Machine Learning Homework 2_Report

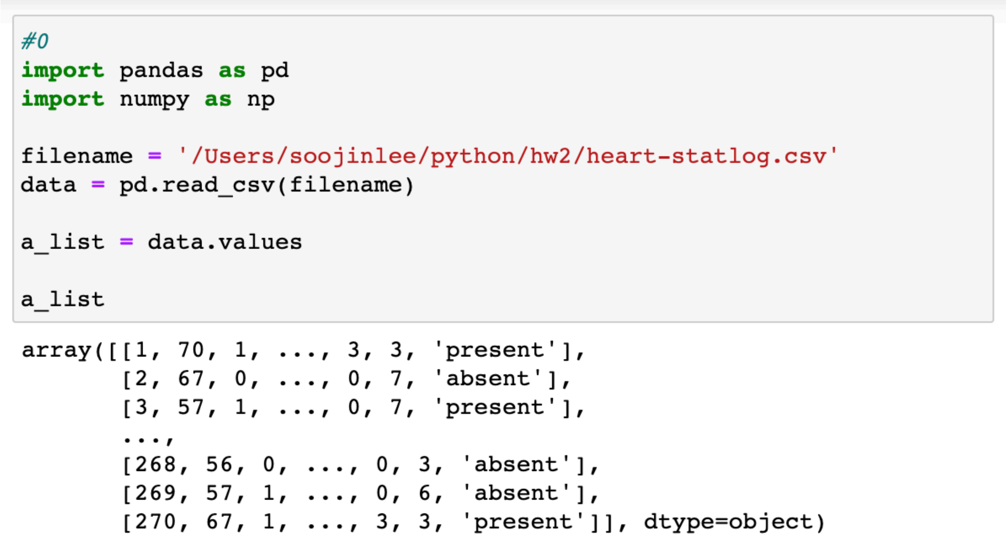
2017112167 이수진

0. Read File

<Program Code>

```
#0
import pandas as pd
import numpy as np

filename = '/Users/soojinlee/python/hw2/heart-statlog.csv'
data = pd.read_csv(filename)

a_list = data.values

a_list
```

<Result>



```
#0
import pandas as pd
import numpy as np

filename = '/Users/soojinlee/python/hw2/heart-statlog.csv'
data = pd.read_csv(filename)

a_list = data.values

a_list
```

```
array([[1, 70, 1, ..., 3, 3, 'present'],
       [2, 67, 0, ..., 0, 7, 'absent'],
       [3, 57, 1, ..., 0, 7, 'present'],
       ...,
       [268, 56, 0, ..., 0, 3, 'absent'],
       [269, 57, 1, ..., 0, 6, 'absent'],
       [270, 67, 1, ..., 3, 3, 'present']], dtype=object)
```

1. Label Encoding

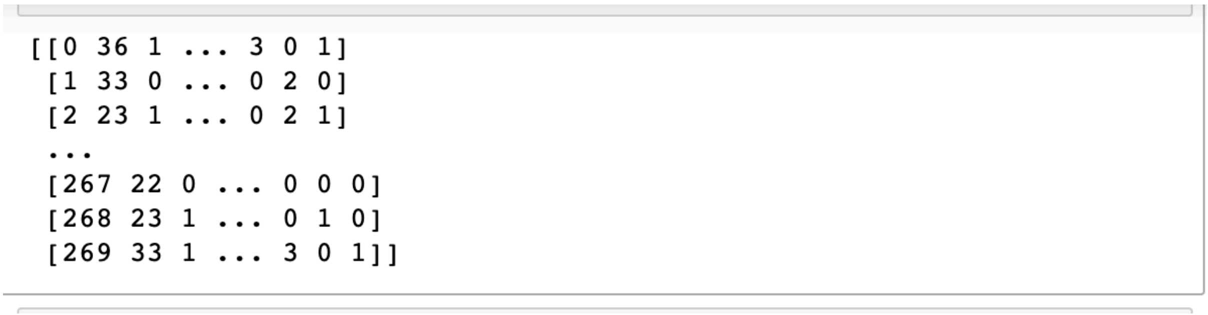
<Program Code>

```
#1

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
```

```
getLabel=[]

tmp_list = a_list

for j in range(len(tmp_list[0])):

    temp=[]

    for i in range(len(tmp_list)):
        temp.append(tmp_list[i][j])

    indexing = {}
    uniqueList = np.unique(temp)

    for i in range(len(uniqueList)):
        indexing[uniqueList[i]] = i

    getLabel.append(indexing)

for j in range(len(tmp_list[0])):

    temp=[]

    for i in range(len(tmp_list)):
        temp.append(tmp_list[i][j])

    le.fit(temp)
    list(le.classes_)
    tempTrans = le.transform(temp)

    for i in range(len(tmp_list)):
        tmp_list[i][j] = tempTrans[i]

a_list_enc = tmp_list




df= pd.DataFrame(data=a_list_enc)
```

```
df.to_csv('a_list_enc.csv',index=False,header=False)


print(a_list_enc)
```

