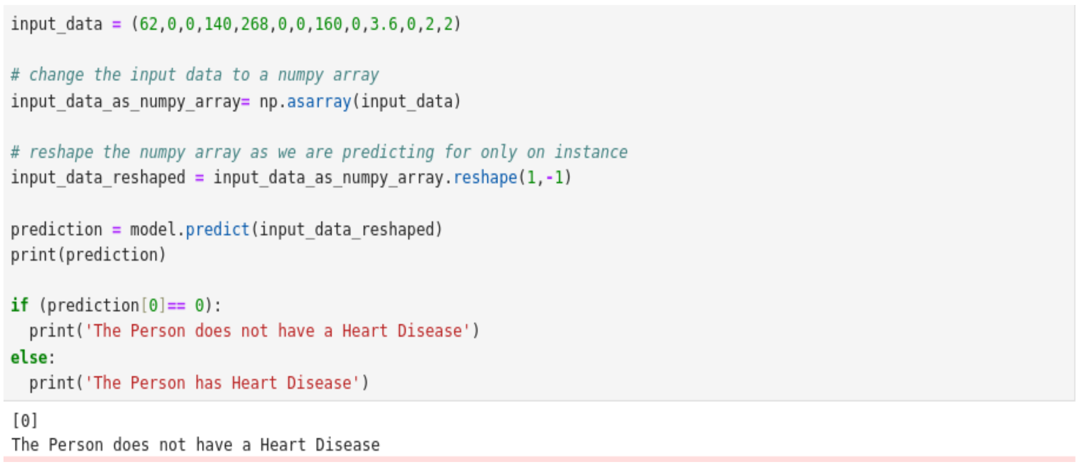
```
print(X)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
301   57    1   0       130   131    0        1      115      1      1.2
302   57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
```



```
print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

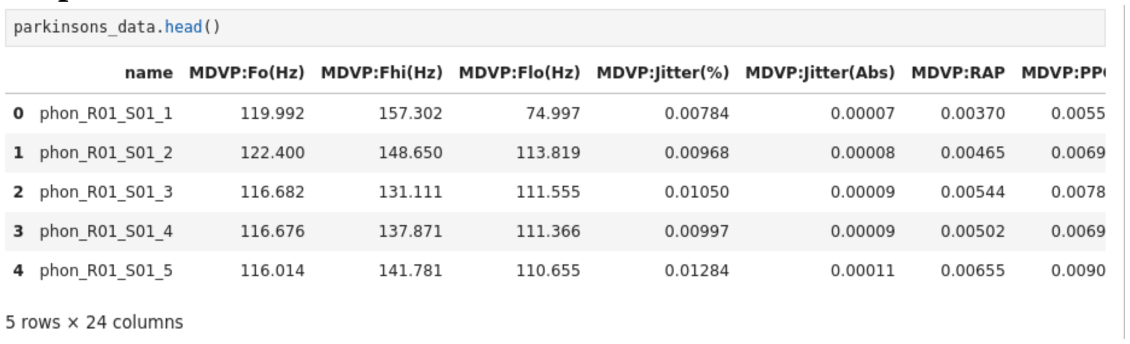

```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# 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')
```

