LAB EXERCISE – 6

Decision Tree Classifier – CART

1. Aim of the Experiment:

Implement and demonstrate the working of the decision tree based CART algorithm using a sample data set. Build the decision tree and use this model to classify a test sample.

Listing 1:

Sample Dataset Used: **Table 6.3**

S.No.	CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Offer
1.	≥9	Yes	Very good	Good	Yes
2.	≥8	No	Good	Moderate	Yes
3.	≥9	No	Average	Poor	No
4.	<8	No	Average	Good	No
5.	≥8	Yes	Good	Moderate	Yes
6.	≥9	Yes	Good	Moderate	Yes
7.	<8	Yes	Good	Poor	No
8.	≥9	No	Very good	Good	Yes
9.	≥8	Yes	Good	Good	Yes
10.	≥8	Yes	Average	Good	Yes

3. Python Program with Explanation:

1. Import the library ‘pandas’ to create a Data frame which is a two-dimensional data structure.

```
import pandas
```

2. Import DecisionTreeClassifier from sklearn.tree.

```
from sklearn.tree import DecisionTreeClassifier
```

3. Import LabelEncoder to normalize labels.

```
from sklearn.preprocessing import LabelEncoder
```

4. Import train_test_split function.

```
from sklearn.model_selection import train_test_split
```

5. Create a list ‘data’ with the sample dataset.

```
data = {'CGPA':['g9','g8','g9','l8','g8','g9','l8','g9','g8','g8'],
        'Inter':['Y','N','N','N','Y','Y','Y','N','Y','Y'],
        'PK':['+++','+','==','==','+','+','+','+++','+','=='],
        'CS':['G','M','P','G','M','M','P','G','G','G'],
        'Job':['Y','Y','N','N','Y','Y','N','Y','Y','Y']}
```

6. Create pandas dataframe “table” using the structure DataFrame with the given dataset ‘data'.

```
table=pandas.DataFrame(data,
columns=["CGPA","Inter","PK","CS","Job"])
```

7. Use a value ["CGPA"]=="g9" in the table to select matching row and count the number of columns.

```
table.where(table["CGPA"]=="g9").count()
```

8. Use LabelEncoder() to encode target labels with value between 0 and no_of_classes-1.

```
encoder=LabelEncoder()
```

9. Then transform non-numerical labels to numerical labels.

```
for i in table:
    table[i]=encoder.fit_transform(table[i])
```

10. Use iloc property to select by position.

Select the columns until (excluding) the fourth column.

```
X=table.iloc[:,0:4].values
```

Select the fourth column

```
y=table.iloc[:,4].values
```

11. Split the dataset into training dataset and test dataset by using the function train_test_split().
This function has several parameters, but we pass 3 parameters, data, test_size and random_state.
X, y is the dataset we are selecting to use.
test_size. to specify the size of the testing dataset. It will be set to 0.25 if the training size is set to default.
random_state to perform a random split.
X_train is the features of the training subset
y_train is the class labels of the target feature of the training subset
X_test holds the features of the testing subset
y_test holds the class labels of the target feature of the testing subset

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)
```

12. Use DecisionTreeClassifier model. It allows some attributes like criterion, splitter, max_features, max_depth, max_leaf_nodes etc., we will use the attribute criterion which takes a value 'gini' to implement a classifier using CART

```
model = DecisionTreeClassifier(criterion='gini')
```

DecisionTreeClassifier takes as input two arrays: an array X_train, holding the training instances, and an array y_train holding the class labels for the training instances.

13. Then train the classifier using the function fit().

```
model.fit(X_train,y_train)
```

14. After training, the fitted model can be used to predict a new instance.

# The non-numerical equivalent of the new instance [1,0,0,1] given is ['g9', 'Y', '***', 'M']

```
print([1,0,0,1])
```

```
if model.predict([[1,0,0,1]])==1:
    print("Got JOB")
else:
    print("Didnt get JOB")
```

# The non-numerical equivalent of the new instance [2,0,2,0] given is [‘l8’, ‘Y’, ‘==’, ‘G’]

