**Ex:4-Decision Tree Classification**

# Import necessary libraries

from sklearn.datasets import load\_iris

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

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

# Load the Iris dataset

iris = load\_iris()

X = iris.data # Features

y = iris.target # Labels

# Split into train and test sets (80% train, 20% test)

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

# Create the Decision Tree model

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

# Train the model

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

# Predict on test data

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

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

# Visualize the decision tree

plt.figure(figsize=(12, 8))

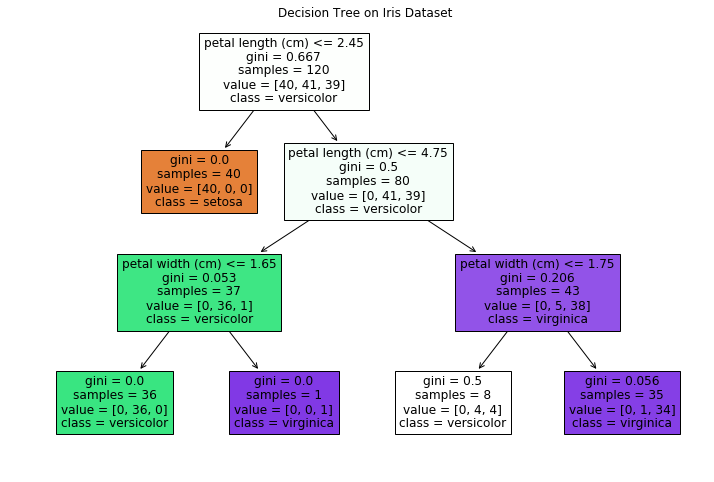
plot\_tree(clf, filled=True, feature\_names=iris.feature\_names, class\_names=iris.target\_names)

plt.title("Decision Tree on Iris Dataset")

plt.show()

**Output:**

**Accuracy: 1.00**

****

**Ex:5-Decision Tree Regression**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeRegressor

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

from sklearn.metrics import mean\_squared\_error, r2\_score

# --- 1. Generate Sample Data ---

# For demonstration purposes, let's create some synthetic data.

# In a real-world scenario, you would load your own dataset.

np.random.seed(42) # for reproducibility

X = np.sort(5 \* np.random.rand(80, 1), axis=0)

y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Introduce some noise or a more complex relationship for better illustration

y[::5] += 3 \* (0.5 - np.random.rand(16))

# --- 2. Split Data into Training and Testing Sets ---

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

# --- 3. Create and Train the Decision Tree Regressor Model ---

# Initialize the DecisionTreeRegressor.

# Key parameters to consider:

# - max\_depth: The maximum depth of the tree. Controls overfitting.

# A deeper tree can capture more complexity but might overfit.

# - min\_samples\_leaf: The minimum number of samples required to be at a leaf node.

# Helps prevent creating leaves with very few samples, reducing noise.

# - random\_state: For reproducibility of the results.

regressor = DecisionTreeRegressor(max\_depth=5, random\_state=42)

# Train the model using the training data

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

# --- 4. Make Predictions ---

y\_pred\_train = regressor.predict(X\_train)

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

# To visualize the full range of the model's predictions,

# let's create a finer grid of X values

X\_grid = np.arange(min(X), max(X), 0.01)[:, np.newaxis]

y\_grid\_pred = regressor.predict(X\_grid)

# --- 5. Evaluate the Model ---

# Mean Squared Error (MSE): Average of the squared differences between predicted and actual values.

mse\_train = mean\_squared\_error(y\_train, y\_pred\_train)

mse\_test = mean\_squared\_error(y\_test, y\_pred\_test)

# R-squared (R2) Score: Represents the proportion of variance in the dependent variable

# that can be predicted from the independent variable(s). Closer to 1 is better.

r2\_train = r2\_score(y\_train, y\_pred\_train)

r2\_test = r2\_score(y\_test, y\_pred\_test)

print(f"--- Model Evaluation ---")

print(f"Training MSE: {mse\_train:.4f}")

print(f"Testing MSE: {mse\_test:.4f}")

print(f"Training R-squared: {r2\_train:.4f}")

print(f"Testing R-squared: {r2\_test:.4f}")

# --- 6. Visualize the Results ---

plt.figure(figsize=(10, 6))

plt.scatter(X\_train, y\_train, s=20, edgecolor="black", c="darkorange", label="Training data")

plt.scatter(X\_test, y\_test, s=20, edgecolor="black", c="cornflowerblue", label="Testing data")

plt.plot(X\_grid, y\_grid\_pred, color="red", linestyle='--', linewidth=2, label="Decision Tree Prediction")

plt.xlabel("Features (X)")

plt.ylabel("Target (y)")

plt.title("Decision Tree Regression")

plt.legend()

plt.show()

# --- Optional: Visualize the Tree Structure (requires graphviz and pydotplus) ---

# If you have graphviz installed and pydotplus:

# from sklearn.tree import export\_graphviz

# import graphviz

#

# dot\_data = export\_graphviz(regressor, out\_file=None,

# feature\_names=["Feature\_X"],

# filled=True, rounded=True,

# special\_characters=True)

# graph = graphviz.Source(dot\_data)

# graph.render("decision\_tree\_regression") # This will save a PDF file of the tree

# graph # To display in a Jupyter Notebook/IPython environment

**Output:**

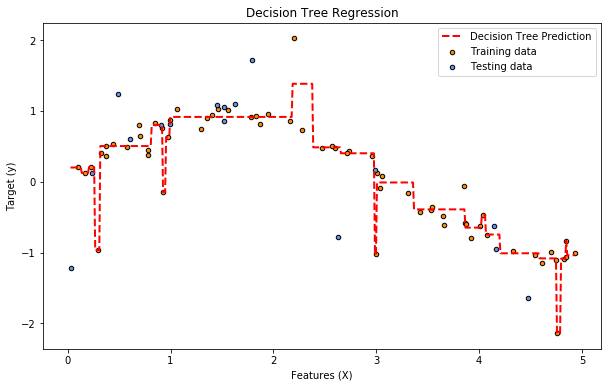
--- Model Evaluation ---

Training MSE: 0.0212

Testing MSE: 0.4227

Training R-squared: 0.9659

Testing R-squared: 0.5573

****