```
# ✅ A PYTHON PROGRAM TO IMPLEMENT DECISION TREE (Iris Dataset)
# -------------------------------------------------------------

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import numpy as np
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

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

# 📌 Plot decision surface using pairs of features
plt.figure(figsize=(15, 10))
feature_pairs = [[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]

for pairidx, pair in enumerate(feature_pairs):
    X_pair = X[:, pair]

    # Train decision tree classifier
    clf = DecisionTreeClassifier().fit(X_pair, y)

    # Decision boundary grid
    x_min, x_max = X_pair[:, 0].min() - 1, X_pair[:, 0].max() + 1
    y_min, y_max = X_pair[:, 1].min() - 1, X_pair[:, 1].max() + 1
```

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

    plt.subplot(2, 3, pairidx + 1)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot decision boundary
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu, alpha=0.6)

    # Plot data points
    for i, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == i)
        plt.scatter(
            X_pair[idx, 0],
            X_pair[idx, 1],
            c=color,
            label=iris.target_names[i],
            edgecolor="black",
            s=20
        )

    plt.xlabel(iris.feature_names[pair[0]])
    plt.ylabel(iris.feature_names[pair[1]])
    plt.title(f"Features: {iris.feature_names[pair[0]]} vs {iris.feature_names[pair[1]]}")

