

**Artificial Intelligence & Expert System (CT-361)**

**Project Report**

**Multi-Disease Prediction System Using Machine Learning and Streamlit**

**GROUP MEMBERS:**

ALIZEH MOHSIN (CR-23)

MAHNOOR SHAMIM(CR-49)

## **Introduction:**

Chronic and life-threatening diseases such as diabetes, heart conditions, Parkinson’s disease, lung cancer, and thyroid disorders pose a significant challenge to global healthcare systems. A major difficulty in managing these diseases lies in their silent progression—many go undiagnosed in early stages due to the absence of clear symptoms or the unavailability of specialized diagnostic facilities, especially in remote or underdeveloped regions.

With the rapid growth of Artificial Intelligence (AI) and Machine Learning (ML), there is now a promising alternative to traditional diagnostic methods. These technologies allow for the automated analysis of complex medical data, enabling accurate and timely predictions based on patterns that may not be immediately visible to human practitioners.

This project leverages pre-trained supervised ML models and a web-based user interface to build an **AI-powered Multi-Disease Prediction System**. It allows users to enter relevant medical parameters and receive real-time predictions for five critical diseases. By integrating multiple disease models into a single Streamlit application, this system offers an accessible, modular, and scalable solution to aid early diagnosis and promote preventive healthcare.

## **Objectives:**

* The main goals of this project are outlined as follows:
* **Develop a Unified Multi-Disease Prediction System:**
  + Combine multiple supervised ML models, each trained to detect a specific disease—Diabetes, Heart Disease, Parkinson’s Disease, Lung Cancer, and Hypo-Thyroid disorder—into a single application.
* **Design a User-Friendly Web Interface:**
  + Utilize the Streamlit framework to build an intuitive and responsive web app where users can input relevant health data and view predictions in real-time.
* **Enable Modular Integration of Pre-Trained Models:**
  + Structure the application in such a way that each disease-specific model is independently loaded and can be updated or replaced without affecting other components, allowing for ease of maintenance and scalability.
* **Ensure Compatibility with Diverse Input Formats:**
  + Handle numeric and categorical health parameters using Streamlit input widgets, ensuring flexibility in data entry and seamless communication with the backend models.
* **Promote Accuracy and Explainability:**
  + Aim for high accuracy in disease prediction through the use of validated ML models while also offering simple explanations of predictions to improve user trust and understanding.
* **Support Preventive Healthcare:**
  + Provide early risk assessments that can help users seek medical consultation before diseases progress, thereby contributing to improved public health outcomes.

## **Technologies Used:**

|  |  |
| --- | --- |
| **Component** | **Tool/Library** |
| Programming Language | Python |
| Web Framework | Streamlit |
| ML Models | Trained using scikit-learn |
| Serialization Format | Pickle (.sav files) |
| UI Enhancements | HTML/CSS within Streamlit |
| Disease Data Source | Public Datasets from Kaggle/UCI |

## **System Architecture Overview:**

1. **Frontend (User Interface):**
   1. Built with Streamlit.
   2. Accepts user inputs through text and number input fields.
   3. Displays disease predictions and suggestions.
2. **Backend (Machine Learning Models):**
   1. Models loaded using pickle.
   2. Separate model for each disease.
   3. Inputs converted to numpy arrays and passed to .predict() method.
3. **Prediction Flow:**
   1. User selects a disease from dropdown.
   2. Inputs parameters relevant to selected disease.
   3. Clicks prediction button.
   4. System returns diagnosis based on ML model's prediction.

## **Supported Diseases and Models Used:**

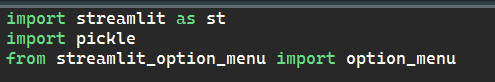
1. **Diabetes:**
   1. Model: Logistic Regression / Random Forest
   2. Inputs: Pregnancies, Glucose, Blood Pressure, etc.
2. **Heart Disease:**
   1. Model: Logistic Regression / Decision Tree
   2. Inputs: Age, Sex, Chest Pain, Blood Pressure, etc.
3. **Parkinson’s Disease:**
   1. Model: Support Vector Machine
   2. Inputs: 22 voice measurement parameters.
4. **Lung Cancer:**
   1. Model: Logistic Regression
   2. Inputs: Symptoms like Smoking, Anxiety, Chest Pain, etc.
5. **Hypo-Thyroidism:**
   1. Model: Random Forest
   2. Inputs: TSH, T3, TT4 levels and demographic data.

