MULTIPLE DISEASE PREDICTION SYSTEM

A PROJECT
SUBMITTED TO THE

THAKUR INSTITUTE OF MANAGEMENT STUDIES, CAREER DEVELOPMENT AND RESEARCH

FOR THE DEGREE OF

MASTERS IN COMPUTER APPLICATION

BY

Mr. Shubham Maurya & Prince Mishra
(Seat No.: 54 & 58)

UNDER THE GUIDANCE OF

Dr. Aprajita Singh

DEPARTMENT OF MCA

THAKUR INSTITUTE OF MANAGEMENT STUDIES, CAREER DEVELOPMENT AND RESEARCH

THAKUR EDUCATIONAL CAMPUS, SHYAMNARAYAN THAKUR MARG, THAKUR VILLAGE, KANDIVALI EAST, MUMBAI, MAHARASHTRA 400101

Year 2023

Project Certificate for MCAStudents

This is to certify that the project is entitled "MULTIPLE DISEASE PREDICTION SYSTEM".Undertaken at the "Thakur Institute of Management Studies Career

Development & Research" by

Mr. Shubham Maurya Roll No.54 and Prince Mishra Roll No.58 in partial fulfillment of MCAdegree(Semester II)

The examination had not been submitted for my other examination and does not form part of any other course that has been undergone by the candidate.

It is further certified that she has completed all required phases of the project.

Signature of External Examiner Signature of Internal Examiner

Signature of Project Guide Signature of Head of Department

College Seal

PREFACE

As part of our MCA curriculum and to gain practical knowledge we are required to make a Mini Project called "MULTIPLE DISEASE PREDICTION SYSTEM" The basic objection behind doing this project is to use our knowledge of programming to develop a working software system. In this project, we have created a web application that detect that the person is having the any disease of Diabetes, Heart or Parkinson's.

The person has to give their basic health data as a input so that the trained model can predict the person is having that disease or not. After the prediction it indicate the with outcome of the text written the person is having such disease or not.

ACKNOWLEDGEMENT

I have a great pleasure for representing this project report & titled "MULTIPLE DISEASE PREDICTION SYSTEM"& I grab this opportunity to convey my immense regards towards all the people who have the valuable contribution in the hour ofneed.

I take this opportunity to thank our director "Mrs. Vinita Gaikwad" of "Thakur Institute Management Studies of Carrier Development & Research. Kandivali(E) for giving me an opportunity to study in this institute & the most needed guidance throughout the course.

I also like to extend my gratitude to "Shirshendu Maitra" Head of thedepartment for their timely & prestigious guidance.

I would also like to thank "Aprajita Singh" who provided the guidance & necessary support during each phase of the project.

I also owe to my fellow friends who have been a constant source of help to solve the problem & also help during the project development process.

Prince and Shubham

Thakur Educational Trust's (Regd.)
THAKUR INSTITUTE OF MANAGEMENT STUDIES,
CAREER DEVELOPMENT & RESEARCH
(Approved by AICTE, Govt. of Maharashtra & Affiliated to University of Mumbai)
ISO 9001 : 2015 Certified • MCA Program Accredited by National Board of Accreditation, New Delhi

CERTIFICATE

This is to certify that the project entitled, MULTIPLE DISEASE PREDICTION SYSTEM work of Prince Mishra – 58 & Shubham Maurya – 54 submitted in partial fulfillment of the requirement for the award of degree of Masters of Computer Application [First Semester] from University of Mumbai.

_____ **PROF. Aprajita Singh**

Internal Examiner **Coordinator**

_____

External Examiner

Date:_____ **College Seal**

DECLARATION

I / We hereby declare that the project entitled, "**MULTIPLE DISEASE PREDICTION SYSTEM**" done at "**Thakur Institute of Management Studies Career Development & Research",** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirement for the award of degree of Masters of Computer Application to be submitted as First semester project as part of our curriculum.

SHUBHAM MAURYA AND PRINCE MISHRA

Table of Contents

Table of Figures

CHAPTER 01

INTRODUCTION

1.1 Problem definition :

The problem at hand is to develop a predictive model that can accurately predict the likelihood of multiple diseases in individuals based on their relevant health data and medical history. The objective is to create a reliable tool that can assist healthcare professionals in early detection and prevention of various diseases, leading to improved patient outcomes and reduced healthcare costs.

1) **Data Collection and Integration**: Gathering diverse and comprehensive health data from 'Kagagl' and different dataset websites. Integrating this data from different formats and systems can be complex and time-consuming.
2) **Feature Selection and Engineering**: Identifying the most relevant features from the collected data and transforming them into suitable representations for machine learning algorithms. This process requires domain knowledge and expertise in selecting informative and discriminative features. the model we have used is Support vector Machine (SVM) for Diabetes and Parkinson's and Logistic regression for the Heart disease.
3) **Algorithm Deployment**: Designing and developing accurate and robust machine learning algorithms capable of handling multi-class classification problems. The data is first standardized for convenience for the model to predict and then fitted in the model to transform the data in the standardized form.
4) **Model Training and Evaluation**: Training the predictive model on a representative dataset and evaluating its performance using appropriate metrics such as accuracy, precision. The data is first stratified or splinted in the 2 parts X and Y. X contains the whole column except the outcome column where the output of the data is indicated with 0 and 1. The model should be capable of generalizing well to unseen data and effectively detect multiple diseases.
5) **Interpretability and Explain ability**: Ensuring the model's interpretability and explain ability to gain trust and acceptance from healthcare professionals. The ability to provide explanations for predictions can help clinicians understand the factors influencing disease predictions and make informed decisions.
6) **Integration into Clinical Workflow**: Developing an intuitive user interface and integrating the predictive model seamlessly into the existing clinical workflow. The tool should be user-friendly, provide actionable insights, and facilitate the efficient utilization of the predictive model by healthcare professionals.

1.2 Objectives:

The objective of the MULTIPLE DISEASE PREDICTION SYSTEM projects is to develop a predictive model that can accurately predict the likelihood of multiple diseases in individuals based on their relevant health data and medical history. The project aims to achieve the following objectives:

1) **Early Detection and Prevention**: Enable healthcare professionals to identify individuals at risk of developing multiple diseases at an early stage. Early detection can facilitate timely interventions, preventive measures, and personalized treatment plans, ultimately leading to improved patient outcomes and reduced disease burden.

2) **Comprehensive Disease Assessment**: Provide a holistic view of an individual's health status by considering the prediction of multiple diseases simultaneously. This approach allows for a more comprehensive evaluation of an individual's health risks and enables healthcare professionals to address multiple health concerns simultaneously.

3) **Improved Clinical Decision Making**: Assist healthcare professionals in making informed clinical decisions by providing them with reliable predictions and risk assessments for multiple diseases. The predictive model can serve as a valuable decision support tool, aiding clinicians in prioritizing interventions, recommending appropriate screenings, and optimizing resource allocation.

4) **Enhanced Patient Care and Engagement**: Promote proactive healthcare management by empowering individuals to take control of their health. By providing individuals with personalized disease risk profiles, they can be educated about potential health risks, encouraged to adopt healthier lifestyles, and actively participate in disease prevention and management programs.

5) **Efficient Resource Allocation**: Enable efficient allocation of healthcare resources by identifying individuals who are at higher risk of developing multiple diseases. This information can aid in optimizing resource utilization, focusing preventive measures on high-risk individuals, and potentially reducing healthcare costs.

6) **Continual Improvement and Adaptability**: Establish a foundation for continual improvement and adaptation of the predictive model. By analyzing real-world outcomes and feedback, the model can be refined over time to enhance its accuracy, reliability, and usability.

1.3. Purpose and scope

1) Purpose

The purpose of a MULTIPLE DISEASE PREDICTION SYSTEM project using supervised data is to develop a predictive model that can accurately forecast the occurrence or probability of multiple diseases or health conditions based on a given set of input variables or features. The project aims to leverage historical data and patterns to create a reliable tool for early detection, prevention, and management of various diseases..

