Name: Mareena Fernandes

TE IT	Roll number : 8669
Expt number : 3	Date of implementation: 10/05/2021
Aim : To implement classification algorithm	
Programming language used : Python	
Related Course outcome : CO2 Upon completion of this course students will be able to Implement the appropriate data mining methods like classification, clustering or association mining on large data sets	

Theory : Data Classification is a two step process. In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step or training phase, where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels. A tuple, X , is represented by an n dimensional attribute vector, $X= (x_1,x_2,.....,x_n)$,depicting n measurements made on the tuple from database attributes, repectively , $A_1, A_2,....,A_n$. Each tuple, X, is assumed to belong to a predefined class as determined by another database attribute called the class label attribute. The class label attribute is discrete-valued and unordered. It is categorical in that each value serves as a category or class. The individual tuples making up the training sets are referred to as training tuples and are selected from the database under analysis. Because the class label of each training tuple is provided, this step is also known as supervised learning. (i.e., the learning of the classifier is "supervised" in that it is told to which class each training tuple belongs). It contrasts with unsupervised learning (or clustering), in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance.

The first step of classification process can also be viewed as the learning of a mapping or function, y=f(x), that can predict the associated class label y of a given tuple X. In this view, we wish to learn a mapping or function that separates the data classes. Typically, this mapping is represented in the form of classification rules, decision tree, or mathematical formulae. In the second step, the model is used for classification. First the predictive accuracy of the classifier is estimated. The test set made up of test tuples and their associated class labels are used to measure the accuracy of the classifier. These tuples are randomly selected from the general data set. They are independent of training tuples, meaning they are not used to construct the classifier. The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classifed by the classifier. The associated class label of each test tuple is compared with the learned classifier's class prediction for that tuple.

Code:

```
import pandas as pd
import numpy as np
import math
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
df = sns.load_dataset('iris')
print(df)
target_col = 'species'
target = df[target_col]
columns = [col for col in df.columns]
columns.pop()
le = LabelEncoder()
target = le.fit_transform(target)
print(target)
df[target_col] = pd.Series(target)
X_train, X_test, y_train, y_test = train_test_split(df, target, test_size = 0.2, random_state = 43)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
unique_counts = np.bincount(target)
probabilities = unique_counts / len(target)
sum_prob = 0
for p in probabilities:
 sum_prob += p * math.log2(p)
root_entropy = - sum_prob
print(root_entropy)
def calcEntropy(col):
 counts = np.bincount(col)
 probabilities = counts / len(col)
 entropy = 0
 for p in probabilities:
 if p > 0:
 entropy += p * math.log2(p)
 if entropy == 0:
 return entropy
 return -entropy
def calcInformationGain(df, col, cur_entropy):
 med = df[col].mean()
 left = df[df[col] <= med]
 right = df[df[col] > med]
 left_prob = left.shape[0] / df.shape[0]
 right_prob = right.shape[0] / df.shape[0]
```

```
 left_entropy = calcEntropy(left['species'])
 right_entropy = calcEntropy(right['species'])
 return (cur_entropy - (left_prob * left_entropy + right_prob * right_entropy)), med, left, right
MAX_NODES = 100
num_nodes = 1
dtree = []
# Recursively build tree by splitting dataframe into 2 on each call
def buildDecisionTree(df, parent_index, branch):
 global num_nodes
 entropy = calcEntropy(df[target_col])
 highest_gain_col = {'gain': 0, 'col': None, 'mid': None, 'left': None, 'right': None}  for col in
columns:
 gain, mid, left, right = calcInformationGain(df, col, entropy)
 if gain >= highest_gain_col['gain']:
 highest_gain_col['gain'] = gain
 highest_gain_col['col'] = col
 highest_gain_col['mid'] = mid
 highest_gain_col['left'] = left
 highest_gain_col['right'] = right

 node = {}
 node['left'] = None
 node['right'] = None
 node['col'] = highest_gain_col['col']
 node['mid'] = round(highest_gain_col['mid'], 2)
 node['class'] = df[target_col].value_counts().idxmax()
 dtree.append(node)
 num_nodes += 1
 if parent_index != -1:
 dtree[parent_index][branch] = len(dtree) - 1
 if highest_gain_col['gain'] >= entropy:
 return
 else:
 new_index = len(dtree) - 1
 buildDecisionTree(highest_gain_col['left'], new_index, 'left')  try:
 temp = highest_gain_col['right'].shape[0]
 buildDecisionTree(highest_gain_col['right'], new_index, 'right')  except:
 print("Error")
 pass

buildDecisionTree(X_train, -1, None)
print(dtree)
y_pred = []
for ind in X_test.index:
```

```
 index = 0
 while True:
 col = dtree[index]['col']
 mid = dtree[index]['mid']
 if X_test[col][ind] <= mid:
 if dtree[index]['left'] == None:
 y_pred.append(dtree[index]['class'])
 break
 else:
 index = dtree[index]['left']
 else:
 if dtree[index]['right'] == None:
 y_pred.append(dtree[index]['class'])
 break
 else:
 index = dtree[index]['right']
print(y_pred)
print("Classification report - \n",
classification_report(y_test,y_pred))
```

Output:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

Classification report -

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	0.88	0.93	8
2	0.90	1.00	0.95	9
accuracy			0.97	30
macro avg	0.97	0.96	0.96	30
weighted avg	0.97	0.97	0.97	30

Post-lab Questions:

1. What is tree pruning?

Ans:

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

There are two approaches to prune a tree −

- Pre-pruning − The tree is pruned by halting its construction early.
- Post-pruning - This approach removes a sub-tree from a fully grown tree.

2. Explain the difference between classification and prediction.

Ans:

Classification	Prediction
Classification is the method of recognizing to which group; a new process belongs to a background of a training data set containing a new process of observing whose group membership is familiar.	Predication is the method of recognizing the missing or not available numerical data for a new process of observing.
A classifier is built to detect explicit labels.	A predictor will be built that predicts a current valued job or command value.
In classification, authenticity depends on detecting the class label correctly.	In predication, the authenticity depends on how well a given predictor can guess the value of a predicated attribute for new data.
In classification, the sample can be called the classifier.	In predication, the sample can be called the predictor.