APYTHONPROGRAMTOIMPLEMENTDECISIONTREE

Ex.No.: 7

Date of Experiment: 04/10/2024

Date of Submission:

AIM:-

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

ALGORITHM:-

Step1: Import the iris dataset from the "sklearn.datasets" library.

Step2: Import all the other necessary libraries(numpy as np, matplotlib.pyplot as plt and DecisionTreeClassifier from sklearn.tree).

Step3: Declare and initialize the parameters(n_classes = 3,plot_colors = "ryb" and plot_step = 0.02)

Step4: Loop through the list of features and assign "X" with all the pairs in the list and "Y" with the target list.

Step5: Train the model and assign it to a variable name "clt".

Step6: Use the "pairidx" variable to plot the graph.

Step7: Assign "x_min", "x_max", "y_min" and "y_max" variables with the respective values from the list.

Step8: Assign the variables "X" and "Y" the values obtained by using the "meshgrid()" function on arranged x min,x max and y min,y max.

Step9: Plot the graph using the "tight_layout" function and the following parameters(h_pad=0.5, w_pad=0.5, pad=2.5).

Al23331 231501077

Step10: Assign the prediction using the variables "xx" and "yy" and then reshape Z to the shape of "xx".

Step11: Plot the graphs using "xx", "yy" and "Z" as the parameters and with the "RdYlBu"(red,yellow and blue) color scheme.

Step12: Plot all the x label and y label feature pairs.

Step13: Plot all the training points with "RdYlBu" color scheme, black color to represent the points and with size equal to 15.

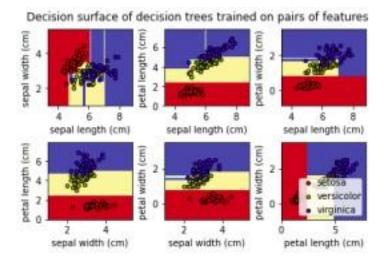
Step14: Plot the final decision tree with the title "Decision tree trained on all the iris

features". IMPLEMENTATION:-

```
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)
AI23331
```

```
x \min_{x} \max = X[:, 0].\min() - 1, X[:, 0].\max() + 1
  y \min_{x \in X} y \max_{x \in X} = 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")
```

Al23331 231501077



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.

Al23331 231501077