Name: Abhishek Guleri
Roll No. : 185509
Lab: Data Mining
Assignment No. : 6
Branch: CSE DD

---

1. **Demonstration of Association rule process on dataset PIMA India diabetic using Apriori algorithm**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori

data = pd.read_csv(r'diabetes.csv')

# Dataframe into a list of lists
records = []
for i in range(0, 768):
   records.append([str(data.values[i,j]) for j in range(0, 9)])

association_rules = apriori(records, min_support=0.0045, min_confidence=0.2,
min_lift=3, min_length=2)
association_results = list(association_rules)

print("Total number of rules mined = " , len(association_results))
print(association_results[0])

for item in association_results:

   # first index of the inner list
   # Contains base item and add item
   pair = item[0]
   items = [x for x in pair]
   print("Rule: " + items[0] + " -> " + items[1])

   #second index of the inner list
   print("Support: " + str(item[1]))

   #third index of the list located at 0th
   #of the third index of the inner list

   print("Confidence: " + str(item[2][0][2]))
   print("Lift: " + str(item[2][0][3]))
```

```
    print("====================================")
```

Result

Total number of rules mined = 87
RelationRecord(items=frozenset({'10.0', '115.0'}), support=0.005208333333333333, ordered_statistics=[OrderedStatistic(items_base=frozenset({'115.0'}), items_add=frozenset({'10.0'}), confidence=0.25, lift=6.620689655172415)]

Rule: 10.0 -> 115.0
Support: 0.005208333333333333
Confidence: 0.25
Lift: 6.620689655172415
====================================
Rule: 108.0 -> 24.0
Support: 0.005208333333333333
Confidence: 0.25
Lift: 3.0476190476190474
====================================
Rule: 11.0 -> 80.0
Support: 0.005208333333333333
Confidence: 0.2352941176470588
Lift: 4.015686274509803
====================================
Rule: 4.0 -> 110.0
Support: 0.005208333333333333
Confidence: 0.26666666666666666
Lift: 3.011764705882353
====================================
Rule: 112.0 -> 24.0
...

2. **Write a program to implement the DHC algorithm**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN


X, y = make_blobs(n_samples=500,n_features=2,
centers=4, cluster_std=1,
center_box=(-10.0, 10.0),
```

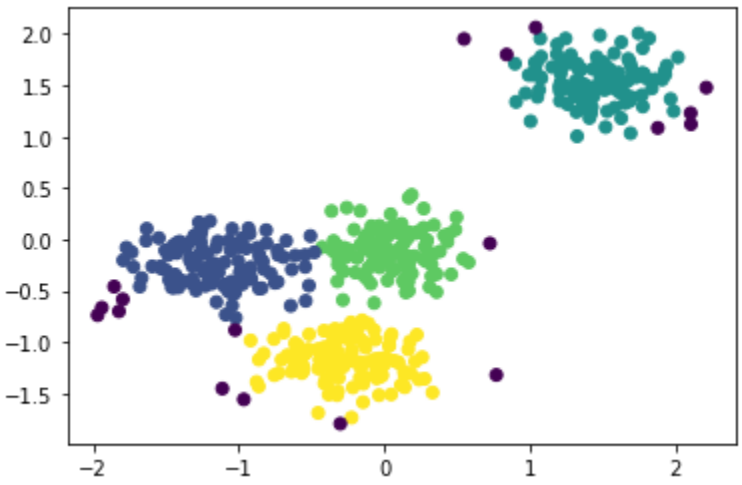
```
shuffle=True, random_state=1)

X = StandardScaler().fit_transform(X)
y_pred = DBSCAN(eps=0.3, min_samples=30).fit_predict(X)

plt.scatter(X[:,0], X[:,1], c=y_pred)
print('Number of clusters: {}'.format(len(set(y_pred[np.where(y_pred !=
-1)]))))
print('Homogeneity: {}'.format(metrics.homogeneity_score(y, y_pred)))
print('Completeness: {}'.format(metrics.completeness_score(y, y_pred)))
```

Result

```
Number of clusters: 4
Homogeneity: 0.9060238108583653
Completeness: 0.8424339764592357
```



3. Write a program to implement the Decision tree.

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn import tree

# Load the iris dataset
iris=load_iris()
print(iris.feature_names)
```

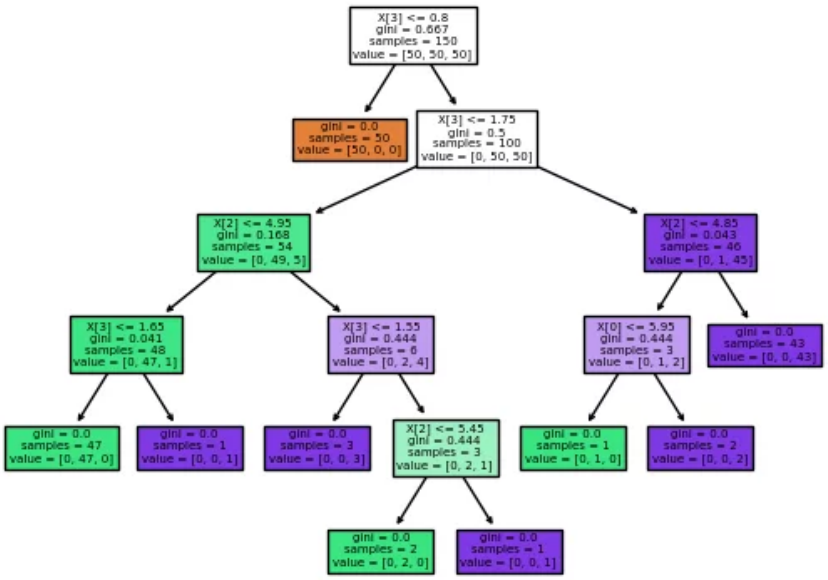
```
#Spilitting the dataset
removed =[0,50,100]
new_target = np.delete(iris.target,removed)
new_data = np.delete(iris.data,removed, axis=0)

#train classifier
clf = tree.DecisionTreeClassifier()
clf=clf.fit(new_data,new_target)
prediction = clf.predict(iris.data[removed])

print("Original Labels",iris.target[removed])
print("Labels Predicted",prediction)

tree.plot_tree(clf, filled=True)
```

Result



4. **Write a program to implement the Naive Bayes classification.**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

```
# importing the dataset
dataset = pd.read_csv('NaiveBayes.csv')

# split the data into inputs and outputs
X = dataset.iloc[:, [0,1]].values
y = dataset.iloc[:, 2].values

# assign test data size 25%
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size= 0.25,
random_state=0)

# importing standard scaler
from sklearn.preprocessing import StandardScaler

# scalling the input data
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)

classifer = GaussianNB()

# training the model
classifer.fit(X_train, y_train)

# testing the model
y_pred = classifer1.predict(X_test)

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

# printing the report
print(classification_report(y_test, y_pred))
```

Result

```
Accuracy: 0.91
              precision    recall  f1-score   support

           0       0.93      0.94      0.93        68
           1       0.87      0.84      0.86        32

    accuracy                           0.91       100
   macro avg       0.90      0.89      0.90       100
weighted avg       0.91      0.91      0.91       100
```