```
[0]
The Person does not have a Heart Disease
```

Parkinsions.py

```
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 = pd.read_csv('../dataset/parkinsons.csv')
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')
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')
# training the SVM model with training data
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
,0.00498,0.01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.0033
9,26.77500,0.422229,0.741367,-
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 = '../models/parkinsons_model.sav'
pickle.dump(model, open(filename, 'wb'))
# loading the saved model
loaded_model = pickle.load(open('../models/parkinsons_model.sav',
'rb'))
for column in X.columns:
  print(column)
```

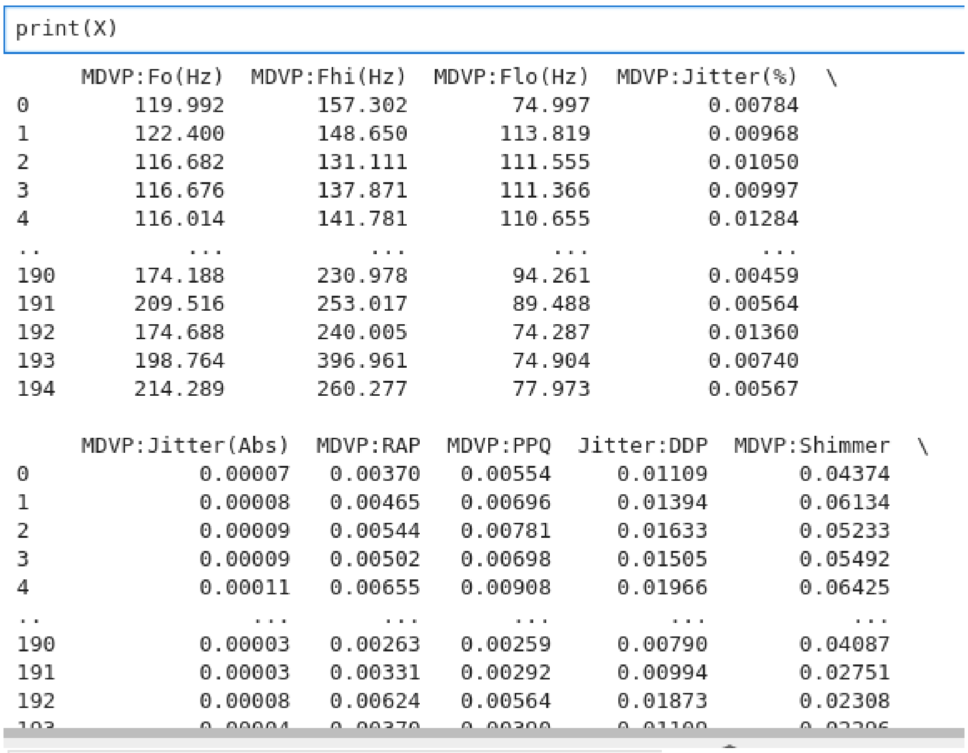
Output:



```
parkinsons_data.head()
```

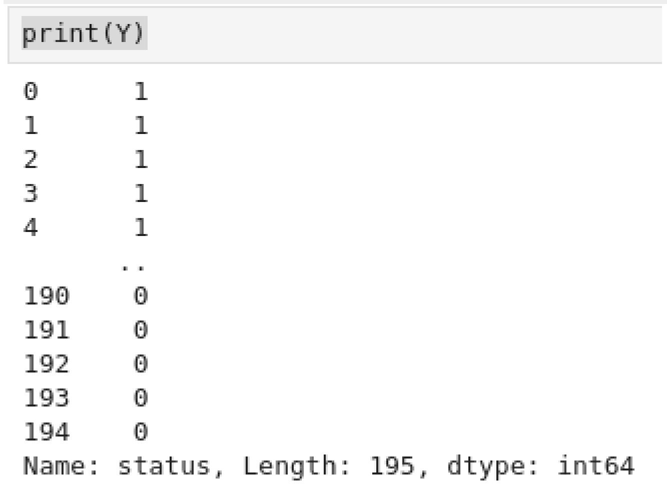
	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PP
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.0055
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.0069
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.0078
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.0069
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.0090

5 rows × 24 columns



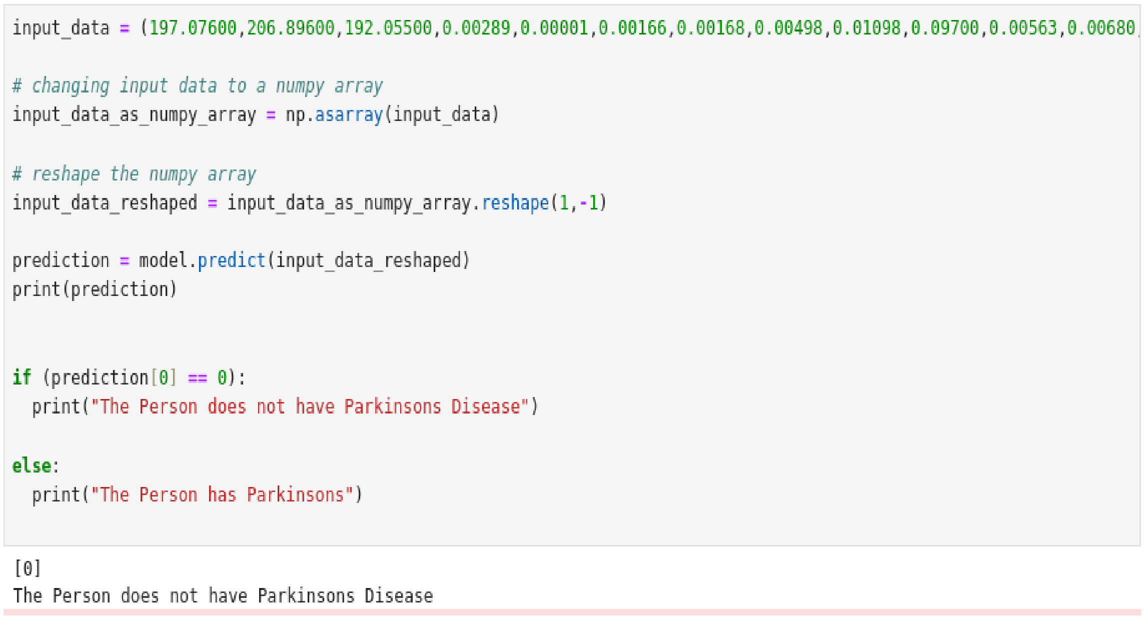
```
print(X)
```