2) Applicability

The MULTIPLE DISEASE PREDICTION SYSTEM project using supervised data has broad applicability across various domains and can be beneficial in several contexts. Here are some areas where such projects can be applied:

Clinical Decision Support

Disease Prevention and Early Intervention

Public Health Planning

Personalized Medicine and Precision Healthcare

Clinical Trials and Research

Health Insurance and Actuarial Analysis

4. Scope

The scope of a MULTIPLE DISEASE PREDICTION SYSTEM project using supervised data encompasses various aspects related to data collection, preprocessing, model development, and evaluation. Here are some key components that define the scope of such a project:
Data Collection:
Data Preprocessing:
Feature Selection/Engineering:
Model Development:
Model Evaluation:
Model Deployment and Integration:
Ethical and Legal Considerations:

5. Achievements

A MULTIPLE DISEASE PREDICTION SYSTEM project using supervised data can achieve several significant outcomes and benefits. Here are some of the key achievements that can be realized through such a project:
Improved Disease Detection:
Early Intervention and Treatment:
Personalized Healthcare:
Optimal Resource Allocation:
Research and Insights:
Cost Reduction and Efficiency:
Empowering Individuals:

Enhanced Public Health Planning:

CHAPTER 2: SURVEY OF TECHNOLOGIES

The Summary of Technologies that are used in the development of project is given below:

PYTHON:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

The reason behind choosing Python for development of my project is that the Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions.

Google Colab:

Google Colab (short for Google Colaboratory) is a cloud-based integrated development environment (IDE) provided by Google. It allows users to write and execute Python code through their web browsers without requiring any local installations or setup. Google Colab is particularly popular among data scientists, machine learning practitioners, and researchers for its ability to run code on powerful virtual machines (VMs) provided by Google.

Here are some key features and characteristics of Google Colab:

Jupyter Notebook Integration: Google Colab is built on the Jupyter Notebook framework, which provides an interactive and collaborative environment for writing code. Notebooks consist of cells where you can write and execute code, view outputs, and add text explanations or visualizations.

Cloud-Based Computing: With Google Colab, your code is executed on remote servers in the cloud, leveraging Google's infrastructure. This eliminates the need for local hardware resources and allows you to take advantage of the high-performance computing capabilities provided by Google's servers.

Free to Use: Google Colab is available for free to users. It provides access to a limited amount of computing resources, including CPU and GPU instances. However, there is an

option to upgrade to Colab Pro, a subscription-based service, which offers additional benefits such as faster runtime, increased memory, and longer session durations.

GPU Support: Colab provides free access to GPU resources, which is particularly valuable for running computationally intensive tasks like deep learning. GPU acceleration can significantly speed up training models and processing large datasets.

Easy Data Sharing and Collaboration: Colab allows you to share your notebooks with others, making it easy to collaborate on code projects. You can share notebooks as view-only or give others permission to edit and run the code. It also supports real-time collaboration, where multiple users can work on a notebook simultaneously.

Integration with Other Google Services: Colab seamlessly integrates with other Google services like Google Drive and GitHub. You can easily import datasets from Drive, save your notebooks, or access code repositories hosted on GitHub.

Pre-installed Libraries and Packages: Colab comes with pre-installed popular libraries and packages commonly used in data science and machine learning, such as TensorFlow, Keras, PyTorch, NumPy, Pandas, and more. This makes it convenient to start working on projects without worrying about installing dependencies.

Notebooks with Markdown Support: Colab notebooks support markdown, allowing you to include formatted text, equations, images, and visualizations alongside your code. This makes it suitable for creating rich, explanatory documents that combine code, text, and visualizations.

Google Colab provides an accessible and powerful platform for writing, executing, and sharing code, making it a popular choice among data scientists, machine learning practitioners, and researchers for prototyping, experimentation, and collaborative coding.

Conda

Conda Conda is a popular package and environment management tool in the field of machine learning (ML) and data science. It is often used in conjunction with Python programming language to create and manage isolated environments with specific packages and dependencies required for ML projects.
Here are some key aspects of Conda for ML:

Package Management: Conda simplifies the installation and management of software packages, including ML libraries and frameworks. It provides a vast repository of pre-built packages, making it easy to install and update popular ML libraries such as TensorFlow, PyTorch, scikit-learn, and more. Conda ensures that the required packages and their dependencies are resolved and installed correctly, saving time and avoiding conflicts.

Environment Management: Conda enables the creation of isolated environments, also known as Conda environments, where you can manage different versions of packages and their dependencies separately. This allows you to have multiple environments tailored to specific ML projects or research needs. Each environment can have its own set of packages, Python version, and dependencies, ensuring reproducibility and avoiding conflicts between different projects.

Cross-platform Support: Conda is platform-agnostic and works on various operating systems, including Windows, macOS, and Linux. It provides a consistent environment management experience regardless of the underlying operating system, making it convenient for ML practitioners working on different platforms.

Easy Package Installation: Conda simplifies the installation process by automatically resolving package dependencies. It handles the installation of not only Python packages but also non-Python packages, which can be required for some ML libraries that have external dependencies, such as CUDA for GPU acceleration.

Conda-Forge: Conda-Forge is a community-driven repository that provides additional packages for Conda. It expands the range of available packages beyond the official Conda repository, making it a valuable resource for ML practitioners to find and install specialized packages and tools.

Reproducibility and Collaboration: Conda environments facilitate reproducibility by capturing and managing the specific package versions and dependencies required for a project. You can share the environment configuration file (usually named environment.yml) with collaborators or store it in a version control system, ensuring consistent and reproducible environments across different machines or team members.

Integration with Other Tools: Conda can be seamlessly integrated with other ML tools and workflows. It can be used in conjunction with popular ML development environments like Jupyter Notebook or integrated into continuous integration (CI) and deployment pipelines to ensure consistent and reproducible ML workflows.

Overall, Conda provides a robust package and environment management solution for ML practitioners. It simplifies the process of installing and managing ML libraries, enables reproducibility, and allows for efficient collaboration on ML projects by providing isolated and customizable environments.

Streamlit:

Streamlit is a powerful Python library that is gaining popularity in the field of machine learning (ML) for building interactive and customizable web applications. It serves as a valuable tool for ML practitioners in the following ways:

Rapid Prototyping: Streamlit allows ML practitioners to quickly create interactive prototypes and demos of their ML models or data analysis projects. It provides a simple and intuitive way to convert Python scripts into interactive web applications, eliminating the need for web development expertise. This accelerates the development cycle and facilitates faster iteration and experimentation.

Easy Deployment: Streamlit simplifies the deployment of ML models and applications. With just a few lines of code, ML practitioners can convert their models or analysis scripts into web apps that can be easily deployed on local machines, cloud platforms, or shared with others. This ease of deployment enables seamless sharing and collaboration with stakeholders, clients, or team members.

Customizable User Interfaces: Streamlit offers a wide range of interactive widgets and components that can be used to build user-friendly interfaces for ML applications. ML practitioners can add sliders, dropdown menus, checkboxes, and other interactive elements to enable users to explore and interact with their models or visualizations. This enhances user experience and makes ML applications more accessible to non-technical users.

Data Visualization: Streamlit provides various plotting and visualization capabilities that enable ML practitioners to create interactive and dynamic visualizations of their data. It supports popular visualization libraries such as Matplotlib, Plotly, and Seaborn, allowing practitioners to present insights and results in a visually appealing and interactive manner. These visualizations can help stakeholders understand complex ML models or analyze data effectively.

Experiment Tracking and Documentation: Streamlit allows ML practitioners to document their ML experiments, results, and insights directly within the web application. This feature enables them to capture and communicate their thought process, methodologies, and findings alongside the interactive components. It serves as a valuable tool for reproducibility, knowledge sharing, and collaboration within the ML community.

Integration with ML Libraries: Streamlit seamlessly integrates with popular ML libraries and frameworks such as TensorFlow, PyTorch, and scikit-learn. ML practitioners can combine the capabilities of these libraries with the interactivity and user interface features of Streamlit to build sophisticated ML applications. They can showcase model predictions, visualize model outputs, or even create interactive dashboards for monitoring and analyzing ML models in real-time.

Community and Ecosystem: Streamlit has a vibrant and growing community of users and contributors. The Streamlit community actively shares examples, best practices, and open-source components, making it easier for ML practitioners to learn and leverage the capabilities of the library. This active ecosystem ensures continuous improvement, support, and availability of resources for ML practitioners using Streamlit.
In summary, Streamlit is an important tool for ML practitioners as it simplifies the process of building interactive web applications, enhances deployment capabilities, enables customization of user interfaces, facilitates data visualization, and supports documentation and experiment tracking. It empowers ML practitioners to create compelling and accessible applications for showcasing, sharing, and collaborating on their ML models and data analysis projects.

Spyder:

Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software.[4][5] It is released under the MIT license.[6]
Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first-party and third-party plugins,[7] includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint[8] and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.[9][10]

Spyder uses Qt for its GUI and is designed to use either of the PyQt or PySide Python bindings.[11] QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

CHAPTER 3: REQUIREMENTS AND ANALYSIS

1. Problem Defination

The problem definition for a MULTIPLE DISEASE PREDICTION SYSTEM project involves identifying and predicting the occurrence of multiple diseases based on relevant features and historical data. The goal is to develop a predictive model that can accurately classify individuals into different disease categories, enabling early detection and proactive interventions.

