A PYTHON PROGRAM TO IMPLEMENT DECISION TREE

Aim:

To implement a decision tree using a python program for the given dataset and plot the trained decision tree.

Algorithm:

Step 1: Import the Iris Dataset

● Import `load_iris` from `sklearn.datasets`.

Step 2: Import Necessary Libraries

● Import numpy as np.

● Import matplotlib.pyplot as plt.

● Import `DecisionTreeClassifier` from `sklearn.tree`.

Step 3: Declare and Initialize Parameters

● Declare and initialize `n_classes = 3`.

● Declare and initialize `plot_colors = "ryb"`.

● Declare and initialize `plot_step = 0.02`.

Step 4: Prepare Data for Model Training

● Load the iris dataset using `load_iris()`.

● Assign the dataset's data to variable `X`.

● Assign the dataset's target to variable `Y`.

Step 5: Train the Model

● Create an instance of `DecisionTreeClassifier`.

● Fit the classifier using `clf.fit(X, Y)`.

Step 6: Initialize Pair Index and Plot Graph

● Loop through each pair of features using `for pairidx, pair in enumerate(combinations (range(X.shape[1]), 2)):`

● Inside the loop, assign `X` with the selected pair of features (e.g., `X = iris.data[:, pair]`).

● Assign `Y` with the target list (e.g., `Y = iris.target`).

Step 7: Assign Axis Limits

1. Inside the loop, assign `x_min` with the minimum value of the selected feature minus 1 (e.g., `x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1`).

2. Assign `x_max` with the maximum value of the selected feature plus 1.

3. Assign `y_min` with the minimum value of the second selected feature minus 1 (e.g., `y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1`).

4. Assign `y_max` with the maximum value of the second selected feature plus 1.

Step 8: Create Meshgrid

1. Use `np.meshgrid` to create a grid of values from `x_min` to `x_max` and `y_min` to `y_max` with steps of `plot_step`.

2. Assign the results to variables `xx` and `yy`.

Step 9: Plot Graph with Tight Layout

1. Use `plt.tight_layout()` to adjust the layout of the plots.

2. Set `h_pad=0.5`, `w_pad=0.5`, and `pad=2.5`.

Step 10: Predict and Reshape

1. Use the classifier to predict on the meshgrid (e.g., `Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])`).

2. Reshape `Z` to the shape of `xx`.

Step 11: Plot Decision Boundary

1. Use `plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)` to plot the decision boundary with the "RdYlBu" color scheme.

Step 12: Plot Feature Pairs

1. Inside the loop, label the x-axis and y-axis with the feature names (e.g., `plt.xlabel(iris.feature_names[pair[0]])` and `plt.ylabel(iris.feature_names[pair[1]])`).

Step 13: Plot Training Points

1. Use `plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.RdYlBu, edgecolor='k', s=15)` to plot the training points with the "RdYlBu" color scheme, black edge color, and size 15.

Step 14: Plot Final Decision Tree

1. Set the title of the plot to "Decision tree trained on all the iris features" (e.g., `plt.title("Decision tree trained on all the iris features")`).

2. Display the plot using `plt.show()`.