**SOURCE CODE**

import pandas as pd

df = pd.read\_csv('/content/HeartDiseaseTrain-Test.csv')

# 3️⃣ Data Exploration

print("\nFirst 5 rows of the dataset:")

print(df.head())

print("\nShape of the dataset:", df.shape)

print("\nColumns:", df.columns.tolist())

print("\nData Types & Missing Values:")

df.info()

print("\nSummary Statistics:")

print(df.describe())

print("\nMissing values per column:\n", df.isnull().sum())

print("\nDuplicate rows:", df.duplicated().sum())

# 4️⃣ Data Visualization (Optional, for better understanding)

import seaborn as sns

import matplotlib.pyplot as plt

# Example: Plot distribution of target variable (assuming 'target' is the disease indicator)

sns.countplot(x='target', data=df)

plt.title('Distribution of Disease Presence (0=No, 1=Yes)')

plt.show()

# 5️⃣ Prepare Features and Target

target = 'target'  # Change to actual target column if named differently

features = df.columns.drop(target)

print("\nFeatures:", features.tolist())

# 6️⃣ Convert Categorical Columns

categorical\_cols = df.select\_dtypes(include=['object']).columns.tolist()

print("\nCategorical Columns:", categorical\_cols)

df\_encoded = pd.get\_dummies(df, drop\_first=True)

# 3️⃣ Check for Missing Values

print("\n🟠 Missing values per column:\n", df.isnull().sum()

# Fill missing numeric columns with median

numeric\_cols = df.select\_dtypes(include=['int64', 'float64']).columns

for col in numeric\_cols:

    if df[col].isnull().sum() > 0:

        median\_val = df[col].median()

        df[col].fillna(median\_val, inplace=True)

        print(f"Filled missing values in '{col}' with median: {median\_val}")

# Fill missing categorical columns with mode

categorical\_cols = df.select\_dtypes(include=['object']).columns

for col in categorical\_cols:

    if df[col].isnull().sum() > 0:

        mode\_val = df[col].mode()[0]

        df[col].fillna(mode\_val, inplace=True)

        print(f"Filled missing values in '{col}' with mode: {mode\_val}")

# 4️⃣ Check for Duplicates

duplicates = df.duplicated().sum()

print(f"\n🟠 Duplicate rows: {duplicates}")

if duplicates > 0:

    df.drop\_duplicates(inplace=True)

    print("✅ Duplicates removed.")

# 5️⃣ Detect and Handle Outliers (Optional: here we cap them using IQR method)

def cap\_outliers(column):

    Q1 = df[column].quantile(0.25)

    Q3 = df[column].quantile(0.75)

    IQR = Q3 - Q1

    lower = Q1 - 1.5 \* IQR

    upper = Q3 + 1.5 \* IQR

    df[column] = df[column].clip(lower, upper)

    print(f"✅ Outliers capped for '{column}'")

for col in numeric\_cols:

    cap\_outliers(col)

# 6️⃣ Encode Categorical Features

print("\n✅ Categorical Columns:", categorical\_cols.tolist())

df\_encoded = pd.get\_dummies(df, drop\_first=True)

# 7️⃣ Feature Scaling

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X = df\_encoded.drop('target', axis=1)  # Replace 'target' with your actual target column

y = df\_encoded['target']

X\_scaled = scaler.fit\_transform(X)

# 8️⃣ Train-Test Split

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X\_scaled, y, test\_size=0.2, random\_state=42

)

# 9️⃣ Model Training

from sklearn.ensemble import RandomForestClassifier

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

model = RandomForestClassifier(random\_state=42)

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

# 🔟 Predictions

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

# 🔍 Evaluation

print("\nAccuracy:", accuracy\_score(y\_test, y\_pred))

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

# 🔮 Predicting a new patient

# Example: Replace with actual input values

new\_patient = {

    'age': 55,

    'sex': 1,

    'cp': 3,

    'trestbps': 140,

    'chol': 250,

    'fbs': 0,

    'restecg': 1,

    'thalach': 150,

    'exang': 0,

    'oldpeak': 2.3,

    'slope': 0,

    'ca': 0,

    'thal': 2

}

# Convert input to DataFrame

new\_df = pd.DataFrame([new\_patient])

# Combine with original data to ensure same columns

df\_temp = pd.concat([df.drop(target, axis=1), new\_df], ignore\_index=True)

df\_temp\_encoded = pd.get\_dummies(df\_temp, drop\_first=True)

# Reindex to match training columns

df\_temp\_encoded = df\_temp\_encoded.reindex(columns=df\_encoded.drop(target, axis=1).columns, fill\_value=0)

# Scale new input

new\_input\_scaled = scaler.transform(df\_temp\_encoded.tail(1))

# Make prediction

predicted\_disease = model.predict(new\_input\_scaled)

print("\nPredicted Disease Presence (1=Yes, 0=No):", predicted\_disease[0])