## **Key Features:**

* Centralized platform for multiple disease predictions.
* Interactive UI with background customization and animations.
* Minimal design with essential medical parameters.
* High-speed inference using lightweight ML models.
* Easy to deploy on web servers or localhost.

#### **Code Implementation:**

1. **Imports and Setup:**



**Explanation:**

* streamlit as st:  
   This imports the Streamlit library and allows us to use all its functions by calling st.function\_name(). Streamlit is used to build fast and interactive web apps for data science and ML.
* pickle:  
   A standard Python module used for saving and loading Python objects. Here, it loads the pre-trained ML models stored in .sav format.

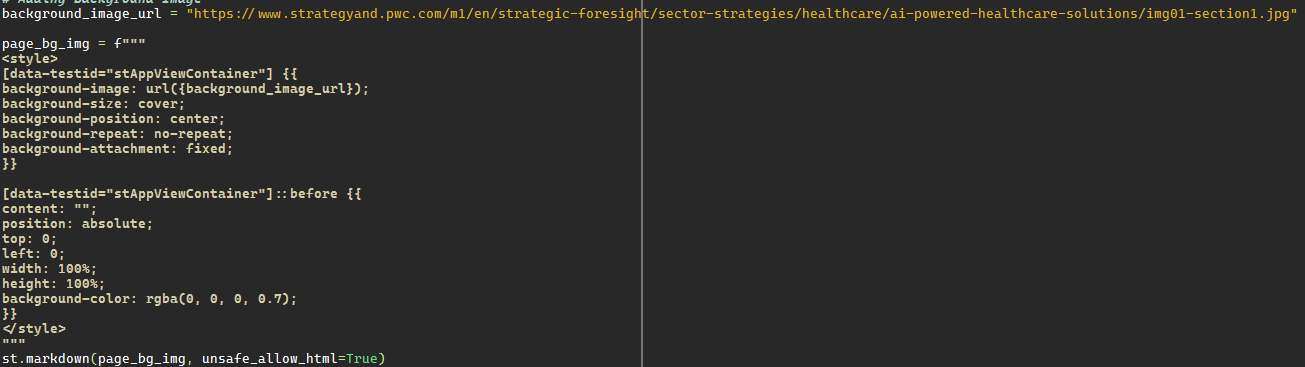
1. **Streamlit Page Configuration and UI Customization:**



### **Explanation:**

* Sets the configuration of your Streamlit app:
  + page\_title: The title shown on the browser tab.
  + page\_icon: A medical symbol emoji for visual identity.
  + You can also configure layout, sidebar state.

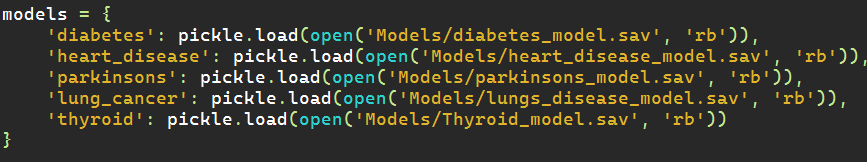
1. **Custom Background Image for UI Aesthetics:**



### **Explanation:**

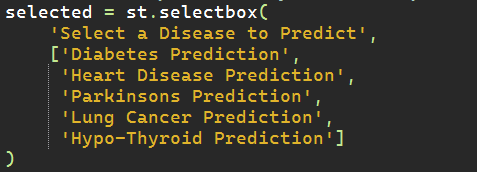
* CSS styles applied to the main app ([data-testid="stApp"]):
  + **Background image** is set and blended with a dark overlay using background-blend-mode: darken to make white text readable.
  + padding, color, and positioning improve UX.