```
print([2,0,2,0])
if model.predict([[2,0,2,0]])==1:
    print("Got JOB")
else:
    print("Didnt get JOB")
```

Complete Program:

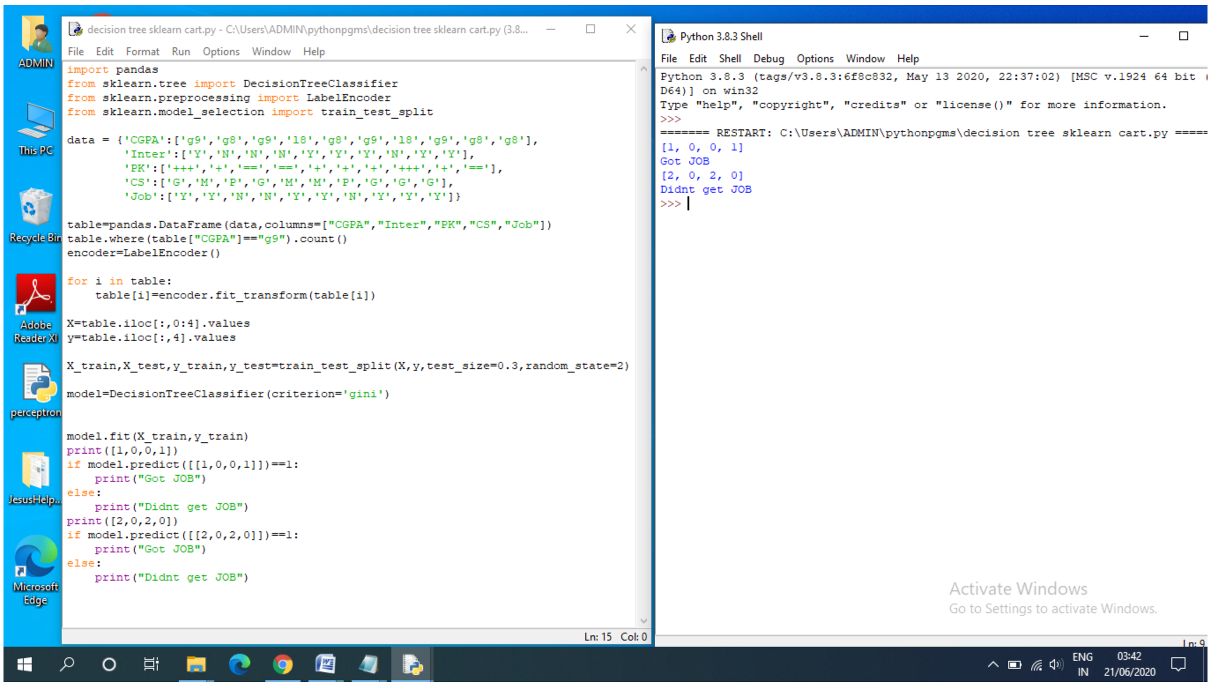
```
import pandas

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

data = {'CGPA':['g9','g8','g9','l8','g8','g9','l8','g9','g8','g8'],

        'Inter':['Y','N','N','N','Y','Y','Y','N','Y','Y'],

        'PK':['+++','+','==','==','+','+','+','+++','+','=='],

        'CS':['G','M','P','G','M','M','P','G','G','G'],

        'Job':['Y','Y','N','N','Y','Y','N','Y','Y','Y']}


table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])

table.where(table["CGPA"]=="g9").count()

encoder=LabelEncoder()


for i in table:

    table[i]=encoder.fit_transform(table[i])
```

```
X=table.iloc[:,0:4].values

y=table.iloc[:,4].values

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=2)

model=DecisionTreeClassifier(criterion='gini')

model.fit(X_train,y_train)

print([1,0,0,1])

if model.predict([[1,0,0,1]])==1:

    print("Got JOB")

else:

    print("Didnt get JOB")

print([2,0,2,0])

if model.predict([[2,0,2,0]])==1:

    print("Got JOB")

else:

    print("Didnt get JOB")
```

Output:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit
(AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

======= RESTART: C:\Users\ADMIN\pythonpgms\decision tree sklearn cart.py
=======

[1, 0, 0, 1]

Got JOB

[2, 0, 2, 0]

Didnt get JOB

>>>
```

Screenshot of the Output:



```
import pandas
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

data = {'CGPA':['g9','g8','g9','l8','g8','g9','l8','g9','g8','g8'],
        'Inter':['Y','N','N','N','Y','Y','Y','N','Y','Y'],
        'PK':['+++','+','==','==','+','+','+','+++','+','=='],
        'CS':['G','M','P','G','M','M','P','G','G','G'],
        'Job':['Y','Y','N','N','Y','Y','N','Y','Y','Y']}

table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
table.where(table["CGPA"]=="g9").count()
encoder=LabelEncoder()

for i in table:
    table[i]=encoder.fit_transform(table[i])

X=table.iloc[:,0:4].values
y=table.iloc[:,4].values

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=2)

model=DecisionTreeClassifier(criterion='gini')


model.fit(X_train,y_train)
print([1,0,0,1])
if model.predict([[1,0,0,1]])==1:
    print("Got JOB")
else:
    print("Didnt get JOB")
print([2,0,2,0])
if model.predict([[2,0,2,0]])==1:
    print("Got JOB")
else:
    print("Didnt get JOB")
```

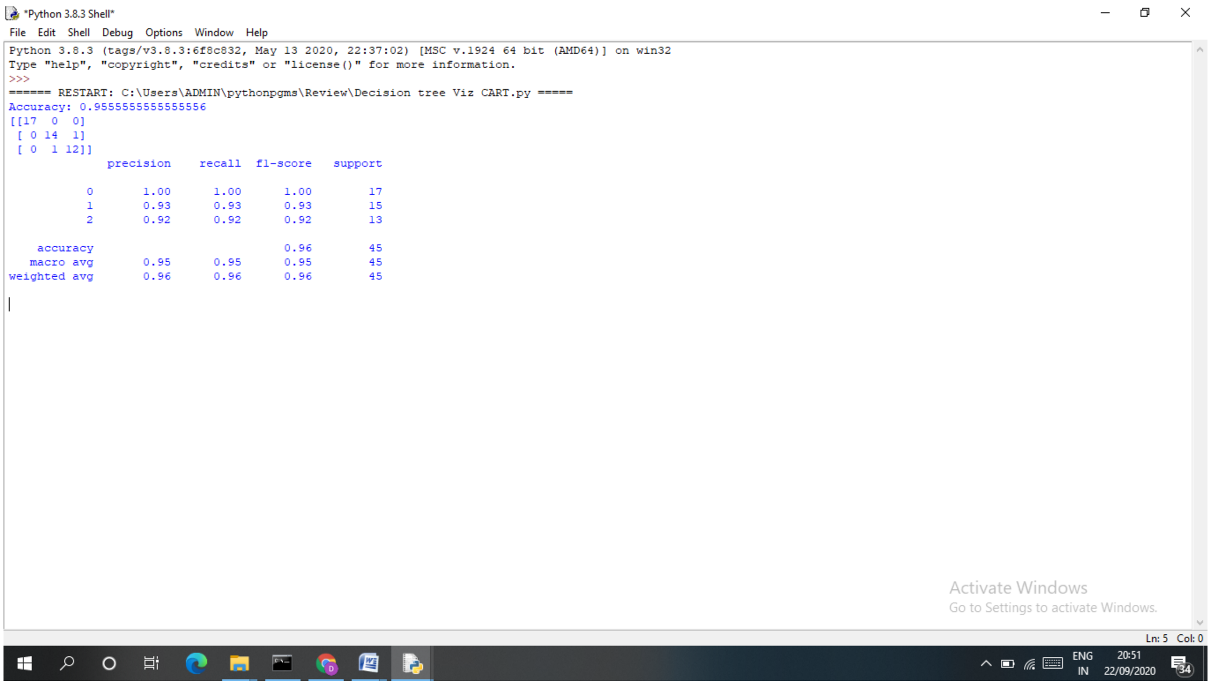
```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:\Users\ADMIN\pythonpgms\decision tree sklearn cart.py =====
[1, 0, 0, 1]
Got JOB
[2, 0, 2, 0]
Didnt get JOB
>>> 
```

Listing 2:

Program Code:

```
from matplotlib import pyplot as plt

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn import metrics

from sklearn.metrics import classification_report, confusion_matrix



# Load the Iris dataset

iris = datasets.load_iris()

X = iris.data

y = iris.target
```

```
# Split the data matrix into train and test dataset
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=2)


# Train the model using DecisionTreeClassifier CART
clf = DecisionTreeClassifier(criterion='gini',random_state=1234)

model = clf.fit(X_train, y_train)

y_pred = model.predict(X_test)


# Evaluating Classification Model Accuracy

print("Accuracy:",metrics.accuracy_score(y_test, y_pred)) # classification rate of
100%,good accuracy.


#Print Confusion Matrix

print(confusion_matrix(y_test, y_pred))


#Print Classification Report and plot the tree graph

print(classification_report(y_test, y_pred))

fig = plt.figure(figsize=(10,8))

_ = tree.plot_tree(clf,

                   feature_names=iris.feature_names,

                   class_names=iris.target_names,

                   filled=True)

plt.show()
```



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
====== RESTART: C:\Users\ADMIN\pythonpgms\Review\Decision tree Viz CART.py =====
Accuracy: 0.9555555555555556
[[17  0  0]
 [ 0 14  1]
 [ 0  1 12]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.93      0.93      0.93        15
           2       0.92      0.92      0.92        13

    accuracy                           0.96        45
   macro avg       0.95      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45

```

Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Activate Windows
Go to Settings to activate Windows.
Ln: 5 Col: 0
ENG IN 20:51 22/09/2020

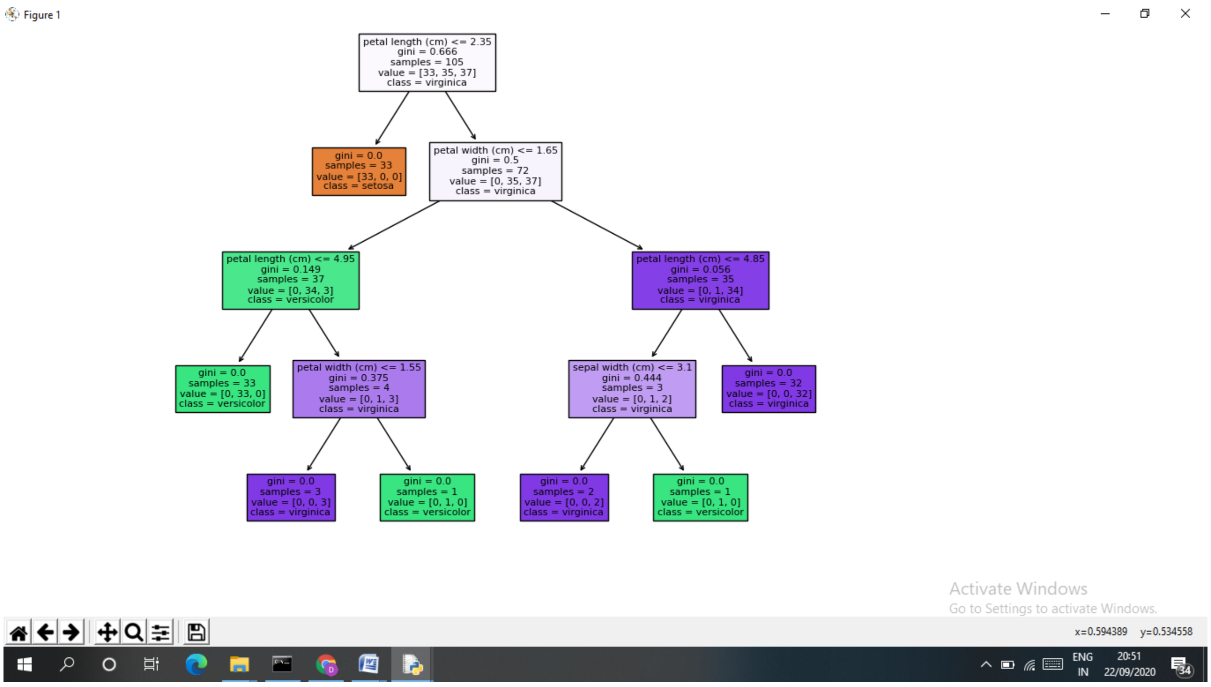


Figure 1
petal length (cm) <= 2.35
gini = 0.666
samples = 105
value = [33, 35, 37]
class = virginica
gini = 0.0
samples = 33
value = [33, 0, 0]
class = setosa
petal width (cm) <= 1.65
gini = 0.5
samples = 72
value = [0, 35, 37]
class = virginica
petal length (cm) <= 4.95
gini = 0.149
samples = 37
value = [0, 34, 3]
class = versicolor
petal length (cm) <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
class = virginica
gini = 0.0
samples = 33
value = [0, 33, 0]
class = versicolor
petal width (cm) <= 1.55
gini = 0.375
samples = 4
value = [0, 1, 3]
class = virginica
sepal width (cm) <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
Activate Windows
Go to Settings to activate Windows.
x=0.594389 y=0.534558
ENG IN 20:51 22/09/2020