<Result>



```
[[0 36 1 ... 3 0 1]
 [1 33 0 ... 0 2 0]
 [2 23 1 ... 0 2 1]
 ...
 [267 22 0 ... 0 0 0]
 [268 23 1 ... 0 1 0]
 [269 33 1 ... 3 0 1]]
```

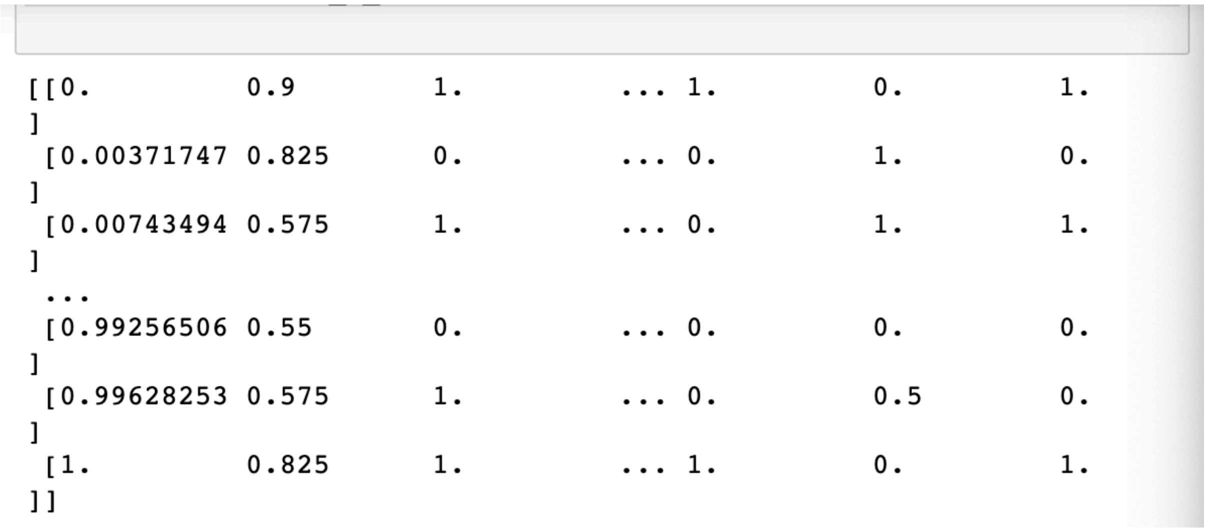
2. Normalize

   1) MinMaxScaler

<Program Code>

```
#2-1

from sklearn.preprocessing import MinMaxScaler

list_temp = a_list_enc


min_max_scaler = MinMaxScaler(feature_range=(0, 1))
min_max_scaler.fit(list_temp)

minmaxscaled_a_list = min_max_scaler.transform(a_list)

print(minmaxscaled_a_list)
```

<Result>



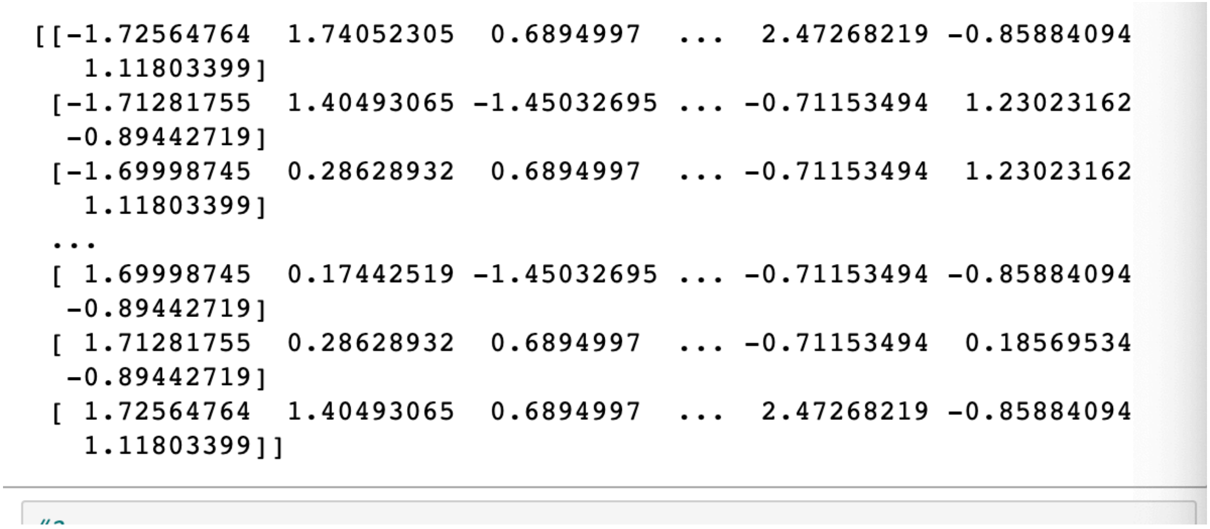
```
[[0.         0.9        1.         ... 1.         0.         1.
 ]
 [0.00371747 0.825      0.         ... 0.         1.         0.
 ]
 [0.00743494 0.575      1.         ... 0.         1.         1.
 ]
 ...
 [0.99256506 0.55       0.         ... 0.         0.         0.
 ]
 [0.99628253 0.575      1.         ... 0.         0.5        0.
 ]
 [1.         0.825      1.         ... 1.         0.         1.
 ]]
```