2. Requirement Specification

Function of software:

1) **Google colab**: For the collaboration of the code and training and testing of the Model for prediction and then after testing by using the pickle library we loaded and saved our model in file form.
2) **Conda:** Conda we used it getting suitable environment for our model to run on the web page.
3) **Streamlit:** It is the machine learning framework.We used streamlit because for the development of the webpage for the Machine learning it is very easy to develop. It just need to import required library and we can use the features of streamlit. And for the develop of web page we just used the python code for taking and acting for the input.
4) **Spyder:** It was very easy to frame a web page in spyder it easy to import the library in spyder ide and we just used the python for it also.

3.3 Planning Scheduling

- Requirements- March
- Design – March and April
- Coding –April and MAy
- Testing – April and May
- Deployment – May
- Documentation – May

4. Software and Hardware Requirements

- Software Requirements: -

Operating System - Windows

Web-Technology - Streamlit frame work in Spyder

Back-End - Python in Google Colab

Environment - Conda

Libraries - Numpy, Pandas, Matplotlib, Sci-kit learn, Pickle, Streamlit

- Hardware Requirements: -

Pentium-IV (Processor)

256 MB RAM

512 KB Cache Memory

Hard Disk 10 GB

Microsoft Compatible 101 or More Key Board

3.4 Hardware requirements:

- Keyboard
- Any mouse
- Bandwidth with 100kbps or above

3.5 Software requirement:

- Programming language: Python and libraries of python required for the ML
- Database: Dataset

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

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

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

3.6 Justification of selection of technology

Streamlit:

Streamlit is an open-source Python library and app framework which helps to create a custom frontend for Machine Learning and Data Science projects. To create such an application at the core we will use an ML model which makes predictions based on input features located on the left side of the page.

Streamlit's Features

- Free and open source.
- Build apps in a dozen lines of Python with a simple API.
- No callbacks.
- No hidden state.
- Works with TensorFlow, Keras, PyTorch, Pandas, Numpy, Matplotlib, Seaborn, Altair, Plotly, Bokeh, Vega-Lite, and more.

CONDA:

Benefits of Using Python Anaconda

- It is free and open-source
- It has more than 1500 Python/*R data science packages*
- Anaconda simplifies package management and deployment
- It has tools to easily collect data from sources using machine learning and AI
- It creates an environment that is easily manageable for deploying any project
- Anaconda is the industry standard for developing, testing and training on a single machine
- It has good community support- you can ask your questions there.

Gantt Chart:

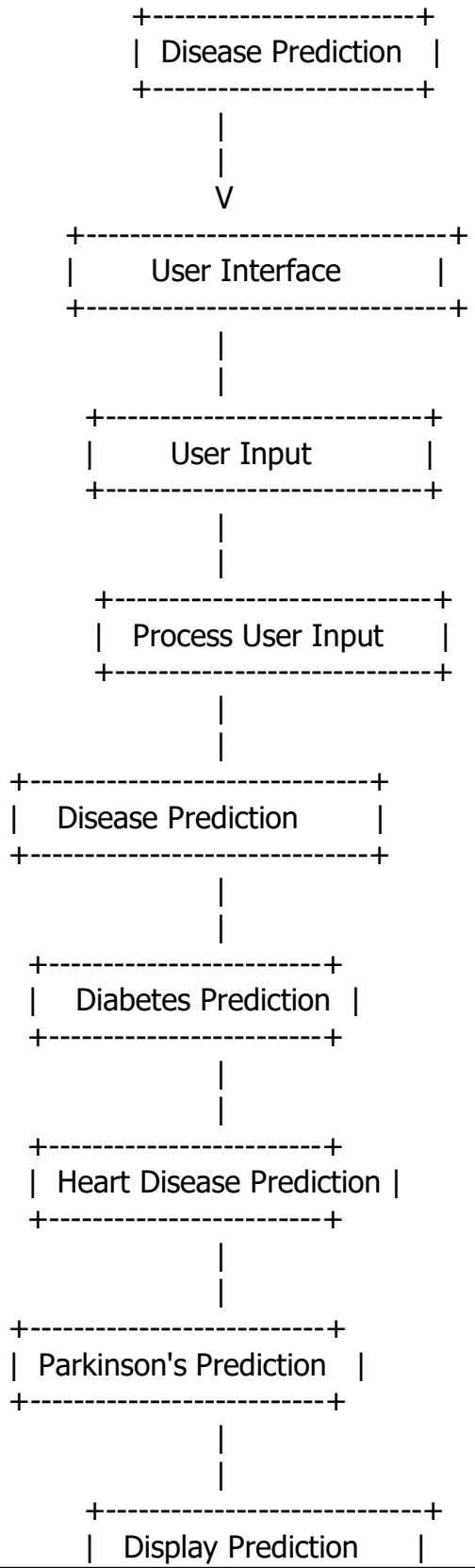
Task	Start Date	End Date	Duration
Project Initiation	15-Mar-23	20-Mar-23	20-Mar-23
Data Collection	21-Mar-23	25-Mar-23	25-Mar-23
Data Preprocessing	26-Mar-23	05-Apr-23	05-Apr-23
Feature Engineering	06-Apr-23	12-Apr	12-Apr
Model Development	13-Apr-23	20-Apr-23	20-Apr-23
Model Evaluation	21-Apr-23	30-Apr-23	30-Apr-23
UI Development	01-May-23	15-May-23	15-May-23
Integration Testing	16-May-23	20-May-23	20-May-23
Deployment	21-May-23	25-May-23	25-May-23
Documentation	26-May-23	29-May-23	29-May-23

CHAPTER 04

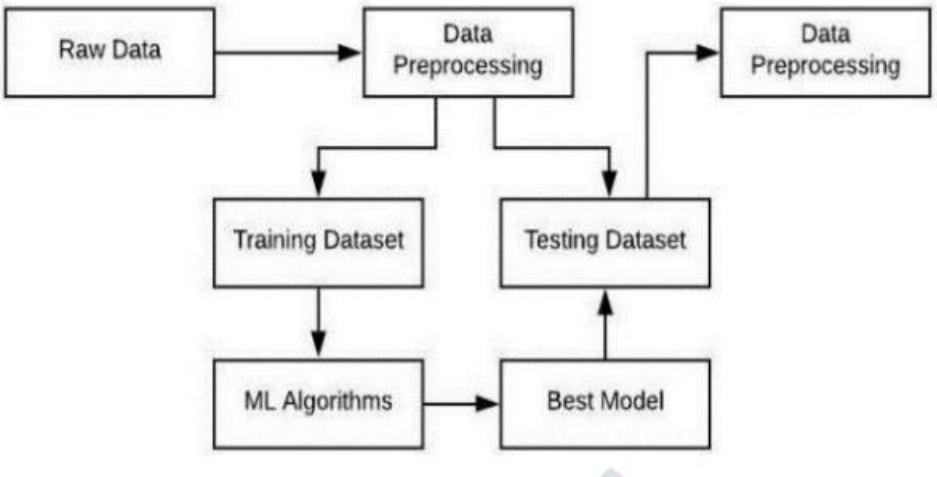
SYSTEM DESIGN

4.1 Basic Module

Use Case Diagram

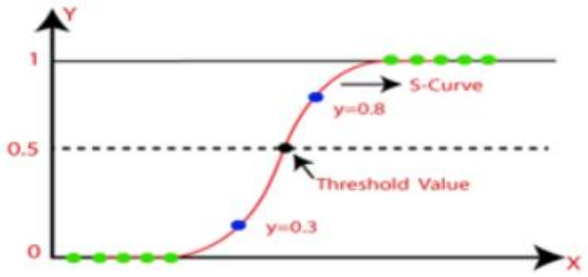


4.2 Block Diagram



4.3 Mathematical Models used

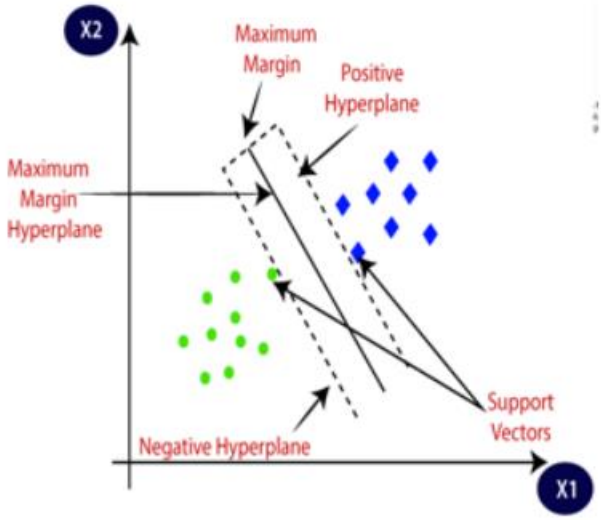
1) Logistic Regression



Mathematical steps to get the Logistic regression Equation:

$$\log\left[\frac{y}{1-y}\right] = b0 + b1x1 + b2x2 + \quad \ldots \quad + bnxn$$

2) SVM(Support Vector Machine)



ADVANTAGES AND DISADVANTAGES OF SYSTEM

A. Advantages

1. It is very useful in Hospitals.
2. Reduce the workload of Hospital staff.
3. Easy to use. Anyone can use it.
4. It predicts robust and accurate results.
5. It takes less time to identify the patient have a particular disease or not.
6. Needy patient can get early treatment by identifying early-stage disease.
7. Cost saving.

