Assignment 5(a):

Problem Statement: Use bill_authentication.csv for the prediction using Decision Tree classifier. Split the dataset into training and test dataset in 80:20 ratio. Train the model on training dataset and use the test dataset for the prediction purpose.

Source Code (Decision Tree Classifier):

In [1]:
```
#Importing Modules and Libraries:
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
#Dataset Loading and Preprocessing:
data = pd.read_csv(r"D:\softwares\python\python programs\ML\Datasets\bill_authentication.csv")
data = data.dropna(axis=0, how='any', inplace=False)
data
```

Out[2]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...	...	...	...	...	...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

In [3]:
```
#Selecting [X1, X2...] and Y values for Classifictaion:
X = data.drop('Class', axis=1, inplace=False)
Y = data.Class
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
x_train
```

Out[3]:

	Variance	Skewness	Curtosis	Entropy
529	-1.38850	12.5026	0.69118	-7.548700
243	2.77440	6.8576	-1.06710	0.075416
1309	-4.28870	-7.8633	11.83870	-1.897800
664	5.35860	3.7557	-1.73450	1.078900
745	0.75736	3.0294	2.91640	-0.068117
...	...	...	...	...
1095	1.16400	3.9130	-4.55440	-3.867200
1130	-2.29180	-7.2570	7.95970	0.921100
1294	-7.03640	9.2931	0.16594	-4.539600
860	-3.46050	2.6901	0.16165	-1.022400
1126	-3.35820	-7.2404	11.44190	-0.571130

1097 rows × 4 columns

In [4]:
```
#Feature Selection to discard Non-Contributing Features:
feature_estimator = DecisionTreeClassifier(criterion='gini', max_depth=7, random_state=42)
feature_select = SelectFromModel(feature_estimator, threshold=0.09)
feature_select.fit(x_train, y_train)
feature_masks = feature_select.get_support()
features = [x_train.columns[i] for i in range(len(x_train.columns)) if feature_masks[i]]
x_train = pd.DataFrame(data=feature_select.transform(x_train), columns=features)
x_test = pd.DataFrame(data=feature_select.transform(x_test), columns=features)
```

```
x_train
```

Out[4]:

	Variance	Skewness	Curtosis
0	-1.38850	12.5026	0.69118
1	2.77440	6.8576	-1.06710
2	-4.28870	-7.8633	11.83870
3	5.35860	3.7557	-1.73450
4	0.75736	3.0294	2.91640
...	...	...	...
1092	1.16400	3.9130	-4.55440
1093	-2.29180	-7.2570	7.95970
1094	-7.03640	9.2931	0.16594
1095	-3.46050	2.6901	0.16165
1096	-3.35820	-7.2404	11.44190

1097 rows × 3 columns

In [5]:
```
#Model Training and Training Accuracy:
model = DecisionTreeClassifier(criterion='gini', random_state=42, max_depth=7)
model.fit(x_train, y_train)
score = model.score(x_train, y_train)
f_importance = model.feature_importances_
print("Training Accuracy:",score)
print("Feature Importances:")
for i in range(len(f_importance)):
    print("--> "+features[i]+":",f_importance[i])
```

```
Training Accuracy: 0.9990884229717412
Feature Importances:
--> Variance: 0.6214568582238973
--> Skewness: 0.22850981108285293
--> Curtosis: 0.15003333069324973
```

In [6]:
```
#Model Testing and Testing Accuracy:
y_pred = model.predict(x_test)
score = accuracy_score(y_test, y_pred)
result = pd.DataFrame(data=x_test.values, columns=x_test.columns)
result['class'] = y_test.values
result['class_pred'] = y_pred
print("Testing Accuracy:",score)
print("Classification Result:")
result
```

```
Testing Accuracy: 0.9781818181818182
Classification Result:
```

Out[6]:

	Variance	Skewness	Curtosis	class	class_pred
0	1.569100	6.346500	-0.1828	0	0
1	-0.278020	8.188100	-3.1338	0	0
2	0.051979	7.052100	-2.0541	0	0
3	-1.755900	11.945900	3.0946	0	0
4	2.428700	9.382100	-3.2477	0	0
...	...	...	...	...	...
270	-2.262500	-0.099335	2.8127	1	1
271	1.437800	0.668370	-2.0267	1	1
272	3.462600	-4.449000	3.5427	0	0
273	-0.278000	8.188100	-3.1338	0	0
274	2.783100	10.979600	-3.5570	0	0

275 rows × 5 columns

In [7]:
```
#Generating Confusion Matrix and Classification Report:
report = classification_report(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
cm = pd.DataFrame(cm)
print("Classification Report:",report)
print("\nConfusion Matrix:")
cm
```

```
Classification Report:              precision    recall  f1-score   support

           0       0.96      1.00      0.98       148
           1       1.00      0.95      0.98       127

    accuracy                           0.98       275
   macro avg       0.98      0.98      0.98       275
weighted avg       0.98      0.98      0.98       275


Confusion Matrix:
```

Out[7]:

	0	1
0	148	0
1	6	121

In [8]:
```
#Displaying the Confusion Matrix:
sns.heatmap(cm, annot=True, cmap='Reds', fmt='g')
```

Out[8]: <Axes: >