1. **Load Pre-trained Machine Learning Models:**

**Explanation:**

* A dictionary models is created to store each model using a key.
* pickle.load(open(..., 'rb')) opens each file in binary read mode and loads the model.
* These models are likely trained using scikit-learn or similar libraries

1. **Select box for Disease Selection:**

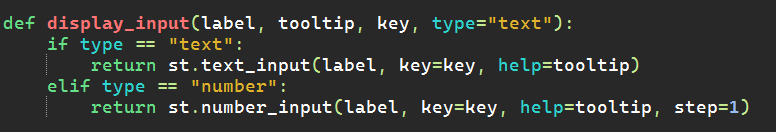


### **Explanation:**

Displays a dropdown list for the user to choose a disease.

The selection is stored in the variable selected.

1. **Custom Function for User Inputs:**

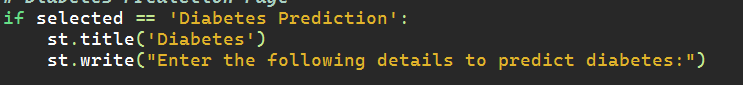
**Explanation:**

* A helper function to show input fields:
  + label: Text shown to the user.
  + tooltip: Hover message explaining the field.
  + key: A unique key to prevent duplicate widgets.
  + type: Can be "number" or "text" to determine the input style.

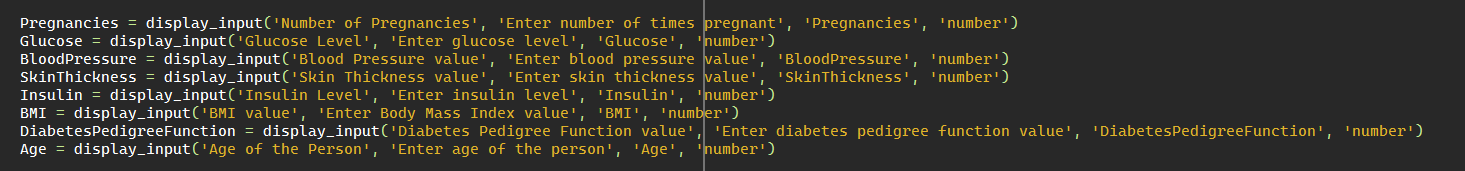
1. **Pages for Each Disease:**

Each condition like if selected == 'Diabetes Prediction': defines a full prediction page.

Let’s break down **one full example** — **Diabetes Prediction**:

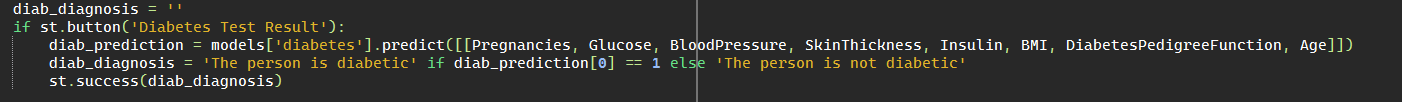
Shows a header/title at the top of the app.