B. Disadvantages

1. There is little bit number of chances of miss prediction in some cases.

Conclusion

We successfully build a system that predict more than one disease with high accuracy.

It is easy to use as well. We achieve accuracy on project as follows:

1) Our project identifies heart disease with an accuracy of 88%.

2) Our project predicts diabetes disease with an accuracy of 78%.

3) Our project predicts Parkinsin’s disease with an accuracy of 89%.

FUTURE SCOPE

1. It will be very useful to healthcare industries like hospitals to identify diseases at early stage.
2. In the future we can add more diseases in the existing system.
3. We can try to improve the accuracy of prediction in order to decrease the mortality rate.
4. Try to make the system more user-friendly.

CHAPTER 05 IMPLEMENTATION AND TESTING

1. Implementation Approaches

The choice of implementation approach depends on factors such as the availability of data, the complexity of diseases, the interpretability of results, and the expertise of the development team. It's important to carefully evaluate the pros and cons of each approach and select the most suitable one for your specific MULTIPLE DISEASE PREDICTION SYSTEM project.

2. Coding Details and Code Efficiency

1. Code Details

5.2 Modification and Improvements

Colab Links for the Model training and testing

1) **Diabetes**
   Colab link for Diabetes Prediction
   https://colab.research.google.com/drive/1NQ_pgO4wY_qcFz8UxtBOffSKIehOYSSm?usp=sharing

2) **Heart Disease Prediction**
   https://colab.research.google.com/drive/1a57mU6FDDLllFZ4wP45W6popQBJn0Vj4?usp=sharing

3) **Parkinson's Disease Prediction**
   https://colab.research.google.com/drive/1UTF7RCrDMNS8LpBx02-tELr5RfXwvItT?usp=sharing

Code of the Web Page on Spyder IDE using streamlit framework

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun  6 00:04:31 2023

@author: Shubham Ramjanak Maurya
"""

import pickle
import streamlit as st
from streamlit_option_menu import option_menu

# loading the saved models

diabetes_model = pickle.load(open('E:/My Projects/MULTIPLE DISEASE PREDICTION SYSTEM System/Saved models/diabetes_trained_model.sav','rb'))

heart_disease_model = pickle.load(open('E:/My Projects/MULTIPLE DISEASE PREDICTION SYSTEM System/Saved models/heart_trained_model.sav','rb'))

parkinsons_model = pickle.load(open('E:/My Projects/MULTIPLE DISEASE PREDICTION SYSTEM System/Saved
```

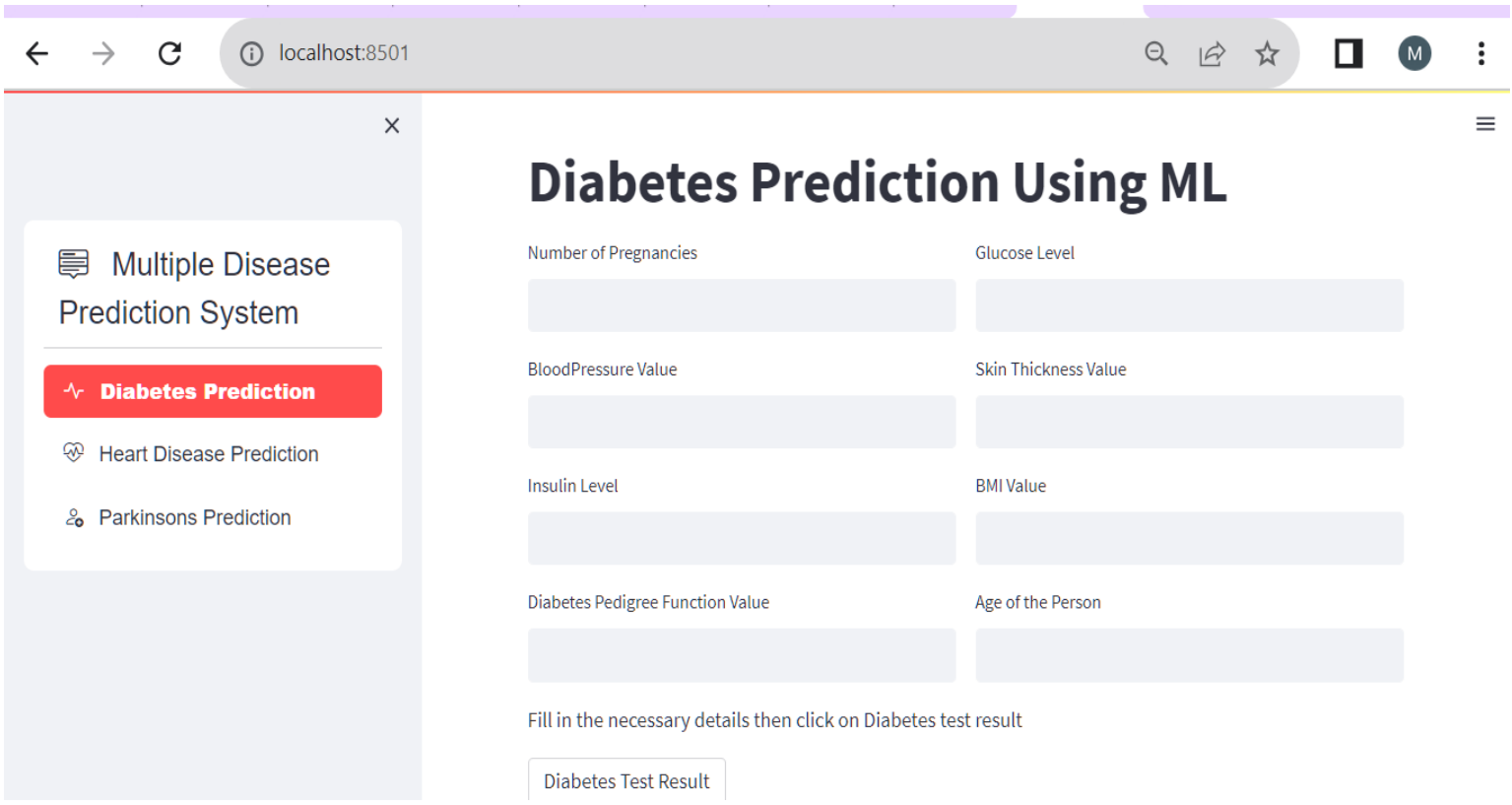
```
models/parkisons_trained_model.sav','rb'))

# create the side bar for navigation

with st.sidebar:

    selected = option_menu('Multiple Disease      Prediction System',

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

                           icons = ['activity','heart-pulse','person-add'],
                           default_index = 0)



# 1) Diabetes Predicition Page
if (selected == 'Diabetes Prediction'):

    st.markdown(
        """
        <style>
        body {
            background-image: url('E:/My Projects/MULTIPLE DISEASE PREDICTION SYSTEM System/img1.jpg');
            background-size: cover;
        }
        </style>
        """,
        unsafe_allow_html=True
    )