2) StandardScaler

<Program Code>

```
#2-2
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaler.fit(a_list_enc)

a_list_enc_norm = scaler.transform(a_list_enc)
print(a_list_enc_norm)

import pandas as pd
df= pd.DataFrame(data=a_list_enc_norm)
df.to_csv('a_list_enc_norm.csv', index=False,header=False)
```

<Result>



```
[[-1.72564764  1.74052305  0.6894997  ...  2.47268219 -0.85884094
   1.11803399]
 [-1.71281755  1.40493065 -1.45032695 ... -0.71153494  1.23023162
  -0.89442719]
 [-1.69998745  0.28628932  0.6894997  ... -0.71153494  1.23023162
   1.11803399]
 ...
 [ 1.69998745  0.17442519 -1.45032695 ... -0.71153494 -0.85884094
  -0.89442719]
 [ 1.71281755  0.28628932  0.6894997  ... -0.71153494  0.18569534
  -0.89442719]
 [ 1.72564764  1.40493065  0.6894997  ...  2.47268219 -0.85884094
   1.11803399]]
```

3. Divide_train_test

<Program Code>

```
 #3
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression


X_data=[]
for i in range(len(a_list_enc_norm)):
    X_data.append(a_list_enc_norm[i][:len(a_list_enc_norm[0])-1])

Y_data=[]
for i in range(len(a_list_enc)):
    Y_data.append(a_list_enc[i][-1])

X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size=0.3, random_st
ate=42)
```

4. Running Neural Network

<Program Code>

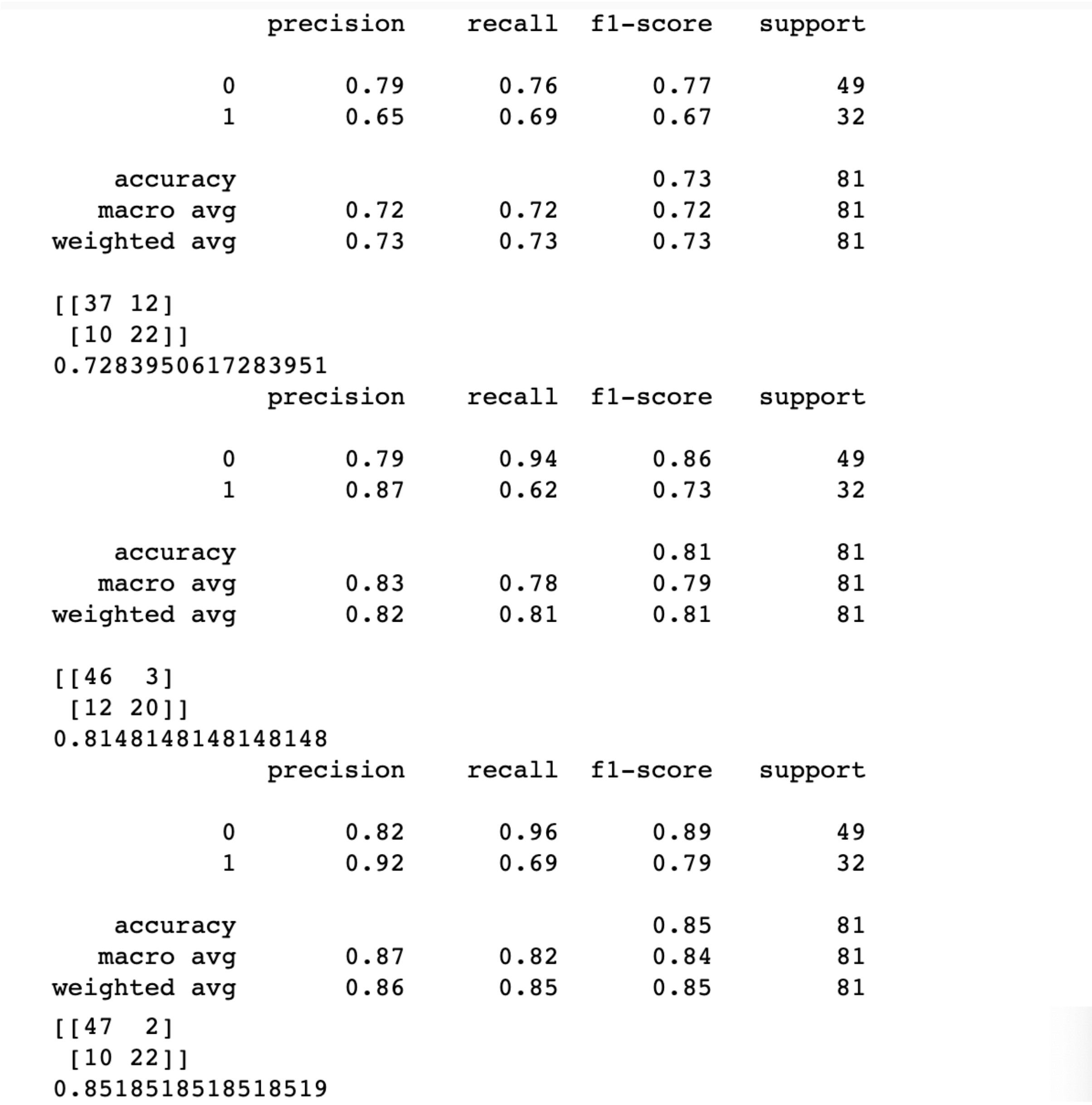
```
#4

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

#a
```

```
accpp0=[]
clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=100)
clf.fit(np.array(X_train), np.array(Y_train))
predictions = clf.predict(X_test)
accpp0.append(accuracy_score(Y_test, predictions))

print(classification_report(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(accuracy_score(Y_test, predictions))

clf = MLPClassifier(hidden_layer_sizes=(40), max_iter=100)
clf.fit(np.array(X_train), np.array(Y_train))
predictions = clf.predict(X_test)
accpp0.append(accuracy_score(Y_test, predictions))

print(classification_report(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(accuracy_score(Y_test, predictions))

clf = MLPClassifier(hidden_layer_sizes=(80), max_iter=100)
clf.fit(np.array(X_train), np.array(Y_train))
predictions = clf.predict(X_test)
accpp0.append(accuracy_score(Y_test, predictions))

print(classification_report(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(accuracy_score(Y_test, predictions))

#b

accpp=[]
clf = MLPClassifier(hidden_layer_sizes=(10,10), max_iter=100)
clf.fit(X_train, Y_train)
predictions =clf.predict(X_test)
accpp.append(accuracy_score(Y_test, predictions))

print(classification_report(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(accuracy_score(Y_test, predictions))
```

