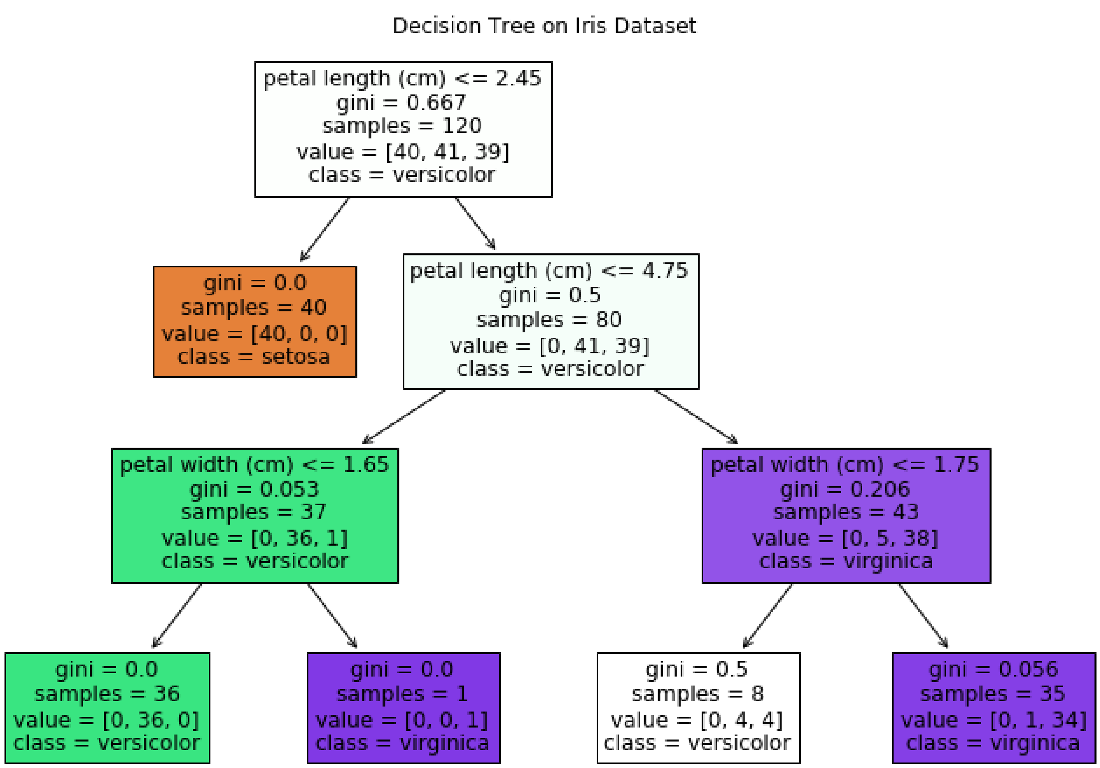
Ex:4-Decision Tree Classification

```
# 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.
```

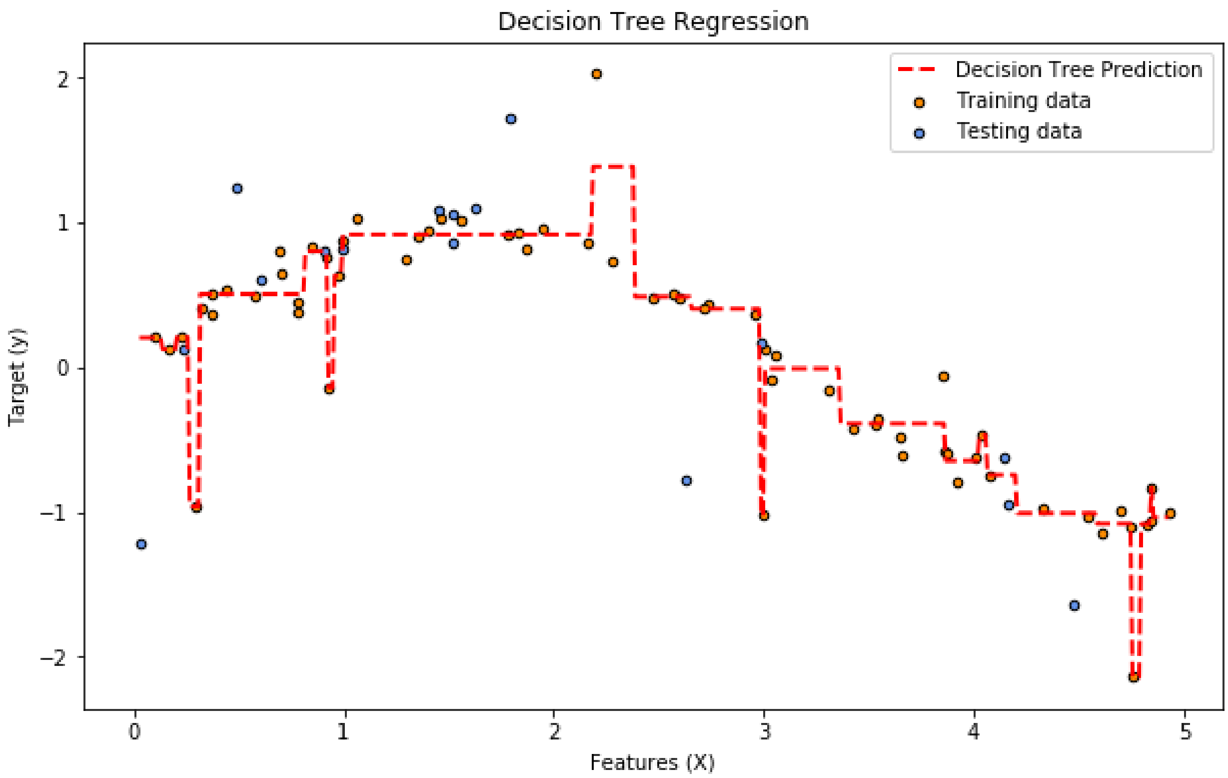
```
# Key parameters to consider:
# - max_depth: The maximum depth of the tree. Controls overfitting.
#           A deeper tree can capture more complexity but might overfit.
# - min_samples_leaf: The minimum number of samples required to be at a leaf node.
#                  Helps prevent creating leaves 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
```


Decision Tree Regression
Decision Tree Prediction
Training data
Testing data
Target (y)
Features (X)
2
1
0
−1
−2
0
1
2
3
4
5