**📁 1. Set Up Project**

Create a GitHub repo named:  
task5-decision-tree-random-forest

Folder structure:

task5-decision-tree-random-forest/

│

├── heart.csv # Dataset

├── decision\_tree\_random\_forest.ipynb

├── README.md

└── images/ # For tree plots/screenshots

**📌 2. Import Required Libraries**

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, cross\_val\_score

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.ensemble import RandomForestClassifier

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

**📥 3. Load the Dataset**

df = pd.read\_csv("heart.csv")

print(df.head())

**🧼 4. Preprocess the Data**

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

y = df["target"]

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

**🌳 5. Train a Decision Tree Classifier**

dtree = DecisionTreeClassifier(max\_depth=4, random\_state=42)

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

# Evaluate

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

print("Decision Tree Accuracy:", accuracy\_score(y\_test, y\_pred))

# Visualization

plt.figure(figsize=(20,10))

plot\_tree(dtree, feature\_names=X.columns, class\_names=["No Disease", "Disease"], filled=True)

plt.savefig("images/decision\_tree.png")

plt.show()

**⚠️ 6. Analyze Overfitting & Control Depth**

depths = range(1, 20)

train\_scores = []

test\_scores = []

for d in depths:

model = DecisionTreeClassifier(max\_depth=d, random\_state=42)

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

train\_scores.append(model.score(X\_train, y\_train))

test\_scores.append(model.score(X\_test, y\_test))

plt.plot(depths, train\_scores, label='Train Accuracy')

plt.plot(depths, test\_scores, label='Test Accuracy')

plt.xlabel('Tree Depth')

plt.ylabel('Accuracy')

plt.title('Overfitting Analysis')

plt.legend()

plt.grid(True)

plt.show()

**🌲 7. Train a Random Forest & Compare**

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

rf\_pred = rf.predict(X\_test)

print("Random Forest Accuracy:", accuracy\_score(y\_test, rf\_pred))

**💡 8. Interpret Feature Importances**

importances = rf.feature\_importances\_

feat\_imp = pd.Series(importances, index=X.columns).sort\_values(ascending=False)

# Plot

feat\_imp.plot(kind='bar', title='Feature Importance')

plt.ylabel('Importance')

plt.show()

**✅ 9. Cross-Validation**

cv\_scores = cross\_val\_score(rf, X, y, cv=5)

print("Cross-Validation Accuracy Scores:", cv\_scores)

print("Mean CV Accuracy:", np.mean(cv\_scores))