Heart Disease Prediction Project

Task Objective:

Predict heart disease risk using patient health data by cleaning, processing, and applying ML models.

Dataset:

UCI Heart Disease dataset with 920 rows and 16 columns including age, sex, chest pain type, blood pressure, cholesterol, etc.

Data Preprocessing:

Handled missing values using median (numeric) and mode (categorical). Converted multi-class target to binary. One-hot encoded categorical features.

Models Applied:

Logistic Regression (scaled numeric features) and Decision Tree (max depth=5).

Evaluation Metrics:

Measured Accuracy, AUC, Confusion Matrix, and ROC curves for both models.

Key Results:

Logistic Regression: Accuracy=84.2%, AUC=0.933
Decision Tree: Accuracy=88.6%, AUC=0.924

Feature Importance:

Top features influencing predictions include chest pain type, thalassemia, ST depression, and exercise-induced angina.

Code:

The following pages contain the complete Python code used in this project.

Heart Disease Prediction Script

```
# ------------------------------
# Heart Disease Prediction Script
# ------------------------------

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve, confusion_matrix

path = "heart_disease_uci.csv"
df = pd.read_csv(path)

target_col = "num"
df[target_col] = df[target_col].apply(lambda x: 1 if x != 0 else 0)

numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = df.select_dtypes(include=['object', 'category']).columns.tolist()
numeric_cols.remove(target_col)

num_imputer = SimpleImputer(strategy="median")
cat_imputer = SimpleImputer(strategy="most_frequent")
df[numeric_cols] = num_imputer.fit_transform(df[numeric_cols])
df[cat_cols] = cat_imputer.fit_transform(df[cat_cols])

df_encoded = pd.get_dummies(df, columns=cat_cols, drop_first=True)
X = df_encoded.drop(columns=[target_col])
y = df_encoded[target_col]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

scaler = StandardScaler()
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()
X_train_scaled[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
X_test_scaled[numeric_cols] = scaler.transform(X_test[numeric_cols])

log_clf = LogisticRegression(max_iter=1000)
log_clf.fit(X_train_scaled, y_train)

tree_clf = DecisionTreeClassifier(max_depth=5, random_state=42)
tree_clf.fit(X_train, y_train)

y_pred_log = log_clf.predict(X_test_scaled)
y_proba_log = log_clf.predict_proba(X_test_scaled)[:, 1]
y_pred_tree = tree_clf.predict(X_test)
y_proba_tree = tree_clf.predict_proba(X_test)[:, 1]

acc_log = accuracy_score(y_test, y_pred_log)
auc_log = roc_auc_score(y_test, y_proba_log)
acc_tree = accuracy_score(y_test, y_pred_tree)
auc_tree = roc_auc_score(y_test, y_proba_tree)

print(f"Logistic Regression: Accuracy={acc_log:.4f}, AUC={auc_log:.4f}")
print(f"Decision Tree: Accuracy={acc_tree:.4f}, AUC={auc_tree:.4f}")

cm_log = confusion_matrix(y_test, y_pred_log)
cm_tree = confusion_matrix(y_test, y_pred_tree)
```

Heart Disease Prediction Script

```
fpr_log, tpr_log, _ = roc_curve(y_test, y_proba_log)
fpr_tree, tpr_tree, _ = roc_curve(y_test, y_proba_tree)

log_importance = pd.Series(log_clf.coef_[0], index=X_train.columns).sort_values(key=abs, ascending=False)
tree_importance = pd.Series(tree_clf.feature_importances_, index=X_train.columns).sort_values(ascending=False)

plt.figure(figsize=(14,8))
plot_tree(tree_clf, feature_names=X_train.columns, class_names=["NoDisease","Disease"], filled=True)
plt.show()
```