**GUI DOCUMENTATION**

***A.app.py***

This `app.py` file is a Flask application that serves as a web interface for making predictions using the heart attack prediction model. Let's break down the code and explain its functionality:

1. Flask Application:

- Flask is a micro web framework for Python that allows you to build web applications quickly and with minimal boilerplate code.

- The `Flask` class is used to create an instance of a web application.

- Routes are defined using decorators like `@app.route('/')` and `@app.route('/predict', methods=['POST'])`, which specify the URL paths and HTTP methods that the application should respond to.

2. Model Loading:

- The code loads a pre-trained machine learning model (`stacking\_clf`) using `joblib.load()`. This model was previously trained using machine learning algorithms on heart-related data to predict the likelihood of a heart attack.

- It also loads a scaler (`scaler`) that was used to standardize the input features during model training.

3. Data Preparation:

- The code reads a dataset (`hearthope.csv`) containing heart-related data using `pandas.read\_csv()`.

- It separates the dataset into features (`X`) and the target variable (`y`).

- The dataset is split into training, validation, and test sets using `train\_test\_split()`. The features are standardized using a scaler to ensure consistency in data distribution.

4. Web Form:

- The home route (`'/'`) renders an HTML template (`index.html`) containing a form.

- This form allows users to input heart-related data such as age, sex, chest pain type, etc.

5. Prediction Handling:

- The predict route (`'/predict'`) handles the POST request sent from the form submission.

- It reads the form data, creates a DataFrame with the input data, scales it using the pre-loaded scaler, and selects the top 5 features.

- The model (`stacking\_clf`) then makes predictions on the preprocessed input data.

- The prediction result is rendered using the `result.html` template.

6. HTML Templates:

- The `index.html` template contains a form for user input.

- The `result.html` template displays the prediction result.

7. Main Execution:

- The `if \_\_name\_\_ == '\_\_main\_\_':` block ensures that the Flask app runs only when the script is executed directly, not when imported as a module.

In summary, the code implements a Flask web application that provides a user interface for predicting heart attack risk based on input data. It leverages a pre-trained machine learning model and scaler to make real-time predictions on user-submitted data.

***B.Index.html***

We've created a user-friendly form for predicting heart disease risk based on various input features related to an individual's health. The form collects information such as age, sex, chest pain type, resting blood pressure, serum cholesterol levels, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, and other relevant factors.

Each input field in the form corresponds to a specific feature used in the prediction model. Users can input their data into these fields, and upon submission, the form sends a POST request to the `/predict` route of the Flask application.

The form submission triggers the prediction process, where the Flask app receives the input data, preprocesses it, and passes it to the pre-trained machine learning model for prediction. Once the prediction is made, the result is displayed on the webpage, informing the user whether they are at risk of heart disease or not.

Overall, HTML template serves as the frontend interface for users to interact with the heart disease prediction system, providing a simple and intuitive way for individuals to assess their potential risk of heart-related issues based on their health parameters.

**C.Result.html**

This HTML template (`result.html`) is designed to display the prediction result based on the output of the heart disease prediction model. It provides a clear and concise message to the user regarding their risk of heart disease.

Upon receiving the prediction from the Flask application, the template checks if the prediction value is 1 or 0. If the prediction is 1, indicating a high likelihood of heart disease, the template displays a message stating "High chances of heart attack" in a visually prominent color (tomato red). Conversely, if the prediction is 0, indicating no significant risk of heart disease, the template displays a message stating "No chances of heart attack" in a contrasting color (lime green).

The design is minimalistic, focusing on delivering the prediction result in a straightforward manner, ensuring clarity for the user. The use of distinct colors helps emphasize the prediction outcome, making it easily noticeable and understandable at a glance.