**💓 Heart Disease Prediction - Project Overview**

**📝 Project Summary**

This project builds a machine learning model using the **Cleveland Heart Disease Dataset** from the UCI repository to **predict the risk of heart disease** based on clinical parameters. The model uses **Random Forest Classifier** for prediction and provides a confidence score for individual inputs.

**📚 Libraries Used**

| **Library** | **Purpose** |
| --- | --- |
| pandas | To load and manipulate structured data |
| scikit-learn | To build, train, and evaluate the ML model |
| joblib | To save and load the trained model |
| numpy | (implicitly used via pandas/sklearn) for numeric operations |

To install all required libraries:

pip install pandas scikit-learn joblib

**📂 Dataset Information**

* **Source:** [UCI Heart Disease Dataset](https://archive.ics.uci.edu/ml/datasets/heart+Disease)
* **File:** processed.cleveland.data
* **Attributes Used (13 Features + Target):**

| **Feature** | **Description** |
| --- | --- |
| age | Age of the patient (in years) |
| sex | 1 = Male, 0 = Female |
| cp | Chest pain type (0-3) |
| trestbps | Resting blood pressure (mm Hg) |
| chol | Serum cholesterol (mg/dL) |
| fbs | Fasting blood sugar > 120 mg/dL (1 = Yes) |
| restecg | Resting ECG results |
| thalach | Maximum heart rate achieved |
| exang | Exercise-induced angina (1 = Yes) |
| oldpeak | ST depression induced by exercise |
| slope | Slope of the ST segment (0 = up, 1 = flat, 2 = down) |
| ca | Major vessels colored (0–3) |
| thal | Thalassemia (1 = fixed defect, 2 = reversible, 3 = normal) |
| target | 0 = No heart disease, 1 = Heart disease (binary) |

**🔧 Steps Performed**

**🔹 Step 1: Import Required Libraries**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import joblib

**🔹 Step 2: Load and Clean Dataset**

* Downloaded raw data from the UCI website.
* Replaced '?' (missing values) with pd.NA and removed them using dropna().
* Converted relevant columns (ca, thal, target) to numeric type.
* Converted the target into **binary classification**: 0 = no disease, 1 = has disease.

**🔹 Step 3: Prepare Features and Target**

X = df.drop("target", axis=1)

y = df["target"]

**🔹 Step 4: Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**🔹 Step 5: Train the Random Forest Model**

python

CopyEdit

model = RandomForestClassifier(n\_estimators=100, class\_weight='balanced', random\_state=42)

model.fit(X\_train, y\_train)

**🔹 Step 6: Evaluate Model Performance**

y\_pred = model.predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**🔹 Step 7: Save the Trained Model**

joblib.dump(model, "heart\_disease\_model.pkl")

**🔹 Step 8: Predict for a Custom Patient**

Example input:

python

CopyEdit

input\_dict = {

'age': [45],

'sex': [0],

'cp': [1],

'trestbps': [130],

'chol': [220],

'fbs': [0],

'restecg': [0],

'thalach': [180],

'exang': [0],

'oldpeak': [0.0],

'slope': [2],

'ca': [0],

'thal': [1]

}

input\_df = pd.DataFrame(input\_dict)

**🔹 Step 9: Get Prediction & Risk Confidence**

model = joblib.load("heart\_disease\_model.pkl")

proba = model.predict\_proba(input\_df)

# Confidence & risk level

if proba[0][1] > 0.7:

print("❗ High Risk of Heart Disease")

elif proba[0][1] < 0.3:

print("✅ Low Risk of Heart Disease")

else:

print("⚠️ Moderate Risk of Heart Disease")

**✅ Output Sample**

📊 Confusion Matrix:

[[25 5]

[ 4 27]]

📄 Classification Report:

precision recall f1-score support

0 0.86 0.83 0.84 30

1 0.84 0.87 0.85 31

accuracy 0.85 61

macro avg 0.85 0.85 0.85 61

weighted avg 0.85 0.85 0.85 61

🔍 Prediction Confidence: [[0.89 0.11]]

✅ Low Risk of Heart Disease

**🧠 Model Notes**

* **Random Forest** was chosen due to its robustness and good performance on tabular health data.
* Model handles **imbalanced classes** using class\_weight='balanced'.
* Easy to extend for web apps using Flask or Streamlit.