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



    # getting the input data from user
    # columns forinput fields
    col1, col2, = st.columns(2)

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

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

    with col1:
        BloodPressure = st.text_input('BloodPressure Value')

    with col2:
         SkinThickness = st.text_input('Skin Thickness Value')

    with col1:
         Insulin = st.text_input('Insulin Level')

    with col2:
           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')
```

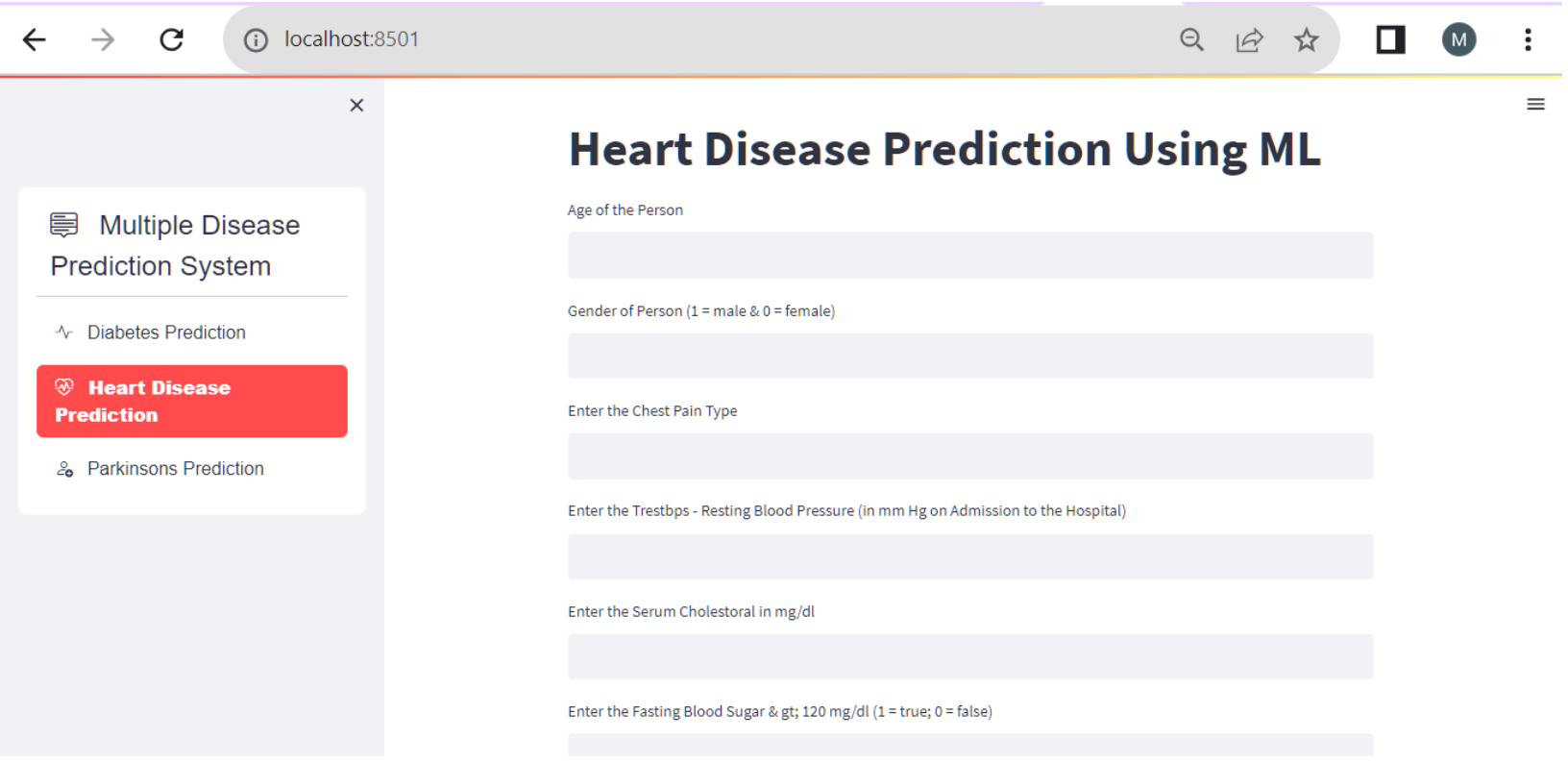
```
    # Display a message using st.markdown()
    st.markdown("Fill in the necessary details then click on Diabetes test result")


#Code for Prediction
diab_diagnosis = ''


# Creating a button for prediction

if st.button('Diabetes Test Result'):
        # Convert input variables to numeric types
        Pregnancies = int(Pregnancies)
        Glucose = float(Glucose)
        BloodPressure = float(BloodPressure)
        SkinThickness = float(SkinThickness)
        Insulin = float(Insulin)
        BMI = float(BMI)
        DiabetesPedigreeFunction = float(DiabetesPedigreeFunction)
        Age = int(Age)

        # Perform prediction
        diab_prediction = diabetes_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI,
DiabetesPedigreeFunction, Age]])
        if diab_prediction[0] == 1:
            diab_diagnosis = 'The Person is Diabetic'
            st.error(diab_diagnosis)
        else:
            diab_diagnosis = 'The Person is Not Diabetic'
            st.success(diab_diagnosis)



# 2) Heart Disease Predicition Page
if (selected == 'Heart Disease Prediction'):

    # page title
    st.title('Heart Disease Prediction Using ML')

    # getting the input data from user
    # columns forinput fields

    age = st.text_input('Age of the Person')
    sex = st.text_input('Gender of Person (1 = male & 0 = female)')
    cp = st.text_input('Enter the Chest Pain Type')
    trestbps = st.text_input('Enter the Trestbps - Resting Blood Pressure (in mm Hg on Admission to the Hospital)')
    chol = st.text_input('Enter the Serum Cholestoral in mg/dl')
    fbs = st.text_input('Enter the Fasting Blood Sugar & gt; 120 mg/dl (1 = true; 0 = false)')
    restecg = st.text_input('Enter the Resting Electrocardiographic Results')
    thalach = st.text_input('Enter the Maximum Heart Rate Achieved')
    exang = st.text_input('Enter the Eexercise Induced Angina (1 = yes and 0 = no)')
    oldpeak = st.text_input('Enter the ST Depression Induced by Exercise Relative to Rest)')
    slope = st.text_input('Enter The Slope of the Peak Exercise ST Segment')
    ca = st.text_input('Enter the Number of Major Vessels (0-3) Colored by Flourosopy')
    thal = st.text_input('Enter the Thalassemia: 0 = Normal; 1 = Fixed defect; 2 = Reversable defect')

    # Display a message using st.markdown()
    st.markdown("Fill in the necessary details and click on Heart Disease test result")


#Code for Prediction
heart_diagnosis = ''