**Input Fields:**



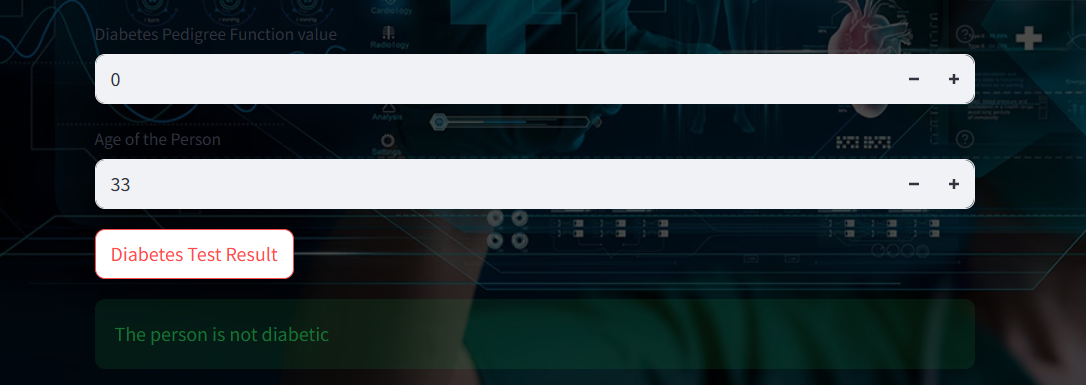
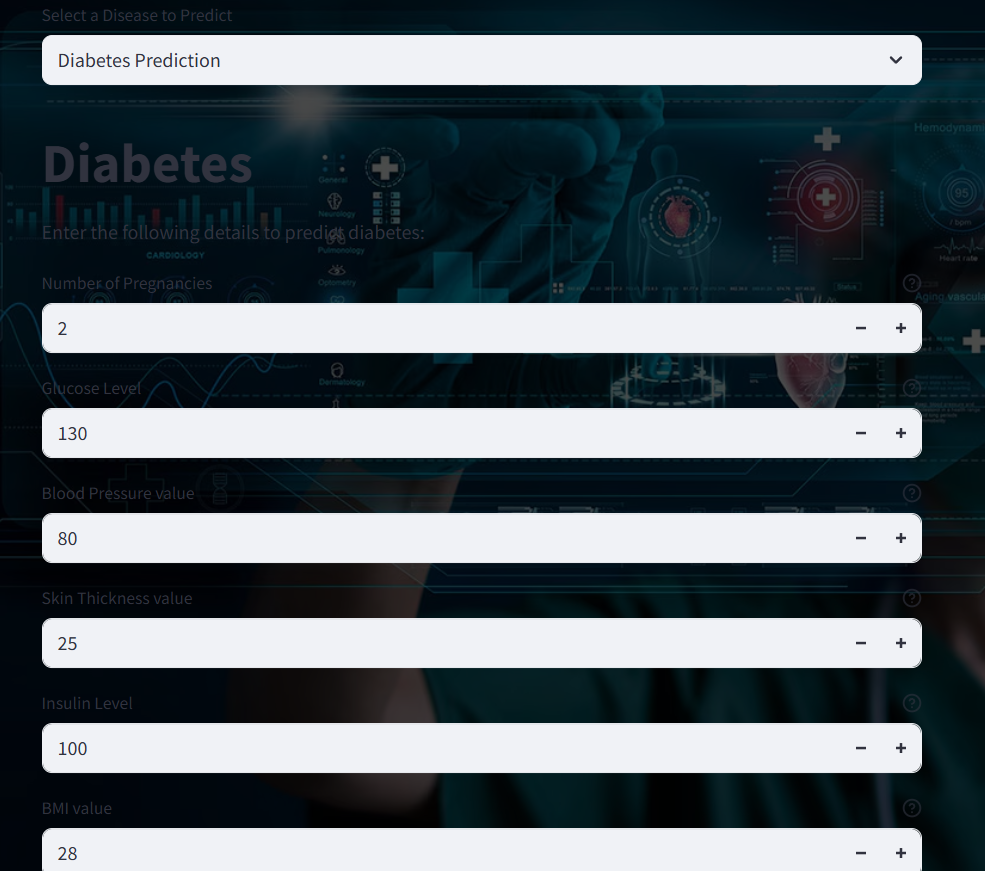
* Calls display\_input() eight times for eight different inputs.
* All values are stored in variables like Glucose, BMI, etc.

**Prediction:**

When the button is clicked:

* The model runs prediction with a list of 8 inputs.
* If prediction is 1 → Person is diabetic.
* If prediction is 0 → Person is not diabetic.
* The result is shown using st.success().

**OUTPUT:**



## **Justification as a Complex Computing Problem:**

This project satisfies the characteristics of a **Complex Computing Problem (CCP)** due to:

1. **Interdisciplinary Integration:**
   1. Combines knowledge of machine learning, healthcare, and UI/UX design.
2. **High Dimensionality and Data Preprocessing:**
   1. Each model handles multi-parameter input with specific preprocessing needs.
3. **Model Management and Modularity:**
   1. Efficient loading and prediction using five separate models.
   2. Requires modular programming and memory optimization.
4. **Real-Time Prediction:**
   1. Processes multiple inputs and delivers prediction instantly.
   2. Requires error handling, input validation, and UI responsiveness.
5. **Scalability:**
   1. Can be extended to more diseases.
   2. Flexible architecture allows easy model swapping/upgrading.
6. **Security and Reliability:**
   1. Models are loaded using serialization (pickle), requiring secure and consistent I/O.
   2. Sensitive medical data is handled through secure local execution or controlled deployment.
7. **Deployment and UI Complexity:**
   1. Streamlit integration with HTML/CSS for customization.
   2. Synchronization of front-end selection and back-end predictions.