Action plan for making Heart attack predictions Model

1. Problem Definition

- Build a machine learning-based **Heart Attack Prediction System** to assess the likelihood of a heart attack based on user input.
- Provide a user-friendly interface for users to input their details and get predictions in real-time.

---

2. Steps to Build the System

Step 1: Data Preprocessing

1. **Import Libraries**:
    - Use pandas for data manipulation.
    - Use scikit-learn for preprocessing and model training.
    - Use matplotlib and seaborn for visualization.
2. **Data Cleaning**:
    - Handle missing values (if any) using mean/mode imputation.
    - Normalize numeric features like age, cholesterol levels, etc.
    - Encode categorical variables using one-hot encoding.
3. All the dependent and independent variable related information including graphical representation in uploaded in Github you can access the report via this link
   Link: https://github.com/Guptaji-0/Health_Prediction-/blob/main/Proceesed_Report.html
   Download the file in your machine to view the report
4. **Data Splitting**:
    - Divide the preprocessed data into a **70-30 train-test split**.
    - Use stratified sampling if the target variable is imbalanced.

Step 2: Feature Selection

1. Analyze feature importance using:
    - Correlation matrix.
    - Feature importance from tree-based models (e.g., Random Forest).
    - Recursive feature elimination.

---

3. Machine Learning Model Development

Step 1: Model Selection

- Train and evaluate the following models:
    1. **Logistic Regression**: Suitable for binary classification problems.
    2. **Naive Bayes**: Works well with categorical and continuous data.
    3. **Random Forest**: Robust model to handle non-linear relationships.
    4. **Decision Tree**: Easy to interpret but prone to overfitting.
    5. **Support Vector Machine (SVM)**: Effective for high-dimensional spaces.

Step 2: Model Training

- Train all selected models using the **training data**.
- Use appropriate hyperparameter tuning methods such as Grid Search or Randomized Search to optimize each model.

Step 3: Model Evaluation

- Evaluate models using:
    - **Accuracy**: Percentage of correct predictions.
    - **Precision, Recall, and F1 Score**: To assess model balance.
- Choose the **best-performing model** based on the evaluation metrics.

---

4. Deployment

Step 1: Create Web Application

- **Use Streamlit** to develop an interactive UI for the prediction system.

Step 2: Frontend Features

1. Input fields for user details:
    - Age, Gender, Cholesterol, Resting BP, etc.
2. Predict Button:
    - On click, fetch data, pass to the trained model, and display results.
3. Visualization:
    - Show insights such as risk factor distribution.

Step 3: Backend Implementation

1. Load the trained model as pickel file
2. Process user input, make predictions, and return results in real-time.

---

5. Testing and Validation

- Test the web application with various input scenarios.
- Perform load testing to ensure scalability.
- Validate predictions with domain experts to ensure real-world applicability.

---

6. Deployment

- Deploy the application using cloud platforms such as:
    - **Heroku**
    - **Stramlit cloud**

---

7. Deliverables

1. **ML Model**: Pretrained best-performing model.
2. **Web Application**: Streamlit-based user interface.