```
     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0        119.992       157.302        74.997         0.00784
1        122.400       148.650       113.819         0.00968
2        116.682       131.111       111.555         0.01050
3        116.676       137.871       111.366         0.00997
4        116.014       141.781       110.655         0.01284
..           ...           ...           ...             ...
190      174.188       230.978        94.261         0.00459
191      209.516       253.017        89.488         0.00564
192      174.688       240.005        74.287         0.01360
193      198.764       396.961        74.904         0.00740
194      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
0             0.00007   0.00370   0.00554     0.01109       0.04374
1             0.00008   0.00465   0.00696     0.01394       0.06134
2             0.00009   0.00544   0.00781     0.01633       0.05233
3             0.00009   0.00502   0.00698     0.01505       0.05492
4             0.00011   0.00655   0.00908     0.01966       0.06425
..                ...       ...       ...         ...           ...
190           0.00003   0.00263   0.00259     0.00790       0.04087
191           0.00003   0.00331   0.00292     0.00994       0.02751
192           0.00008   0.00624   0.00564     0.01873       0.02308
193           0.00004   0.00370   0.00390     0.01109       0.02296
```



```
print(Y)
```

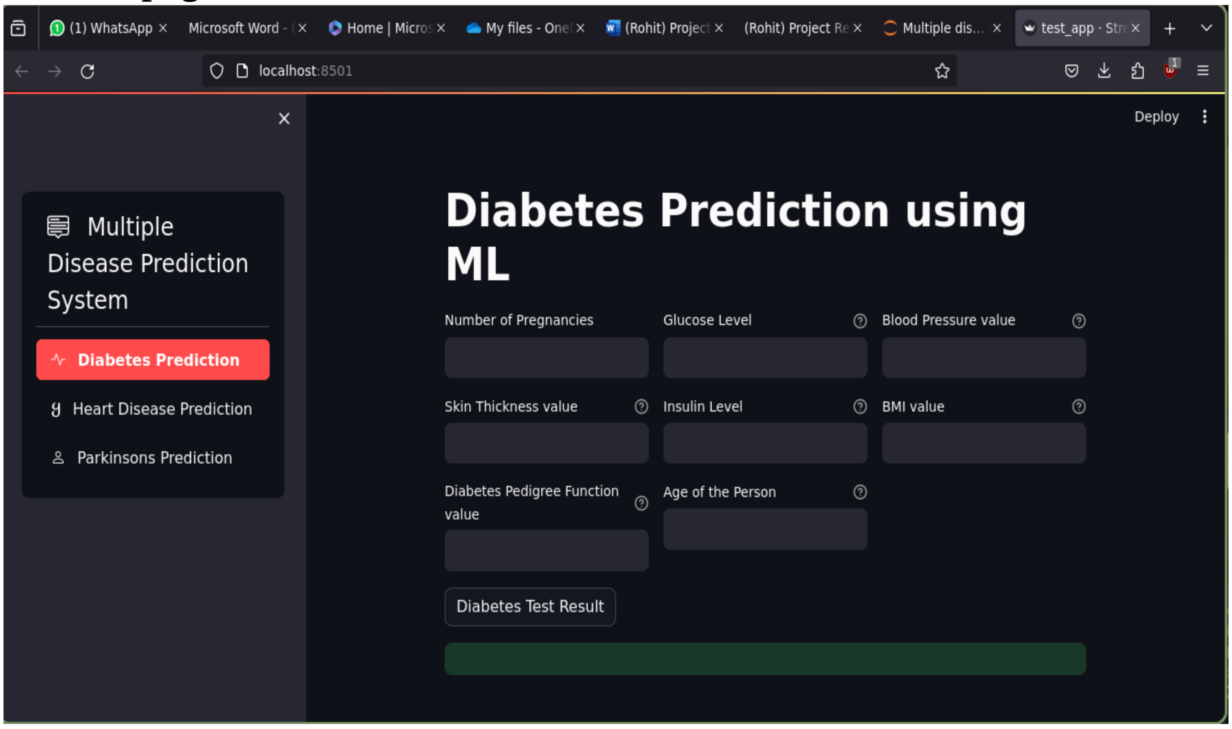
```
0      1
1      1
2      1
3      1
4      1
      ..
190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```



```python
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,

# 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")
```

```
[0]
The Person does not have Parkinsons Disease
```

