Ex. No.: 7
Date: 4/10/24

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

1. Import `load_iris` from `sklearn.datasets`.

Step 2: Import Necessary Libraries

1. Import numpy as np.
2. Import matplotlib.pyplot as plt.
3. Import `DecisionTreeClassifier` from `sklearn.tree`.

Step 3: Declare and Initialize Parameters

1. Declare and initialize `n_classes = 3`.
2. Declare and initialize `plot_colors = "ryb"`.
3. Declare and initialize `plot_step = 0.02`.

Step 4: Prepare Data for Model Training

1. Load the iris dataset using `load_iris()`.
2. Assign the dataset's data to variable `X`.
3. Assign the dataset's target to variable `Y`.

Step 5: Train the Model

1. Create an instance of `DecisionTreeClassifier`.
2. Fit the classifier using `clf.fit(X, Y)`.

Step 6: Initialize Pair Index and Plot Graph

1. Loop through each pair of features using `for pairidx, pair in enumerate(combinations (range(X.shape[1]), 2)):`
2. Inside the loop, assign `X` with the selected pair of features (e.g., `X = iris.data[:, pair]`).

3. 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

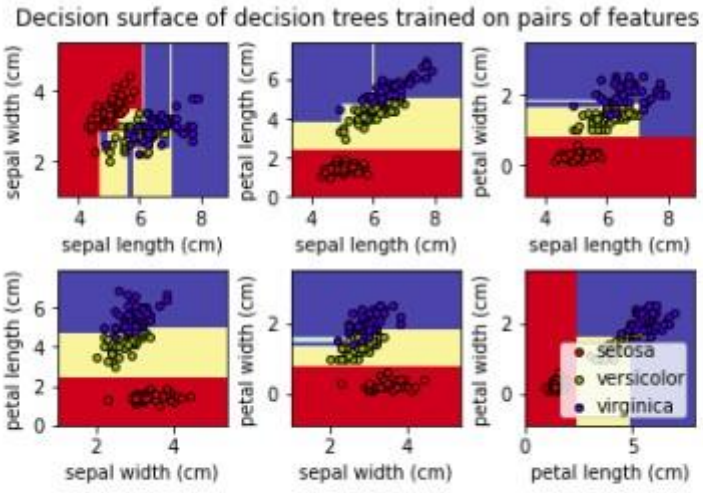
1. Set the title of the plot to "Decision tree trained on all the iris features" (e.g., `plt.title("Decision tree trained on all the iris features")`).
2. Display the plot using `plt.show()`.

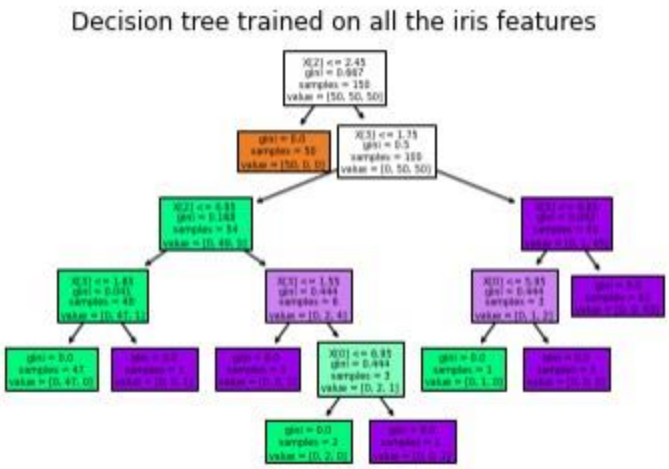
PROGRAM:

```
from sklearn.datasets import load_iris iris =
load_iris() import numpy as np import
matplotlib.pyplot as plt from sklearn.tree
import DecisionTreeClassifier
# Parameters n_classes = 3 plot_colors = "ryb" plot_step = 0.02 for
pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]):
# We only take the two corresponding features X =
iris.data[:, pair] y = iris.target # Train clf =
DecisionTreeClassifier().fit(X, y) # Plot the
decision boundary plt.subplot(2, 3, pairidx + 1)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() +
1 y_min, y_max = X[:, 1].min() - 1, X[:, 1].max()
+ 1 xx, yy = np.meshgrid( np.arange(x_min,
x_max, plot_step), np.arange(y_min, y_max,
plot_step)
)
plt.tight_layout(h_pad=0.5, w_pad=0.5, pad=2.5)
```

```
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.xlabel(iris.feature_names[pair[0]])
plt.ylabel(iris.feature_names[pair[1]])
# 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], cmap=plt.cm.RdYlBu, edgecolor="black",
s=15) plt.suptitle("Decision surface of decision trees trained on pairs of
features") plt.legend(loc="lower right", borderpad=0, handletextpad=0)
plt.axis("tight") from sklearn.tree import plot_tree

plt.figure()                    clf                    =
DecisionTreeClassifier().fit(iris.data,iris.target)
plot_tree(clf, filled=True)
plt.title("Decision tree trained on all the iris features") plt.show()
```





RESULT:

Thus the python program to implement Decision Tree for the given dataset has been successfully implemented and the results have been verified and analyzed