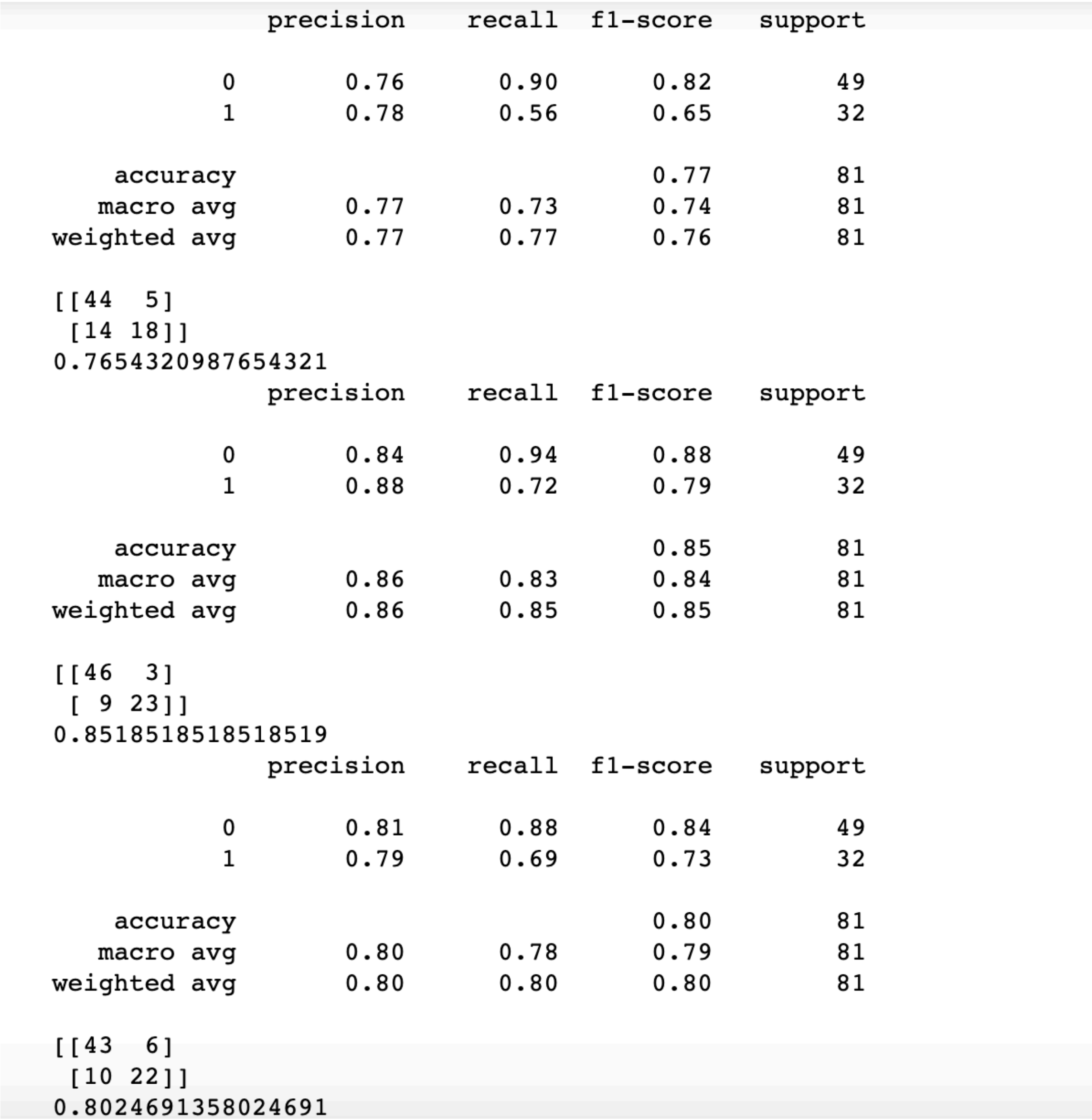
```
clf = MLPClassifier(hidden_layer_sizes=(40,40), max_iter=100)
clf.fit(X_train, Y_train)
predictions =clf.predict(X_test)
accpp.append(accuracy_score(Y_test, predictions))

print(classification_report(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(accuracy_score(Y_test, predictions))
```

```
clf = MLPClassifier(hidden_layer_sizes=(80,80), max_iter=100)
clf.fit(X_train, Y_train)
predictions =clf.predict(X_test)
accpp.append(accuracy_score(Y_test, predictions))


print(classification_report(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(accuracy_score(Y_test, predictions))
```

