Heart Disease Prediction Project

# 1. Import Libraries

This section imports all the necessary Python libraries used in the project:  
  
- pandas: for data manipulation and analysis  
- numpy: for numerical operations  
- matplotlib.pyplot & seaborn: for data visualization  
- sklearn: for machine learning operations including model building, preprocessing, and evaluation  
  
Code Explanation:  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import LabelEncoder, StandardScaler  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_curve, auc  
  
Each import statement brings specific functionality used in the project.

# 2. Load Dataset and Initial Exploration

The dataset is loaded into a pandas DataFrame and initial info about it is displayed.  
  
df = pd.read\_csv('heart\_disease\_uci.csv') # Loads CSV data into a DataFrame  
df.columns = df.columns.str.strip() # Removes whitespace from column names  
print(df.head()) # Prints first 5 rows  
print(df.info()) # Prints concise summary of DataFrame including data types and null counts  
print(df.describe()) # Shows statistical summary of numeric columns  
  
This helps us understand data size, types, and initial stats.

# 3. Handling Categorical Data

Categorical columns (text data) are converted to numeric codes using LabelEncoder.  
This is necessary because machine learning models require numeric input.  
Code:  
for column in df.columns:  
 if df[column].dtype == 'object':  
 le = LabelEncoder()  
 df[column] = le.fit\_transform(df[column])  
 print(f"Encoded '{column}'")  
  
Explanation:  
- Loop through each column  
- If data type is 'object' (text), apply label encoding  
- Replace original text with numeric labels

# 4. Handle Missing Values

Missing or invalid values are replaced with NaN, converted to numeric,  
and then filled with the median of the column.  
Code:  
df.replace('?', np.nan, inplace=True) # Replace '?' with NaN  
df = df.apply(pd.to\_numeric, errors='coerce') # Convert all to numeric, invalid to NaN  
df.fillna(df.median(), inplace=True) # Fill NaNs with median values  
print(df.isnull().sum()) # Confirm no missing values remain  
  
This ensures a clean dataset for modeling.

# 5. Target Column Preparation

Rename target column and convert multi-class to binary classification.  
Code:  
df.rename(columns={'num': 'target'}, inplace=True)  
df['target'] = (df['target'] > 0).astype(int) # 0 = no disease, 1 = disease  
  
This simplifies the problem to a binary classification task.

# 6. Exploratory Data Analysis (EDA)

Several plots visualize correlations and distributions:  
- Correlation Heatmap  
- Target class count plot  
- Distribution of features  
  
Code examples:  
sns.heatmap(df.corr(), annot=True, cmap='coolwarm') # Correlation matrix heatmap  
sns.countplot(x='target', data=df) # Target class distribution  
sns.histplot(df[col], kde=True) # Distribution plots for features  
  
These visuals help identify feature relationships and data balance.

# 7. Feature Scaling and Train-Test Split

Features are separated from target and standardized.  
Then data is split into training and testing sets.  
  
Code:  
X = df.drop('target', axis=1) # Features  
y = df['target'] # Target  
scaler = StandardScaler()  
X = scaler.fit\_transform(X) # Scale features  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) # Split  
  
StandardScaler normalizes feature values for better model performance.

# 8. Model Training with Logistic Regression

Train a logistic regression classifier on training data.  
Code:  
model = LogisticRegression(max\_iter=1000)  
model.fit(X\_train, y\_train)  
  
Logistic regression models the probability of target classes using a sigmoid function.  
max\_iter=1000 allows enough iterations for convergence.

# 9. Model Evaluation

Make predictions on the test set and evaluate accuracy and detailed metrics.  
Code:  
y\_pred = model.predict(X\_test)  
print(accuracy\_score(y\_test, y\_pred))  
print(classification\_report(y\_test, y\_pred))  
  
The classification report includes precision, recall, F1-score, and support for each class.

# 10. Confusion Matrix

Visualize the confusion matrix to see true positives, false positives, true negatives, and false negatives.  
Code:  
cm = confusion\_matrix(y\_test, y\_pred)  
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  
plt.xlabel('Predicted')  
plt.ylabel('Actual')  
plt.title('Confusion Matrix')  
plt.show()  
  
This matrix helps understand model errors and strengths.

# 11. ROC Curve and AUC

Plot ROC curve and calculate AUC to evaluate model discrimination ability.  
Code:  
y\_proba = model.predict\_proba(X\_test)[:, 1]  
fpr, tpr, \_ = roc\_curve(y\_test, y\_proba)  
roc\_auc = auc(fpr, tpr)  
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc\_auc:.2f})')  
plt.plot([0, 1], [0, 1], 'k--')  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('ROC Curve')  
plt.legend()  
plt.show()  
  
AUC near 1 means excellent discrimination; 0.5 means random guessing.

# Algorithm Explanation: Logistic Regression

Logistic Regression is a supervised classification algorithm used to predict the probability of a binary outcome.  
It uses the logistic function (sigmoid) to map linear combinations of features to probabilities between 0 and 1.  
  
Mathematically, logistic regression models:  
 p = 1 / (1 + exp(- (b0 + b1\*x1 + ... + bn\*xn)))  
  
Where:  
- p is the probability of class 1 (heart disease presence)  
- x1, ..., xn are feature values  
- b0, b1, ..., bn are model coefficients learned during training  
  
The model is trained by maximizing the likelihood of observed data or minimizing a loss function like log loss.  
After training, a threshold (usually 0.5) is used to decide class labels.  
  
Advantages:  
- Simple, interpretable  
- Efficient for binary classification  
  
In this project, logistic regression predicts whether a patient has heart disease.