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.

Step 3: Declare and initialize parameters.

* Declare and initialize 'n_classes = 3'.
* 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 target to variable 'y'.

Step 6: 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 pair_idx, pair in enumerate (combinations (range (x.shape [1]), 2)):

Step 7: Assign Axis Limits

1. Inside the loop, assig '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. Assig~~

2. Assign 'y_max' with the maximum value of the second feature plus 1.

Step 8: Create meshgrid

1. Assign the result to variables 'xx' and 'yy'.

Step 9: plot Graph with tight Layout

1. Reshap 'Z' to the shape of 'xx'.

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()])`).

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 with the decision boundary with the "RdYlBu".

Step 13: Plot Final Decision Tree

1. Display ~~title~~ the plot using `plt.show()`.

Program:

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

iris = load_iris()
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02
```

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

for pairidx, pair in enumerate([[0,1],[0,2],[0,3],
                                [1,2],[1,3],[2,3]]):

    X = iris.data[:, pair]
    y = iris.target

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

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

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

plt.suptitle("Decision surface of decision trees
              trained on pairs of features")

plt.legend(loc="lower right", borderpad=0,
           handletextpad=0)

clf = DecisionTreeClassifier()

plt.show()
```

Result:

Thus the python program to Implement Decision tree for the given dataset has been verified and analyzed successfully.