plt.suptitle("Decision Surfaces of Decision Tree (Iris Dataset)")
plt.tight_layout(rect=[0, 0, 1, 0.96])
```

📌 Plot final decision tree trained on all features

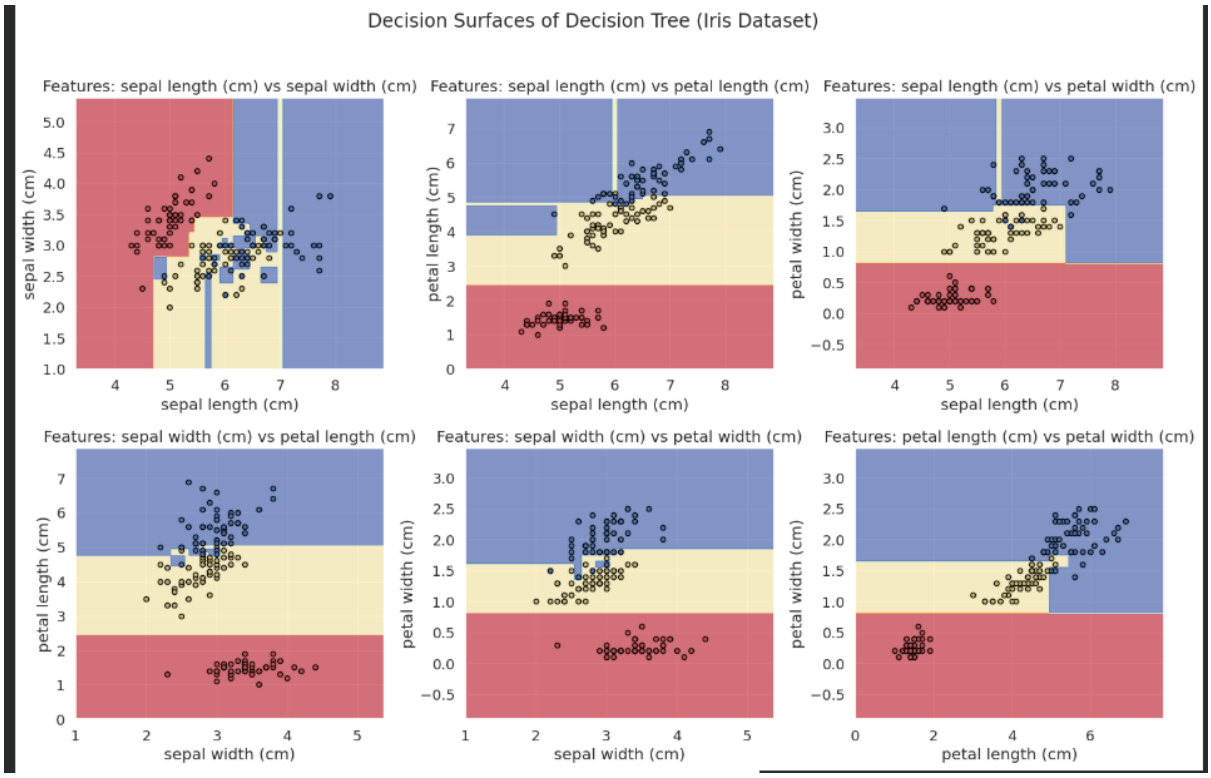
plt.figure(figsize=(12, 8))

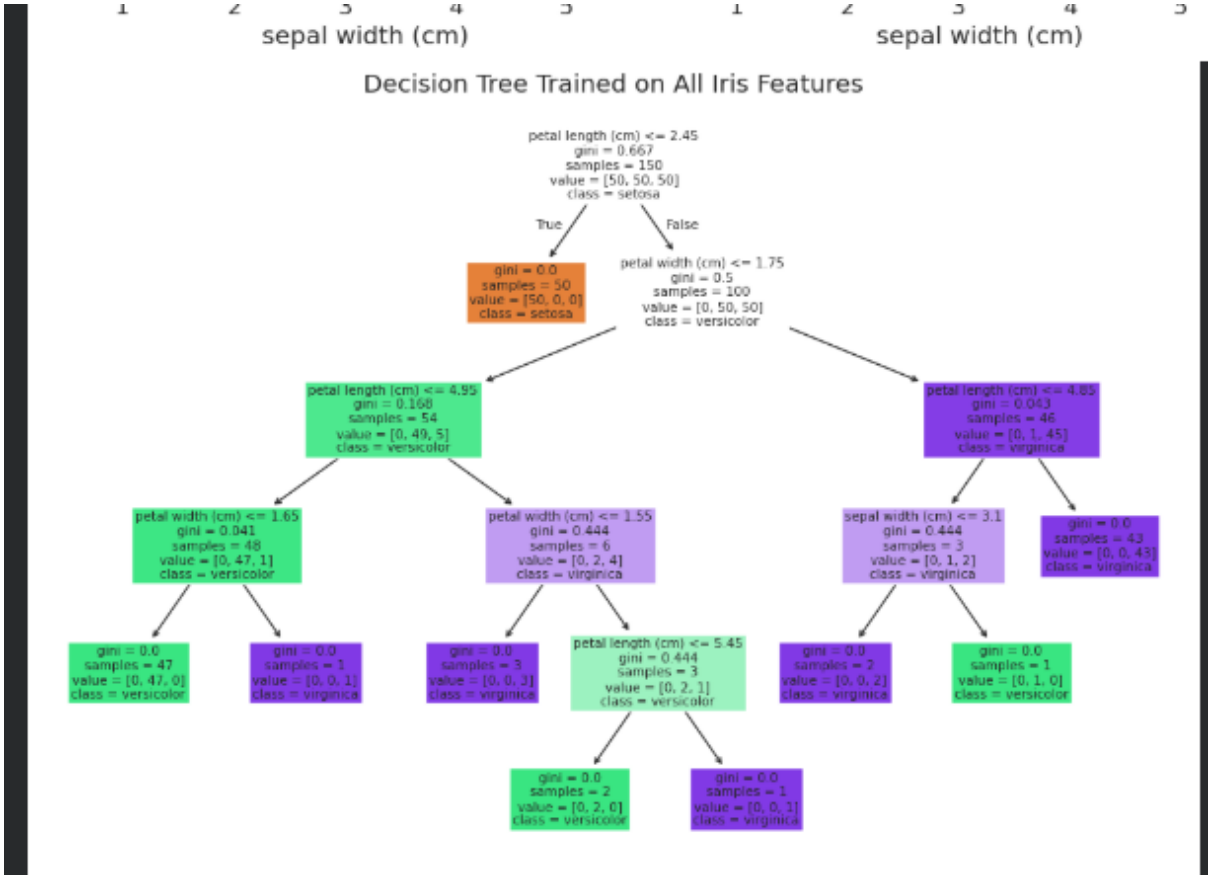
clf_full = DecisionTreeClassifier().fit(X, y)

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()




1 2 3 4 5
sepal width (cm)
1 2 3 4 5
sepal 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
True
False
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, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor