PROGRAM – 7

Program to implement a decision tree and display it

#PROGRAM To Implement a Decision TREE AND to DISPLAY IT

print('PROGRAM To Implement a Decision Tree AND to DISPLAY IT')

from sklearn import datasets

import pandas as pd

iris=datasets.load\_iris()

# df will fold dataset as a table

df=pd.DataFrame(

iris.data,

columns=iris.feature\_names

)

#labels are assigned to df[target] table or array

df['target']=pd.Series(

iris.target

)

from sklearn.model\_selection import train\_test\_split

# Train Test Split Ratio

df\_train,df\_test=train\_test\_split(df,test\_size=0.3)

df['target\_names']=df['target'].apply(lambda y:iris.target\_names[y])

print('Number of Training samples')

print(df\_train.shape[0])

print('Number of Testing samples')

print(df\_test.shape[0])

#Importing Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

clf=DecisionTreeClassifier()

x\_train=df\_train[iris.feature\_names]

x\_test=df\_test[iris.feature\_names]

y\_train=df\_train['target']

y\_test=df\_test['target']

#Training Decision Tree Classifier

clf.fit(x\_train,y\_train)

#Testing the data

y\_test\_pred=clf.predict(x\_test)

print('Class of Testing Samples')

print(y\_test\_pred)

#To display the decision tree in command shell

from sklearn.tree import export\_text

from sklearn import tree

from matplotlib import pyplot as plt

text\_representation = tree.export\_text(clf)

print(text\_representation)

with open("decistion\_tree.log", "w") as fout:

  fout.write(text\_representation)

fig = plt.figure(figsize=(25,20))

\_ = tree.plot\_tree(clf,

feature\_names=iris.feature\_names,

class\_names=iris.target\_names,

filled=True)

fig.savefig("decistion\_tree.png")

OUTPUT:

PROGRAM To Implement a Decision Tree AND to DISPLAY IT

Number of Training samples

105

Number of Testing samples

45

Class of Testing Samples

[0 1 0 1 1 1 2 0 2 1 2 0 0 2 2 0 0 1 0 2 1 0 1 2 1 2 0 0 0 1 1 1 2 0 1 0 0

2 1 1 1 0 0 0 1]

|--- feature\_2 <= 2.60

| |--- class: 0

|--- feature\_2 > 2.60

| |--- feature\_3 <= 1.75

| | |--- feature\_2 <= 4.95

| | | |--- class: 1

| | |--- feature\_2 > 4.95

| | | |--- feature\_3 <= 1.55

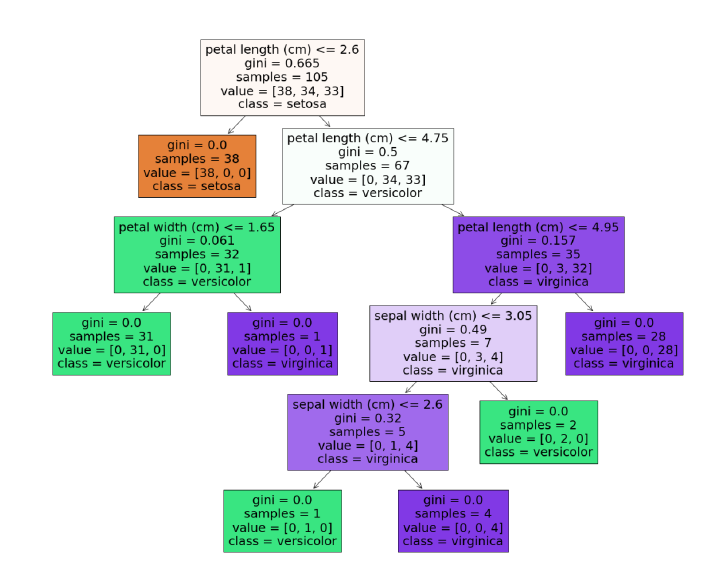
| | | | |--- class: 2

| | | |--- feature\_3 > 1.55

| | | | |--- class: 1

| |--- feature\_3 > 1.75

| | |--- class: 2



b) Program to calculate accuracy of decision tree

#PROGRAM To Calculate Accuracy of Decision Tree

print('PROGRAM To Calculate Accuracy of Decision Tree')

from sklearn import datasets

import pandas as pd

iris=datasets.load\_iris()