<Result>



```
              precision    recall  f1-score   support

           0       0.79      0.76      0.77        49
           1       0.65      0.69      0.67        32

    accuracy                           0.73        81
   macro avg       0.72      0.72      0.72        81
weighted avg       0.73      0.73      0.73        81

[[37 12]
 [10 22]]
0.7283950617283951
              precision    recall  f1-score   support

           0       0.79      0.94      0.86        49
           1       0.87      0.62      0.73        32

    accuracy                           0.81        81
   macro avg       0.83      0.78      0.79        81
weighted avg       0.82      0.81      0.81        81

[[46  3]
 [12 20]]
0.8148148148148148
              precision    recall  f1-score   support

           0       0.82      0.96      0.89        49
           1       0.92      0.69      0.79        32

    accuracy                           0.85        81
   macro avg       0.87      0.82      0.84        81
weighted avg       0.86      0.85      0.85        81
[[47  2]
 [10 22]]
0.8518518518518519
```

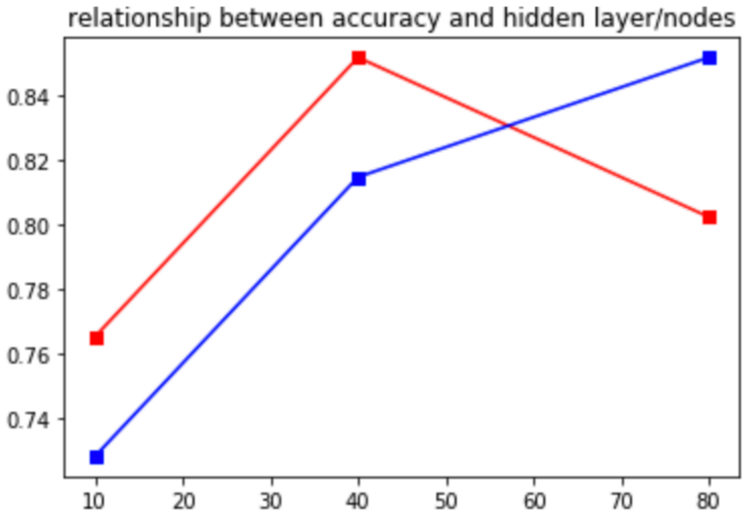


```
              precision    recall  f1-score   support

           0       0.76      0.90      0.82        49
           1       0.78      0.56      0.65        32

    accuracy                           0.77        81
   macro avg       0.77      0.73      0.74        81
weighted avg       0.77      0.77      0.76        81

[[44  5]
 [14 18]]
0.7654320987654321
              precision    recall  f1-score   support

           0       0.84      0.94      0.88        49
           1       0.88      0.72      0.79        32

    accuracy                           0.85        81
   macro avg       0.86      0.83      0.84        81
weighted avg       0.86      0.85      0.85        81

[[46  3]
 [ 9 23]]
0.8518518518518519
              precision    recall  f1-score   support

           0       0.81      0.88      0.84        49
           1       0.79      0.69      0.73        32

    accuracy                           0.80        81
   macro avg       0.80      0.78      0.79        81
weighted avg       0.80      0.80      0.80        81

[[43  6]
 [10 22]]
0.8024691358024691
```