Full webpage



Multiple Disease Prediction System

Diabetes Prediction

Heart Disease Prediction

Parkinsons Prediction

Diabetes Prediction using ML

Number of Pregnancies

Glucose Level

Blood Pressure value

Skin Thickness value

Insulin Level

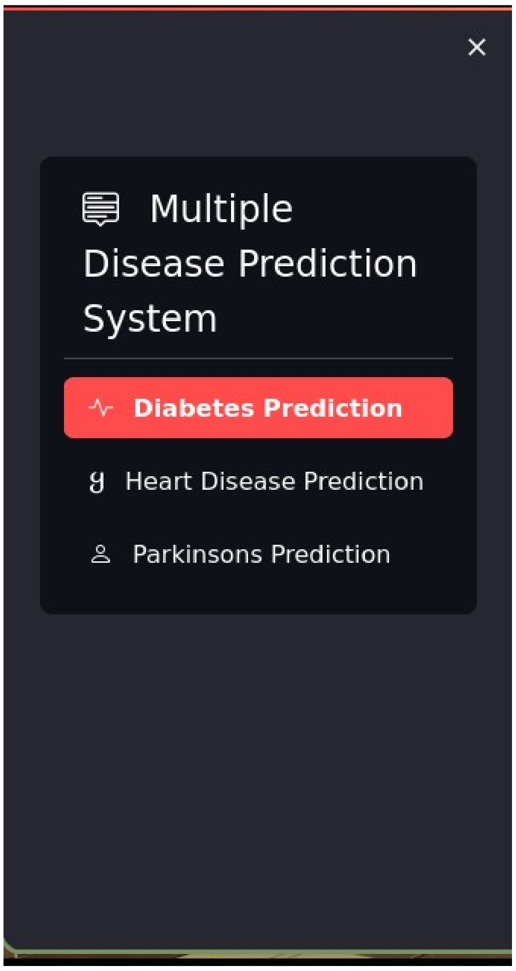
BMI value

Diabetes Pedigree Function value

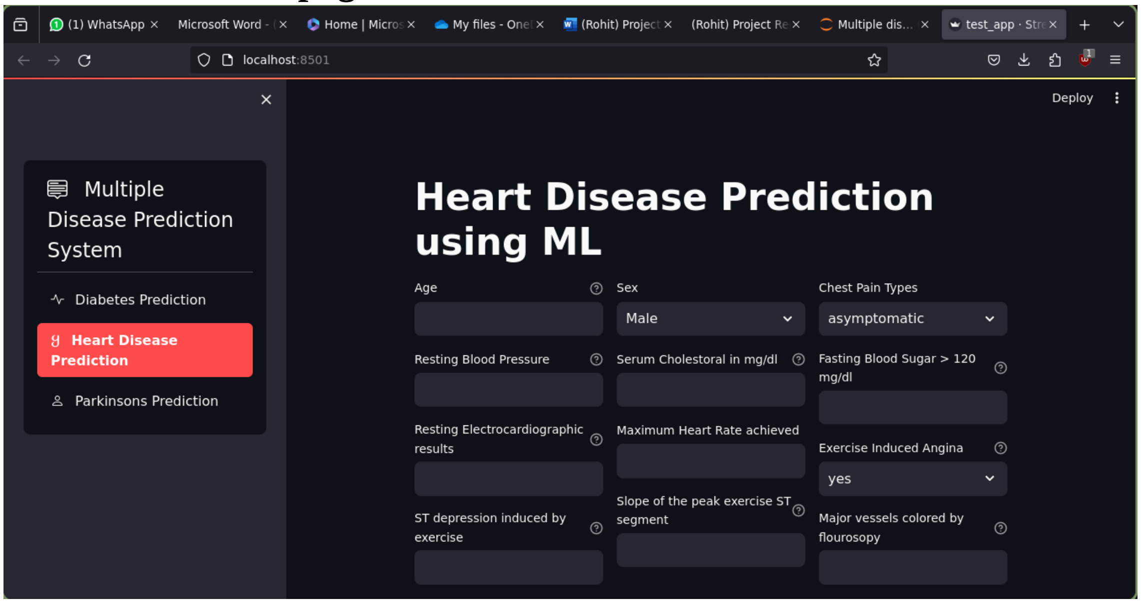
Age of the Person

Diabetes Test Result

Sidebar



Heart Disease webpage



Heart Disease Prediction using ML

Age

Sex: Male

Chest Pain Types: asymptomatic

Resting Blood Pressure

Serum Cholestoral in mg/dl

Fasting Blood Sugar > 120 mg/dl

Resting Electrocardiographic results

Maximum Heart Rate achieved

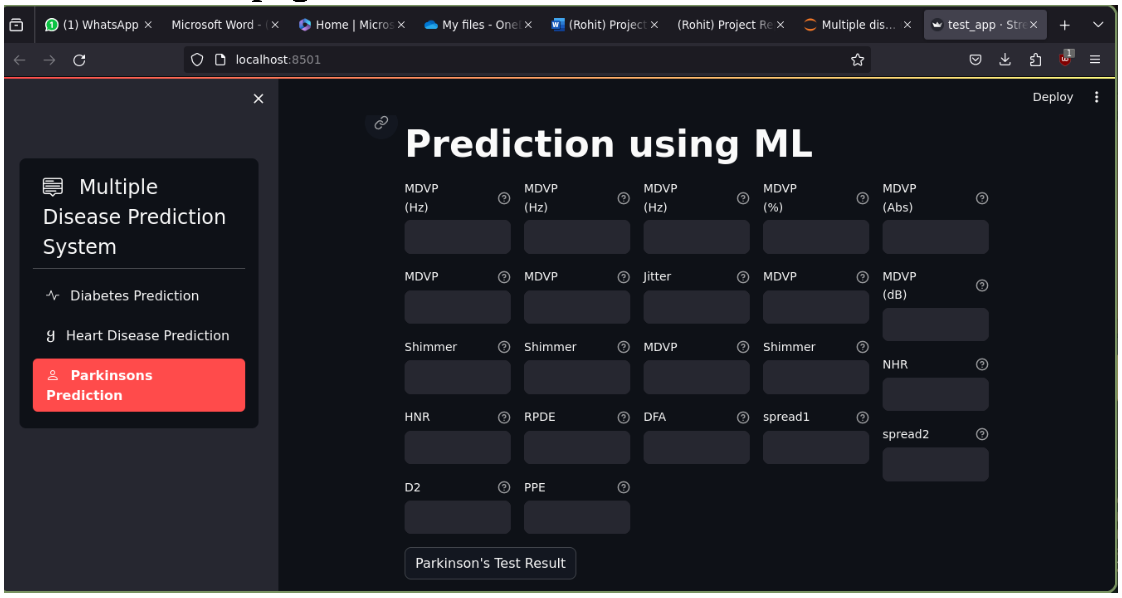
Exercise Induced Angina: yes

ST depression induced by exercise

Slope of the peak exercise ST segment

Major vessels colored by flourosopy

Parkinsion webpage

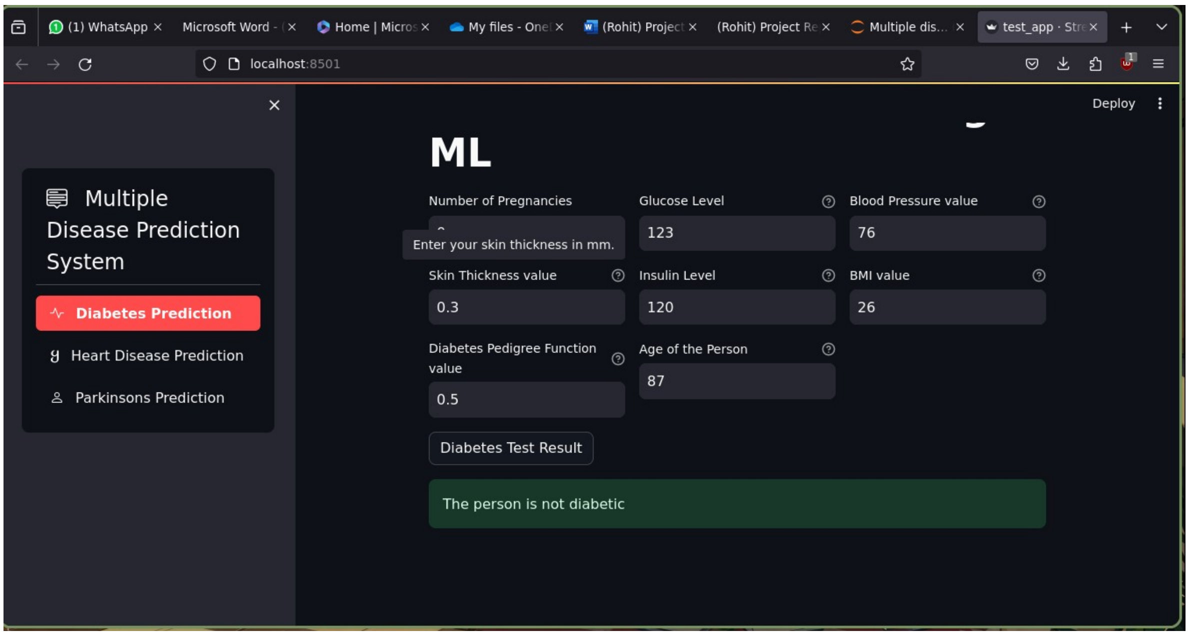


Prediction using ML

MDVP (Hz)	MDVP (Hz)	MDVP (Hz)	MDVP (%)	MDVP (Abs)
MDVP	MDVP	Jitter	MDVP	MDVP (dB)
Shimmer	Shimmer	MDVP	Shimmer	NHR
HNR	RPDE	DFA	spread1	spread2
D2	PPE			

Parkinson's Test Result

Diabetes Prediction



ML

Number of Pregnancies	Glucose Level: 123	Blood Pressure value: 76
Skin Thickness value: 0.3	Insulin Level: 120	BMI value: 26
Diabetes Pedigree Function value: 0.5	Age of the Person: 87	

Enter your skin thickness in mm.

Diabetes Test Result

The person is not diabetic

Conclusion & Future Scope

Conclusion