# Creating a button for prediction

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

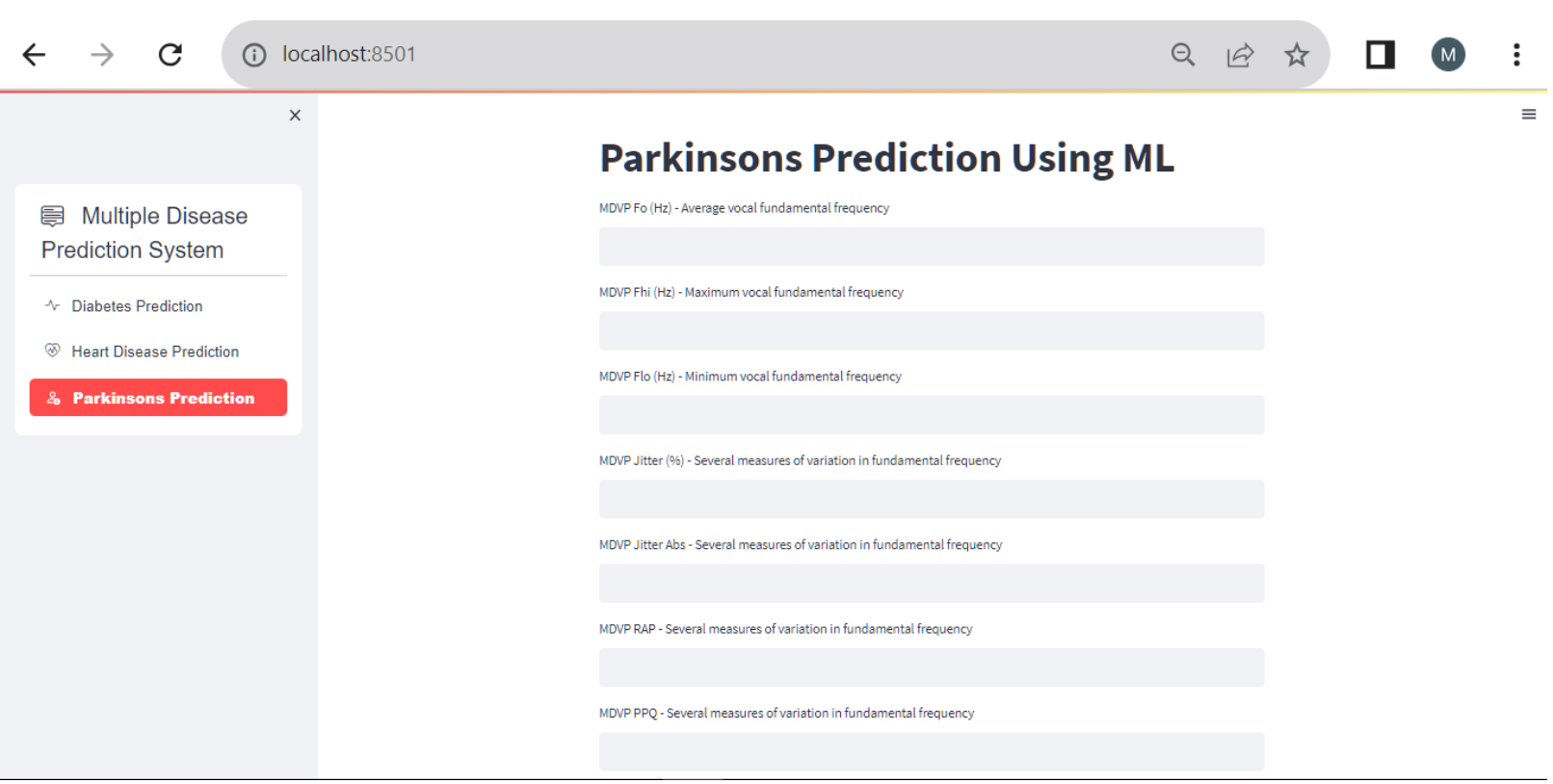
```
        # Convert input variables to numeric types
        age = int(age)
        sex = int(sex)
        cp = int(cp)
        trestbps = float(trestbps)
        chol = float(chol)
        fbs = int(fbs)
        restecg = int(restecg)
        thalach = float(thalach)
        exang = int(exang)
        oldpeak = float(oldpeak)
        slope = int(slope)
        ca = int(ca)
        thal = int(thal)

        # Perform prediction
        heart_prediction = heart_disease_model.predict([[age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang,
oldpeak, slope, ca, thal]])
        if heart_prediction[0] == 0:
            heart_diagnosis = 'The Person does not have Heart Disease'
            st.success(heart_diagnosis)
        else:
            heart_diagnosis = 'The Person has Heart Disease'
            st.error(heart_diagnosis)



# 3) Parkinsons Predicition Page
if (selected == 'Parkinsons Prediction'):

    # page title
    st.title('Parkinsons Prediction Using ML')

    # getting the input data from user

    MDVP_Fo = st.text_input('MDVP Fo (Hz) - Average vocal fundamental frequency')
    MDVP_Fhi = st.text_input('MDVP Fhi (Hz) - Maximum vocal fundamental frequency')
    MDVP_Flo = st.text_input('MDVP Flo (Hz) - Minimum vocal fundamental frequency')
    MDVP_Jitter_percent = st.text_input('MDVP Jitter (%) - Several measures of variation in fundamental frequency')
    MDVP_Jitter_Abs = st.text_input('MDVP Jitter Abs - Several measures of variation in fundamental frequency')
    MDVP_RAP = st.text_input('MDVP RAP - Several measures of variation in fundamental frequency')
    MDVP_PPQ = st.text_input('MDVP PPQ - Several measures of variation in fundamental frequency')
    Jitter_DDP = st.text_input('Jitter DDP - Several measures of variation in fundamental frequency')
    MDVP_Shimmer = st.text_input('MDVP Shimmer - Several measures of variation in fundamental frequency')
    MDVP_Shimmer_dB = st.text_input('MDVP Shimmer dB - ')
    Shimmer_APQ3 = st.text_input('Shimmer APQ3 - ')
    Shimmer_APQ5 = st.text_input('Shimmer APQ5 - ')
    MDVP_APQ = st.text_input('MDVP APQ - ')
    Shimmer_DDA = st.text_input('Shimmer DDA - ')
    NHR = st.text_input('NHR - Measures of ratio of noise to tonal components in the voice')
    HNR = st.text_input('HNR - Measures of ratio of noise to tonal components in the voice')
    RPDE = st.text_input('RPDE - Nonlinear dynamical complexity measures')
    DFA = st.text_input('DFA - Signal fractal scaling exponent')
    spread1 = st.text_input('Spread1 - Nonlinear measures of fundamental frequency variation')
    spread2 = st.text_input('Spread2 - Nonlinear measures of fundamental frequency variation')
    D2 = st.text_input('D2 - Nonlinear dynamical complexity measures')
    PPE = st.text_input('PPE - Nonlinear measures of fundamental frequency variation')

    # Display a message using st.markdown()
    st.markdown("Fill in the necessary details and click on Parkinsons test result")


#Code for Prediction
Parkinsons_diagnosis = ''
```

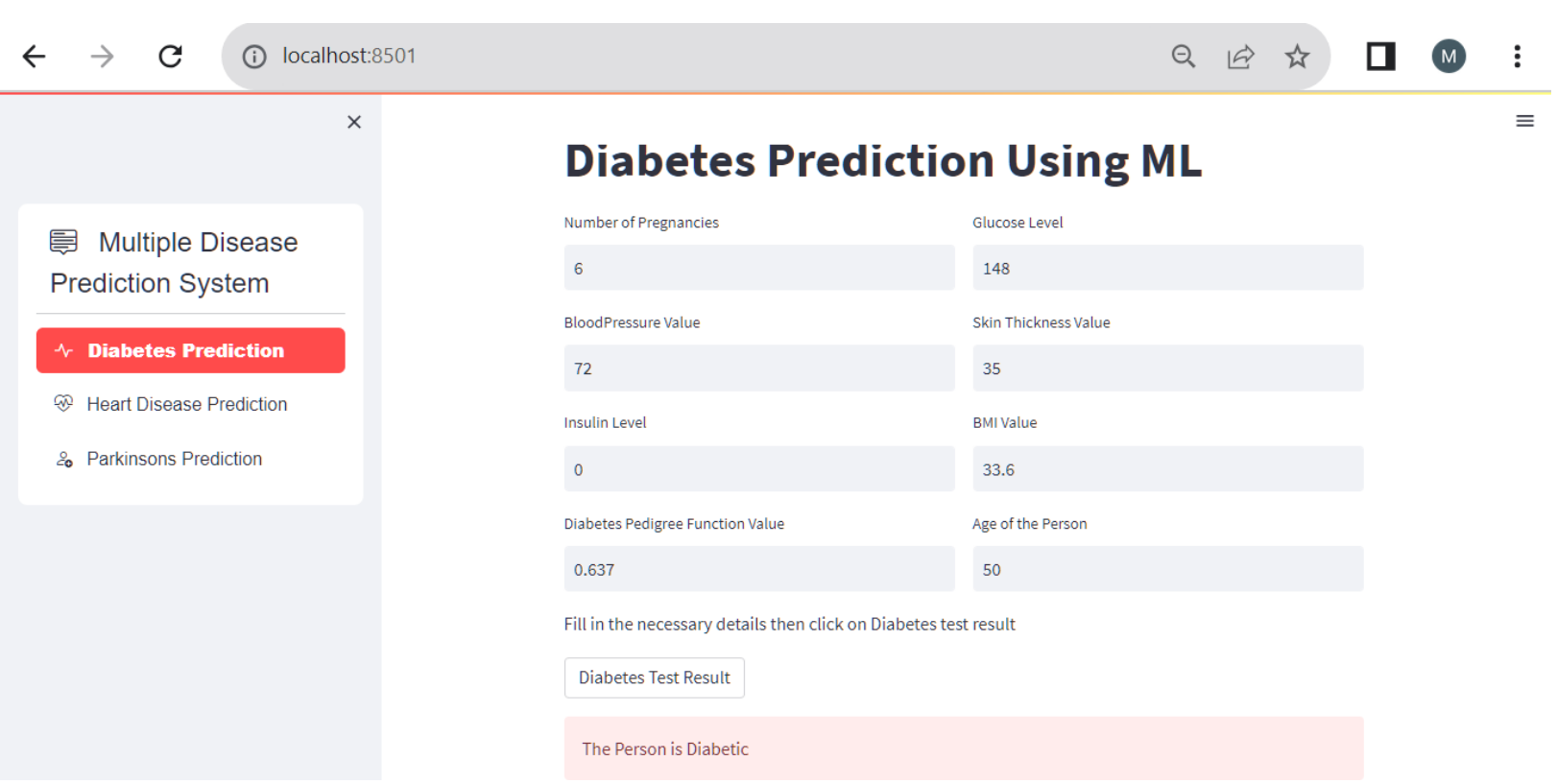
```
# Creating a button for prediction
if st.button('Parkinsons Test Result'):
        # Convert input variables to appropriate types
        MDVP_Fo = float(MDVP_Fo)
        MDVP_Fhi = float(MDVP_Fhi)
        MDVP_Flo = float(MDVP_Flo)
        MDVP_Jitter_percent = float(MDVP_Jitter_percent)
        MDVP_Jitter_Abs = float(MDVP_Jitter_Abs)
        MDVP_RAP = float(MDVP_RAP)
        MDVP_PPQ = float(MDVP_PPQ)
        Jitter_DDP = float(Jitter_DDP)
        MDVP_Shimmer = float(MDVP_Shimmer)
        MDVP_Shimmer_dB = float(MDVP_Shimmer_dB)
        Shimmer_APQ3 = float(Shimmer_APQ3)
        Shimmer_APQ5 = float(Shimmer_APQ5)
        MDVP_APQ = float(MDVP_APQ)
        Shimmer_DDA = float(Shimmer_DDA)
        NHR = float(NHR)
        HNR = float(HNR)
        RPDE = float(RPDE)
        DFA = float(DFA)
        spread1 = float(spread1)
        spread2 = float(spread2)
        D2 = float(D2)
        PPE = float(PPE)


        # Perform prediction
        parkinsons_prediction = parkinsons_model.predict([[MDVP_Fo, MDVP_Fhi, MDVP_Flo, MDVP_Jitter_percent,
MDVP_Jitter_Abs,
                                                MDVP_RAP, MDVP_PPQ, Jitter_DDP, MDVP_Shimmer,
                                                MDVP_Shimmer_dB, Shimmer_APQ3, Shimmer_APQ5,
                                                MDVP_APQ, Shimmer_DDA, NHR, HNR, RPDE, DFA, spread1,
                                                spread2, D2, PPE]])
        if parkinsons_prediction[0] == 0:
           Parkinsons_diagnosis = "The Person does not have Parkinsons Disease"
           st.success(Parkinsons_diagnosis)
        else:
           Parkinsons_diagnosis = "The Person has Parkinsons"
           st.error(Parkinsons_diagnosis)




# Display a message at the end using st.markdown()
st.markdown('Shubham Maurya(54) and Prince Mishra(58)')
```