<Program Code>

```
#4-2

import matplotlib.pyplot as plt
x_array = [10,40,80]
plt.plot(x_array, accpp, color ="red", marker='s')
plt.plot(x_array, accpp0, color ="blue", marker='s')
plt.title('relationship between accuracy and hidden layer/nodes')
plt.show()
```

<Result>

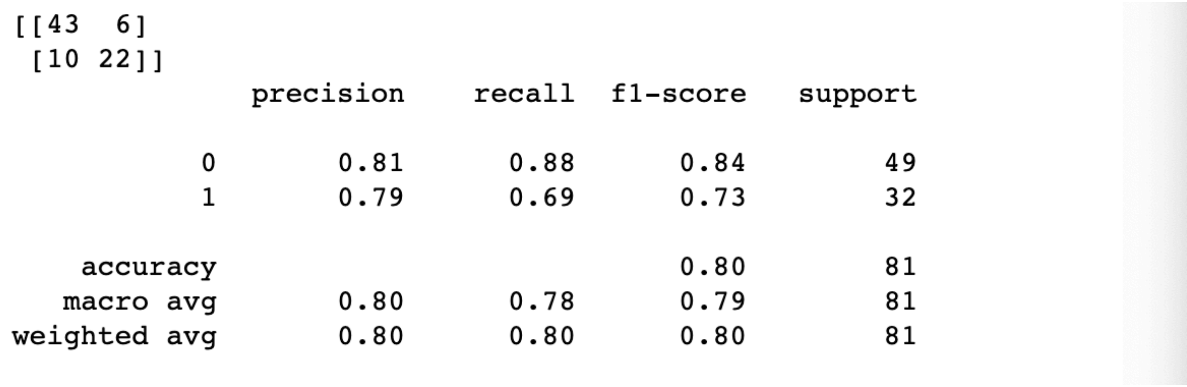


<Program Code>

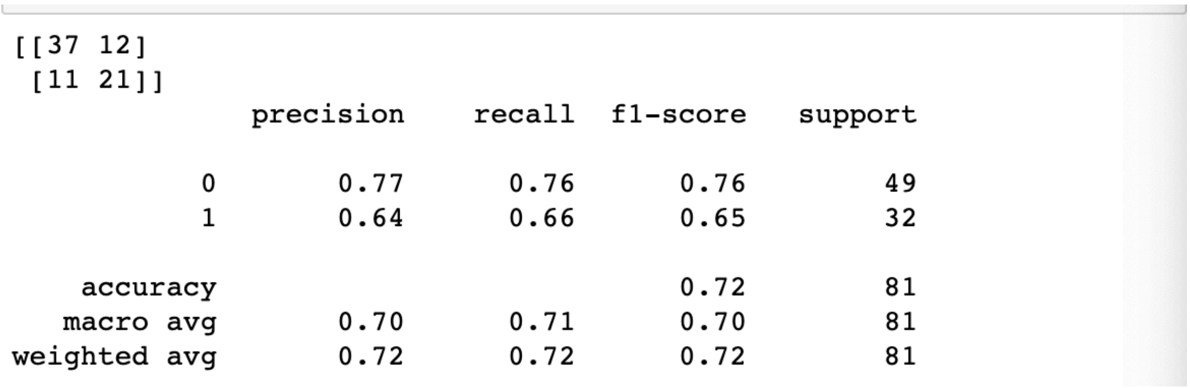
```
#4-3

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='identity',max_iter=200)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))


print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-3

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='identity',max_iter=200)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))


print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-3

arr1 = []
```

```
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='logistic',max_iter=200)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))



#4-3

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='tanh',max_iter=200)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-3

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum =0)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))
```

