EXPERIMENT-7

A python program to implement decision tree

AIM:

To code a python program to implement decision tree.

CODE:

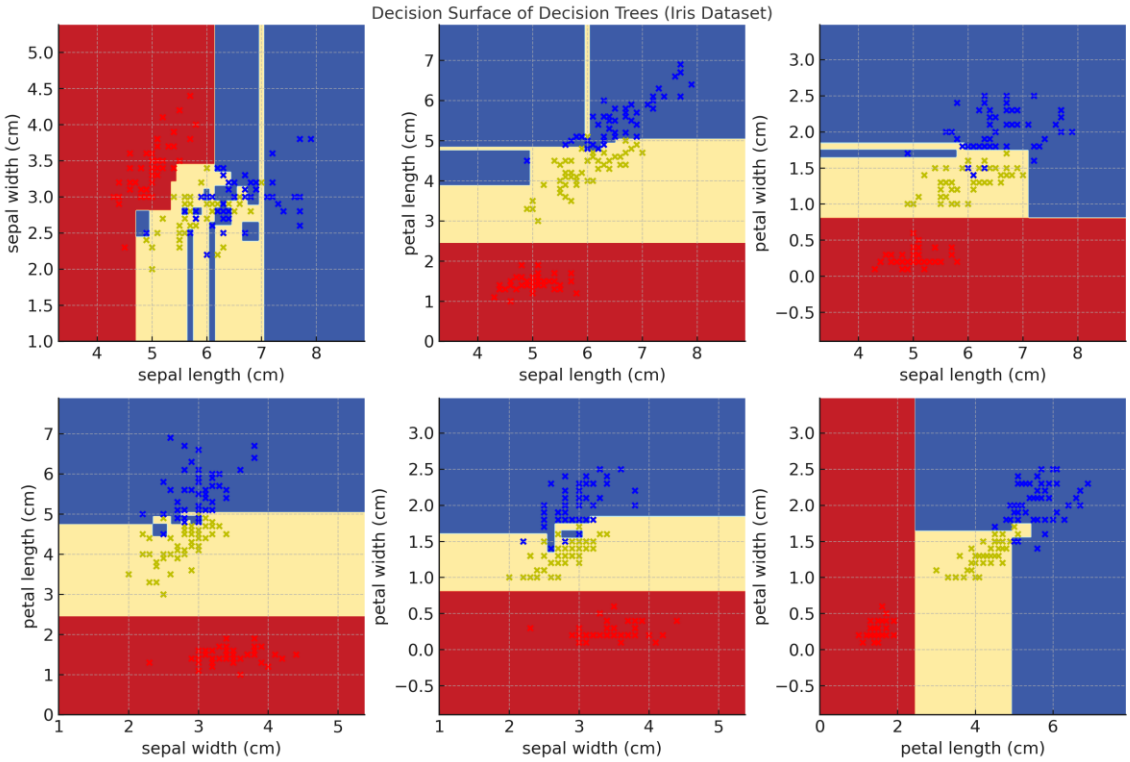
```
# Import required libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import numpy as np
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()

# Parameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02
```

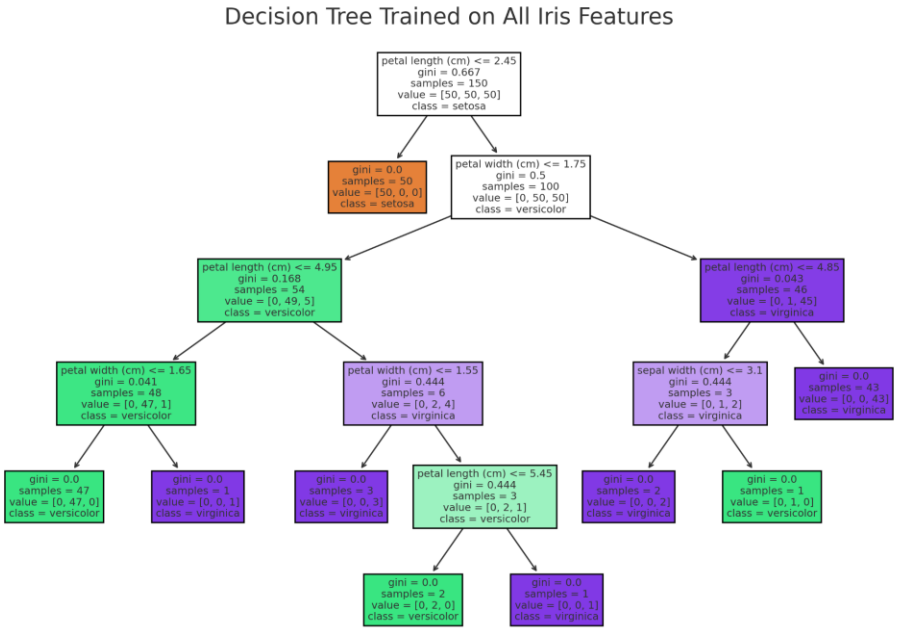
```
# Create a figure for decision boundaries
plt.figure(figsize=(12, 8))

# Loop through all pairs of features
for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]):
    X = iris.data[:, pair]
    y = iris.target

    # Train Decision Tree on the selected features
    clf = DecisionTreeClassifier().fit(X, y)

    # Define plot limits
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    # Create meshgrid
    xx, yy = np.meshgrid(
        np.arange(x_min, x_max, plot_step),
        np.arange(y_min, y_max, plot_step)
    )
```

```
# Predict over the grid
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Create subplot
plt.subplot(2, 3, pairidx + 1)
plt.tight_layout(h_pad=0.5, w_pad=0.5, pad=2.5)

# Plot decision boundary
cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

# Plot the training points
for i, color in zip(range(n_classes), plot_colors):
   idx = np.where(y == i)
   plt.scatter(
      X[idx, 0],
      X[idx, 1],
      c=color,
      label=iris.target_names[i],
      edgecolor="black",
      s=15
   )
```

```
    # Label axes
    plt.xlabel(iris.feature_names[pair[0]])
    plt.ylabel(iris.feature_names[pair[1]])

# Add a common title
plt.suptitle("Decision Surface of Decision Trees (Iris Dataset)")
plt.legend(loc="lower right", borderpad=0, handletextpad=0)
plt.show()

# Plot the Decision Tree trained on all features
plt.figure(figsize=(12, 8))
clf_full = DecisionTreeClassifier().fit(iris.data, iris.target)
plot_tree(clf_full, filled=True,
feature_names=iris.feature_names,
class_names=iris.target_names)
plt.title("Decision Tree Trained on All Iris Features")
plt.show()
```

OUTPUT:


Decision Surface of Decision Trees (Iris Dataset)
sepal width (cm)
sepal length (cm)
petal length (cm)
petal width (cm)


Decision Tree Trained on All Iris Features
petal length (cm) <= 2.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) <= 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) <= 4.85
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) <= 1.65
gini = 0.041
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal width (cm) <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
petal length (cm) <= 5.45
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

RESULT:

Thus python program to implement decision tree is written and the output is verified successfully.