**Step 1: Install Required Libraries**

Make sure you have these installed:

pip install scikit-learn graphviz matplotlib pandas

*(For Windows, also install Graphviz from graphviz.org/download and add it to your PATH.)*

**✅ Step 2: Import Libraries**

import pandas as pd

import numpy as np

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

from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export\_graphviz

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error

import graphviz

**✅ Step 3: Load Dataset**

**For Classification (e.g., Iris dataset)**

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data

y = iris.target

**For Regression (e.g., Boston Housing dataset)**

from sklearn.datasets import fetch\_california\_housing

housing = fetch\_california\_housing()

X\_reg = housing.data

y\_reg = housing.target

**✅ Step 4: Split Data**

# Classification

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Regression

Xr\_train, Xr\_test, yr\_train, yr\_test = train\_test\_split(X\_reg, y\_reg, test\_size=0.3, random\_state=42)

**✅ Step 5: Train Decision Tree Models**

**Classification Tree**

clf = DecisionTreeClassifier(criterion='gini', max\_depth=3, random\_state=42)

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

**Regression Tree**

reg = DecisionTreeRegressor(max\_depth=3, random\_state=42)

reg.fit(Xr\_train, yr\_train)

**✅ Step 6: Predictions & Evaluation**

**Classification**

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

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

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

**Regression**

y\_pred\_reg = reg.predict(Xr\_test)

print("MSE:", mean\_squared\_error(yr\_test, y\_pred\_reg))

**✅ Step 7: Visualize Tree using Graphviz**

dot\_data = export\_graphviz(clf, out\_file=None,

feature\_names=iris.feature\_names,

class\_names=iris.target\_names,

filled=True, rounded=True,

special\_characters=True)

graph = graphviz.Source(dot\_data)

graph.render("decision\_tree") # Saves as decision\_tree.pdf

graph.view()

For **Regression Tree**, replace clf with reg and adjust feature names accordingly.

**✅ Important Hyperparameters**

* criterion: "gini" or "entropy" for classification, "squared\_error" for regression.
* max\_depth: Limits depth of tree to prevent overfitting.
* min\_samples\_split: Minimum samples needed to split a node.
* min\_samples\_leaf: Minimum samples per leaf node.

**✅ Advantages of Decision Trees**

* Handles **non-linear data**.
* No need for feature scaling.
* Easy to visualize and interpret.

**✅ Disadvantages**

* Can **overfit** easily (use pruning or max\_depth).
* Sensitive to small data changes.

👉 Do you want me to:  
✔ **Provide a single complete code block (ready-to-run)**,  
✔ **Jupyter Notebook style with explanations**, or  
✔ **Generate a PDF report with all steps and diagrams (Graphviz tree visualization included)**?