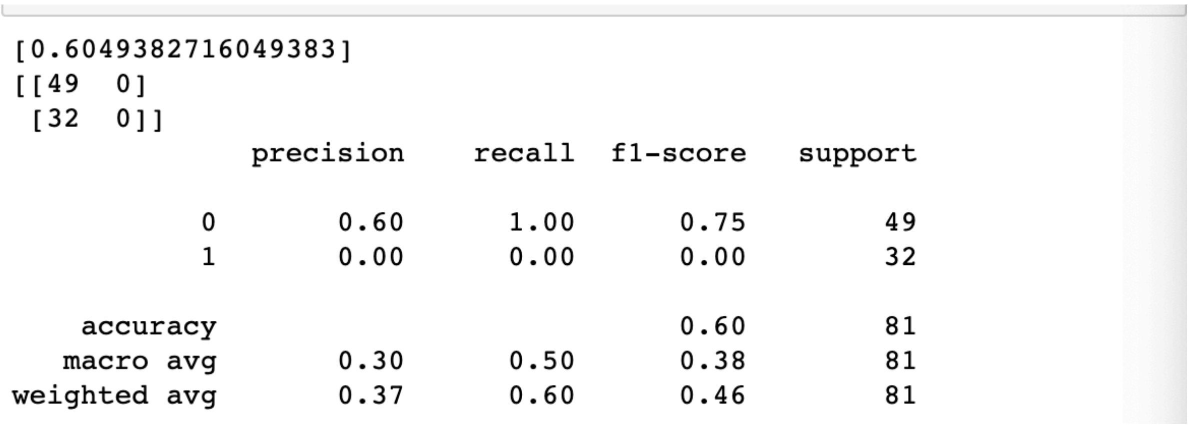
<Result>



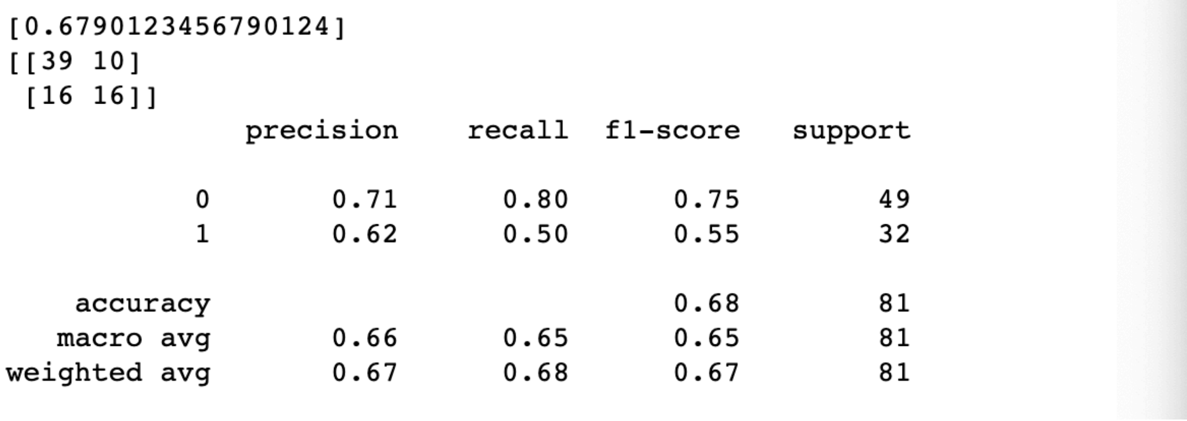
```
[[43  6]
 [10 22]]
              precision    recall  f1-score   support

           0       0.81      0.88      0.84        49
           1       0.79      0.69      0.73        32

    accuracy                           0.80        81
   macro avg       0.80      0.78      0.79        81
weighted avg       0.80      0.80      0.80        81
```



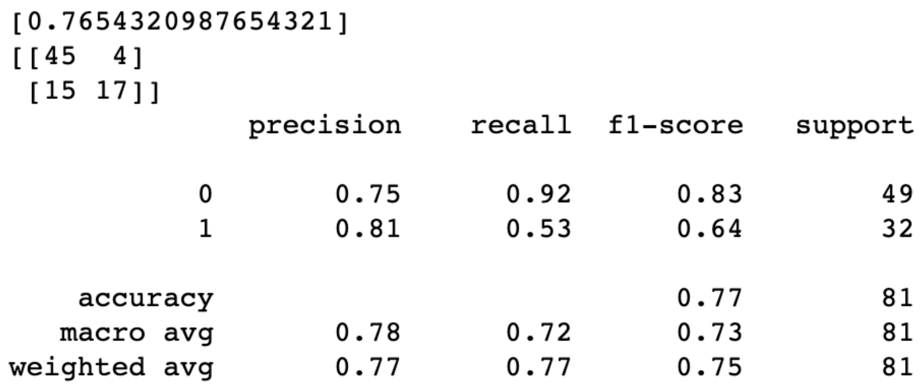
```
[[37 12]
 [11 21]]
              precision    recall  f1-score   support

           0       0.77      0.76      0.76        49
           1       0.64      0.66      0.65        32

    accuracy                           0.72        81
   macro avg       0.70      0.71      0.70        81
weighted avg       0.72      0.72      0.72        81
```



```
[0.6049382716049383]
[[49  0]
 [32  0]]
              precision    recall  f1-score   support

           0       0.60      1.00      0.75        49
           1       0.00      0.00      0.00        32

    accuracy                           0.60        81
   macro avg       0.30      0.50      0.38        81
weighted avg       0.37      0.60      0.46        81
```



```
[0.6790123456790124]
[[39 10]
 [16 16]]
              precision    recall  f1-score   support

           0       0.71      0.80      0.75        49
           1       0.62      0.50      0.55        32

    accuracy                           0.68        81
   macro avg       0.66      0.65      0.65        81
weighted avg       0.67      0.68      0.67        81
```



```
[0.7654320987654321]
[[45  4]
 [15 17]]
              precision    recall  f1-score   support

           0       0.75      0.92      0.83        49
           1       0.81      0.53      0.64        32

    accuracy                           0.77        81
   macro avg       0.78      0.72      0.73        81
weighted avg       0.77      0.77      0.75        81
```