# df will fold dataset as a table

df=pd.DataFrame( iris.data, columns=iris.feature\_names )

#labels are assigned to df[target] table or array

df['target']=pd.Series( iris.target )

from sklearn.model\_selection import train\_test\_split

# Train Test Split Ratio

df\_train,df\_test=train\_test\_split(df,test\_size=0.3)

df['target\_names']=df['target'].apply(lambda y:iris.target\_names[y])

print('Number of Training samples')

print(df\_train.shape[0])

print('Number of Testing samples')

print(df\_test.shape[0])

#Importing Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

clf=DecisionTreeClassifier()

x\_train=df\_train[iris.feature\_names]

x\_test=df\_test[iris.feature\_names]

y\_train=df\_train['target']

y\_test=df\_test['target']

#Training Decision Tree Classifier

clf.fit(x\_train,y\_train)

#Testing the data

y\_test\_pred=clf.predict(x\_test)

print('Class of Testing Samples')

print(y\_test\_pred)

from sklearn.metrics import accuracy\_score

x=accuracy\_score(y\_test,y\_test\_pred)

print('Accuracy')

print(x)

OUTPUT:

PROGRAM To Calculate Accuracy of Decision Tree

Number of Training samples

105

Number of Testing samples

45

Class of Testing Samples

[1 0 0 0 1 2 1 0 2 2 0 1 2 2 1 1 0 1 2 2 1 2 2 0 0 2 0 1 1 2 0 2 2 1 0 1 1

2 1 0 1 1 2 1 0]

Accuracy

0.9111111111111111

(c) Program to implement a Decision Tree Regression

#Program to implement Decision Tree Regression

import numpy as np

n=200

#200 samples

height\_pop1\_f=np.random.normal(loc=155,scale=10,size=n)

height\_pop1\_m=np.random.normal(loc=175,scale=5,size=n)

height\_pop2\_f=np.random.normal(loc=165,scale=10,size=n)

height\_pop2\_m=np.random.normal(loc=185,scale=5,size=n)

height\_f=np.concatenate([height\_pop1\_f,height\_pop2\_f])

height\_m=np.concatenate([height\_pop1\_m,height\_pop2\_m])

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.tree import export\_text

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import train\_test\_split

df\_height=pd.DataFrame( { 'Gender':[1 for i in range(height\_f.size)]+ [2 for i in range(height\_m.size)], 'Height':np.concatenate((height\_f,height\_m))} )

# to calculate mean and median of height

df\_height.groupby('Gender')[['Height']].agg([np.mean,np.median]).round(1)

df\_train,df\_test=train\_test\_split(df\_height,test\_size=0.3)

x\_train,x\_test=df\_train[['Gender']],df\_test[['Gender']]

y\_train,y\_test=df\_train['Height'],df\_test['Height']

print('Training Samples')

print(df\_train)

print('Testing Samples')

print(df\_test)

for criterion in['squared\_error','absolute\_error']: rgrsr=DecisionTreeRegressor(criterion=criterion)

rgrsr.fit(x\_train,y\_train)

print(f'criterion={criterion}:\n')

print(export\_text(rgrsr,feature\_names=['Gender'],spacing=3,decimals=1))

print('Program Executed successfully')

OUTPUT:

Training Samples

Gender Height

62 1 138.355342

713 2 180.390215

676 2 182.020545

68 1 142.013297

719 2 188.114743

.. ... ...

644 2 179.164963

39 1 160.125862

759 2 189.485930

359 1 184.260191

484 2 170.556742

[560 rows x 2 columns]

Testing Samples

Gender Height

394 1 143.860300

409 2 166.027993

22 1 147.815705

651 2 188.313549

509 2 170.091409

.. ... ...

621 2 187.139578

119 1 153.600674

162 1 157.336898

507 2 177.961577

212 1 182.452209

[240 rows x 2 columns]

criterion=absolute\_error:

|--- Gender <= 1.5

| |--- value: [159.8]

|--- Gender > 1.5

| |--- value: [180.5]

Program Executed successfully

Assignment:

Implement a Decision Tree algorithm from scratch, without relying on high-level libraries like **scikit-learn**.

Objective:

Implement a Decision Tree Classifier in Python from scratch. Your classifier should handle categorical and numerical data and include functionality to prevent overfitting through tree pruning.