5.2.1 Unit Testing

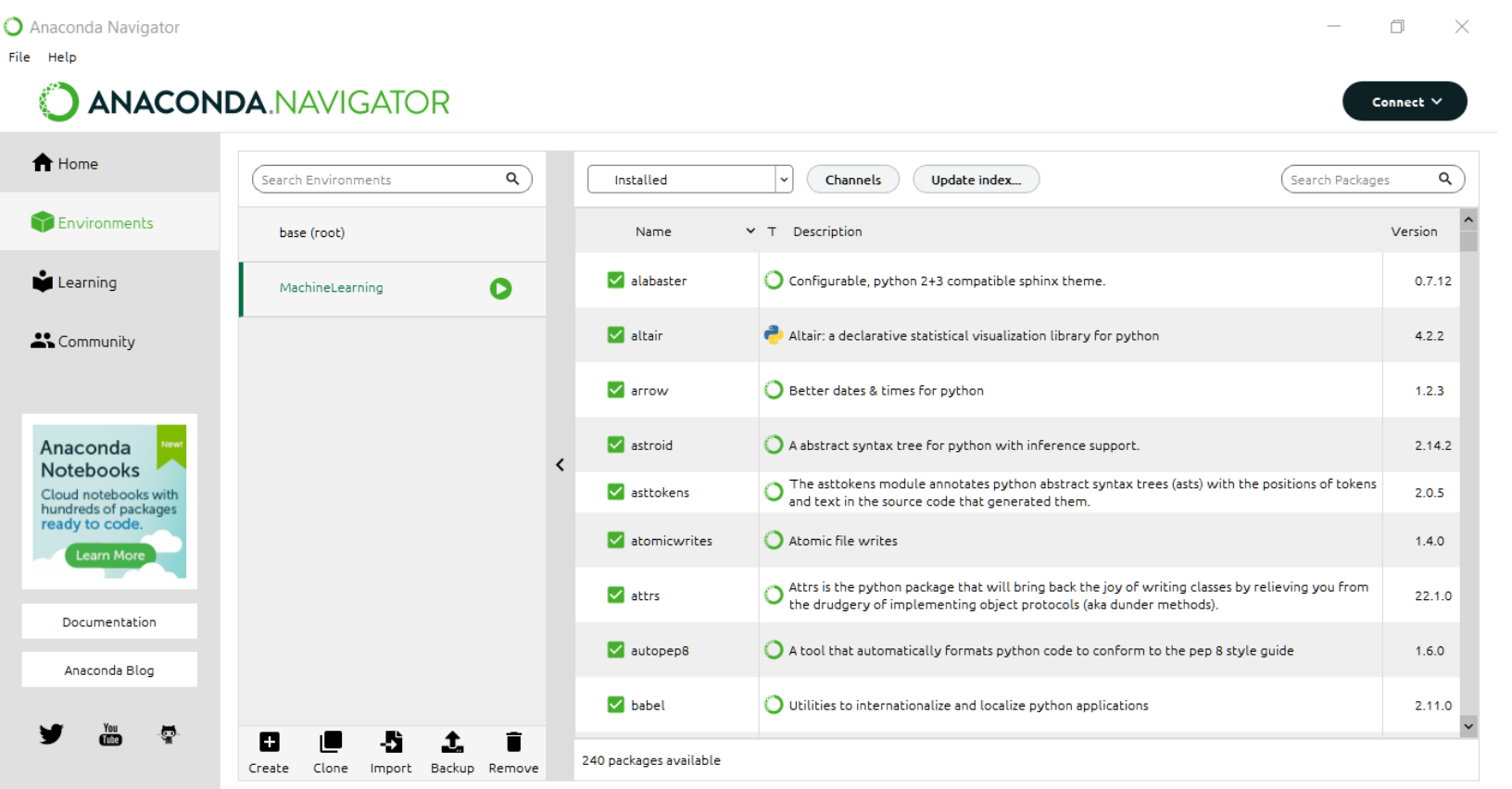


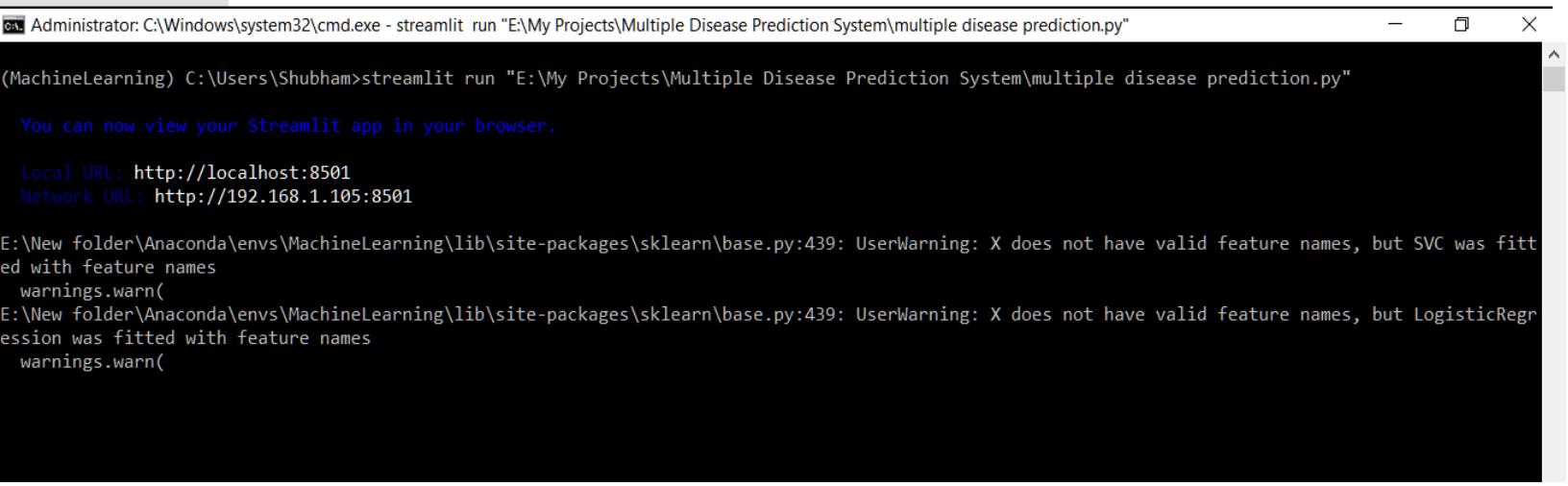



localhost:8501
Multiple Disease Prediction System
Diabetes Prediction
Heart Disease Prediction
Parkinsons Prediction
Parkinsons Prediction Using ML
MDVP Fo (Hz) - Average vocal fundamental frequency
MDVP Fhi (Hz) - Maximum vocal fundamental frequency
MDVP Flo (Hz) - Minimum vocal fundamental frequency
MDVP Jitter (%) - Several measures of variation in fundamental frequency
MDVP Jitter Abs - Several measures of variation in fundamental frequency
MDVP RAP - Several measures of variation in fundamental frequency
MDVP PPQ - Several measures of variation in fundamental frequency


