ML LAB EXERCISE – WEEK - 7

Decision Tree Classifier – ID3

1. Aim of the Experiment:

Implement and demonstrate the working of the decision tree based ID3 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 1**

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. Import metrics module to implement functions to measure classification performance.

```
from sklearn import metrics
```

6. Import classification_report and confusion_matrix from sklearn.metrics to measure the quality of predictions.

```
from sklearn.metrics import classification_report, confusion_matrix
```

7. 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']}
```

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

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

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

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

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

```
encoder=LabelEncoder()
```

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

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

12. Use iloc property to select by position.

Select the columns until (excluding) the fifth column.

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

Select the fifth column

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

13. 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)
```

14. 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 ‘entropy’ to implement a classifier using ID3. The attribute value for max_depth is given as 3 to pre prune the tree.

```
model=DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

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

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

14. To make predictions, the predict method of the DecisionTreeClassifier class is used.

```
y_pred = model.predict(X_test)
```

15. Use sklearn.metrics.accuracy_score() to compute the accuracy by comparing actual test set values and predicted values.

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

16. Generate classification report & confusion matrix to measure the quality of predictions.

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

17. 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
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix

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= 0.20,random_state=2)
model=DecisionTreeClassifier(criterion='entropy', max_depth=3)
model = model.fit(X_train,y_train)

y_pred = model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

print([1,0,0,1])
if model.predict([[1,0,0,1]])==1:
   print("Got JOB")
else:
   print("Didn’t get JOB")
 print([2,0,2,0])
if model.predict([[2,0,2,0]])==1:
   print("Got JOB")
else:
   print("Didn’t get JOB")
```

Output:

```
======== RESTART: decision tree sklearn id3.py =======
Accuracy: 1.0
[[1]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1

[1, 0, 0, 1]
Got JOB
[2, 0, 2, 0]
Didn’t get JOB
>>>
```

Screen Shot of the Output:

```
        '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=1,random_state=2)

model=DecisionTreeClassifier(criterion='entropy', max_depth=3) # ID3 + pre pruni
model = model.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
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

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")
```

```
Accuracy: 1.0
[[1]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1

[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
import graphviz

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Train the model using DecisionTreeClassifier ID3
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3)
model = clf.fit(X, y)

#Visualize the model using tree graph
fig = plt.figure(figsize=(10,8))
_ = tree.plot_tree(clf,
```

```
                feature_names=iris.feature_names,
                class_names=iris.target_names,
                filled=True)
plt.show()
#fig.savefig("decistion_tree.png")
```