In the culmination of the MultiDiseasePrognosis project, a groundbreaking predictive healthcare system has been successfully developed. This innovative solution utilizes advanced machine learning algorithms to forecast diseases, including diabetes, heart diseases, and Parkinson's disease. The amalgamation of sophisticated algorithms, meticulous data analysis, and an intuitive user interface positions MultiDiseasePrognosis as a pioneer in the realm of predictive medicine.

The project's success is a testament to the collaborative efforts of the development team, insights from healthcare professionals, and iterative refinements based on user feedback. MultiDiseasePrognosis represents not only a technological achievement but a significant step toward leveraging data-driven insights for proactive and informed healthcare.

Future Scope

While MultiDiseasePrognosis has achieved notable milestones, recognizing its limitations opens avenues for future development and enhancement:

Limitations:

Data Diversity:
The effectiveness of disease prediction may be influenced by the diversity and representativeness of the training data. Enhancements are needed to ensure the model generalizes well across diverse populations.

Real-time Data Integration:
The system currently relies on historical data; incorporating real-time data from continuous monitoring devices could enhance the timeliness and accuracy of predictions.

Interpretability:

Improvements in explaining the rationale behind predictions are essential for gaining the trust of healthcare professionals and end-users.

Future Improvements:

Model Explainability:
Implement techniques to enhance the interpretability of machine learning models, providing transparent insights into the factors influencing predictions.

Continuous Learning Models:
Develop mechanisms for the models to adapt and learn from new data over time, ensuring that the system remains up-to-date with evolving healthcare trends.

User Feedback Mechanism:
Incorporate a robust feedback loop mechanism allowing users to provide feedback on predictions, contributing to ongoing model refinement.

P**rivacy Measures:**
Strengthen privacy measures to ensure compliance with healthcare regulations and address concerns related to the handling of sensitive health data.

Collaboration with Healthcare Providers:
Forge deeper collaborations with healthcare institutions to integrate clinical insights and ensure the project aligns with real-world healthcare practices.

Enhanced User Interface:
Continuously improve the user interface to enhance user experience, making it more intuitive and accessible to a wider audience.

References/Bibliography

https://scikit-learn.org/stable/
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

https://pandas.pydata.org/docs/

https://www.kaggle.com/datasets/mathchi/diabetes-data-set

https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

https://www.kaggle.com/datasets/vikasukani/parkinsons-disease-data-set

https://docs.streamlit.io/