localhost:8501
Multiple Disease Prediction System
Diabetes Prediction
Heart Disease Prediction
Parkinsons Prediction
Diabetes Prediction Using ML
Number of Pregnancies
6
Glucose Level
148
BloodPressure Value
72
Skin Thickness Value
35
Insulin Level
0
BMI Value
33.6
Diabetes Pedigree Function Value
0.637
Age of the Person
50
Fill in the necessary details then click on Diabetes test result
Diabetes Test Result
The Person is Diabetic


localhost:8501
Multiple Disease Prediction System
Diabetes Prediction
Heart Disease Prediction
Parkinsons Prediction
Enter the ST Depression Induced by Exercise Relative to Rest)
24
Enter The Slope of the Peak Exercise ST Segment
1
Enter the Number of Major Vessels (0-3) Colored by Flourosopy
2
Enter the Thalassemia: 0 = Normal; 1 = Fixed defect; 2 = Reversable defect
3
Fill in the necessary details and click on Heart Disease test result
Heart Disease Test Result
The Person does not have Heart Disease


Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Connect
Home
Environments
Learning
Community
Anaconda Notebooks
Cloud notebooks with hundreds of packages ready to code.
Learn More
Documentation
Anaconda Blog
Search Environments
base (root)
MachineLearning
Create
Clone
Import
Backup
Remove
Installed
Channels
Update index...
Search Packages
Name
T
Description
Version
alabaster
Configurable, python 2+3 compatible sphinx theme.
0.7.12
altair
Altair: a declarative statistical visualization library for python
4.2.2
arrow
Better dates & times for python
1.2.3
astroid
A abstract syntax tree for python with inference support.
2.14.2
asttokens
The asttokens module annotates python abstract syntax trees (asts) with the positions of tokens and text in the source code that generated them.
2.0.5
atomicwrites
Atomic file writes
1.4.0
attrs
Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).
22.1.0
autopep8
A tool that automatically formats python code to conform to the pep 8 style guide
1.6.0
babel
Utilities to internationalize and localize python applications
2.11.0
240 packages available


Administrator: C:\Windows\system32\cmd.exe - streamlit run "E:\My Projects\Multiple Disease Prediction System\multiple disease prediction.py"
(MachineLearning) C:\Users\Shubham>streamlit run "E:\My Projects\Multiple Disease Prediction System\multiple disease prediction.py"
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.1.105:8501
E:\New folder\Anaconda\envs\MachineLearning\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
warnings.warn(
E:\New folder\Anaconda\envs\MachineLearning\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
warnings.warn(

CHAPTER 07 CONCLUSION AND FUTURE WORK

In conclusion, a MULTIPLE DISEASE PREDICTION SYSTEM project aims to develop a system that can accurately predict the occurrence or likelihood of multiple diseases based on input data such as symptoms, medical history, or diagnostic tests. The project involves implementing various components, including user interface, data preprocessing, disease prediction models, and result visualization.

By leveraging different implementation approaches such as rule-based systems, machine learning algorithms, expert systems, hybrid approaches, ensemble methods, or deep learning techniques, the project can achieve accurate and reliable disease predictions. The choice of implementation approach depends on factors such as the availability of data, the complexity of diseases, interpretability requirements, and the expertise of the development team.

The successful implementation of a MULTIPLE DISEASE PREDICTION SYSTEM project can have significant benefits in healthcare. It can assist healthcare professionals in early detection, diagnosis, and treatment planning, leading to improved patient outcomes, reduced healthcare costs, and better resource allocation.

However, it's important to note that disease prediction models are not infallible and should be used as decision support tools rather than replacements for medical expertise. The predictions provided by the system should always be interpreted and validated by qualified healthcare professionals.

Moreover, ethical considerations, data privacy, and security measures should be taken into account throughout the project implementation to ensure patient confidentiality and comply with relevant regulations.

In summary, a MULTIPLE DISEASE PREDICTION SYSTEM project has the potential to make a positive impact on healthcare by assisting in early disease detection and management. It requires careful consideration of implementation approaches, validation of results, and adherence to ethical and privacy considerations to ensure its effectiveness and trustworthiness.

Future Scope:

The future scope of a MULTIPLE DISEASE PREDICTION SYSTEM system project is promising, with several potential areas of development and improvement. Here are some key aspects of future scope for such projects:

Enhanced Accuracy: Continued advancements in machine learning algorithms, deep learning architectures, and data collection techniques can contribute to improving the accuracy of disease predictions. More comprehensive and diverse datasets, including genetic data, medical imaging, wearable sensor data, and electronic health records, can be leveraged to train and refine prediction models.

Integration with Real-Time Data: Integrating real-time data sources, such as wearable devices or IoT-enabled

healthcare devices, can provide continuous monitoring of patients' health parameters. This real-time data can be incorporated into disease prediction models, allowing for timely and proactive interventions and personalized treatment plans.

Personalized Medicine: The future of disease prediction systems lies in personalized medicine, where predictions are tailored to individual patients based on their specific characteristics, genetic profiles, and environmental factors. By incorporating personalized data and considering patient-specific factors, disease prediction models can provide more accurate and personalized risk assessments and treatment recommendations.

Multimodal Data Analysis: Integrating and analyzing multiple types of data, such as genetic, clinical, lifestyle, and social determinants of health, can provide a more holistic view of disease risk factors and prediction. Combining data from different sources can help identify complex relationships and uncover novel risk factors for diseases.

Explainable AI and Interpretability: As disease prediction models become more complex, ensuring interpretability and explainability of the predictions becomes crucial. Future developments should focus on techniques that provide transparency and explanations for the predictions, enabling healthcare professionals to understand and trust the system's decisions.

Decision Support Systems: Disease prediction systems can evolve into comprehensive decision support systems that not only predict diseases but also assist healthcare professionals in treatment planning, medication recommendations, and intervention strategies. Integration with electronic health records and clinical decision support systems can further enhance the value and impact of these systems.

Early Disease Detection and Prevention: By focusing on early disease detection and prevention, disease prediction systems can play a vital role in reducing the burden of chronic diseases. The integration of predictive models into population health management strategies can help identify high-risk individuals, implement preventive measures, and allocate healthcare resources more efficiently.

Telemedicine and Remote Monitoring: With the growing adoption of telemedicine and remote patient monitoring, disease prediction systems can be integrated into these platforms to provide remote risk assessments and enable proactive interventions, especially for individuals in remote or underserved areas.

Collaborative Research and Data Sharing: Collaborative efforts and data sharing initiatives among researchers, healthcare institutions, and technology providers can accelerate the development and validation of disease prediction models. Large-scale collaborations can lead to the creation of comprehensive datasets, benchmarking frameworks, and standardized evaluation protocols.

Ethical and Privacy Considerations: As disease prediction systems become more widespread, ensuring ethical use of patient data, maintaining privacy, and addressing potential biases and discrimination in the predictions will remain critical areas of focus. Ethical guidelines and regulatory frameworks should be developed and updated to ensure responsible deployment and usage of disease prediction systems.

Overall, the future of MULTIPLE DISEASE PREDICTION SYSTEM system projects lies in improving accuracy, personalization, integration with real-time data, and advancing the field towards proactive healthcare interventions. Continued research, technological advancements, and collaborations across various disciplines will contribute to the further development and adoption of these systems for the benefit of patients and healthcare providers.

CHAPTER 8 – REFERENCES

1) Priyanka Sonar, Prof. K. JayaMalini," DIABETES PREDICTION USING DIFFERENT MACHINE LEARNING APPROACHES", 2019 IEEE ,3rd International Conference on Computing Methodologies and Communication (ICCMC)

2) [2] Archana Singh ,Rakesh Kumar, "Heart Disease Prediction Using Machine Learning Algorithms", 2020 IEEE, International Conference on Electrical and Electronics Engineering (ICE3)

3) [3] A.Sivasangari, Baddigam Jaya Krishna Reddy,Annamareddy Kiran, P.Ajitha," Diagnosis of Liver Disease using Machine Learning Models" 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)