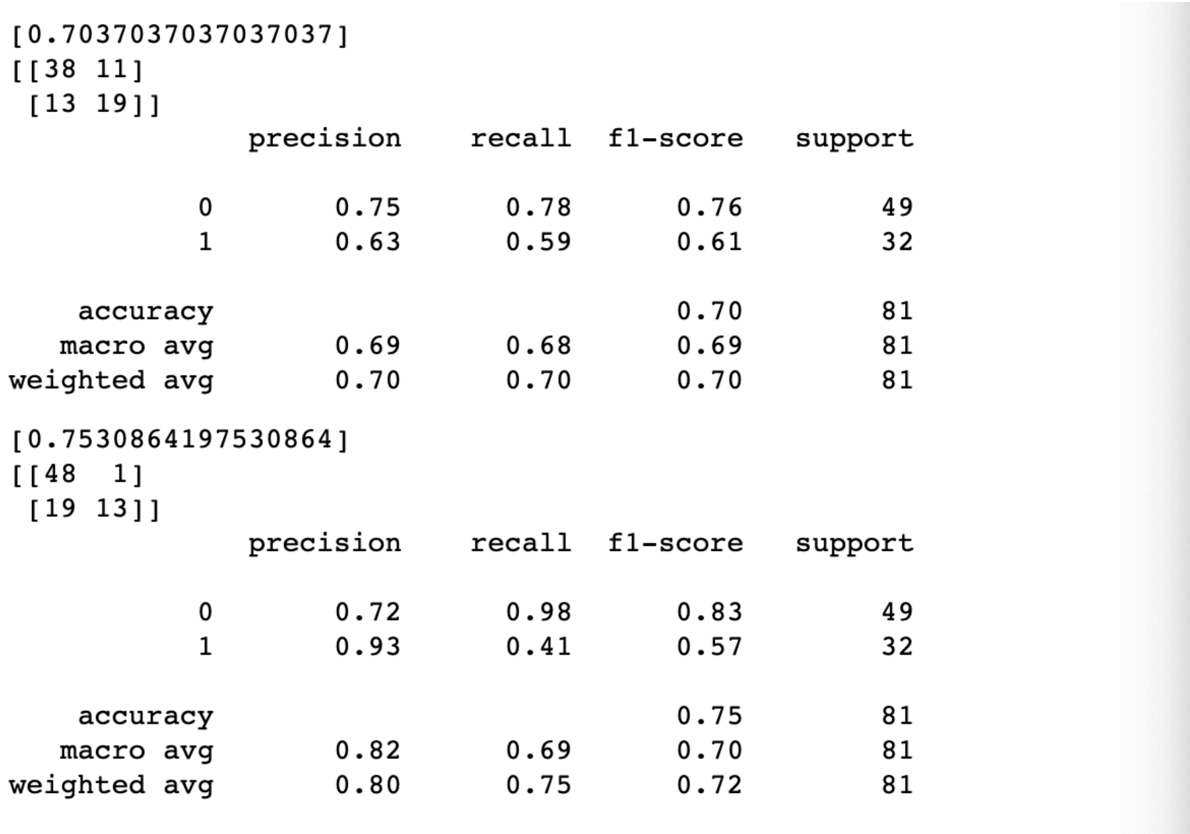
<Program Code>

```
#4-4

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.2)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-4

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.4)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-4

arr1 = []
```

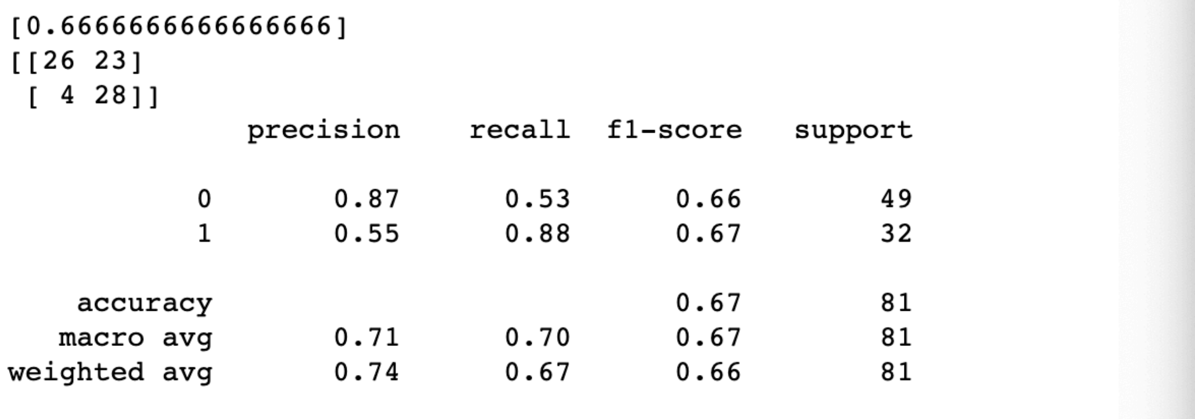
```
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.6)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-4

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-4

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-4

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-5

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8,learning_rate='constant')
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-5

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8,learning_rate='invscaling')
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-5

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8,learning_rate='adaptive')
```

```
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-5

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8,learning_rate='adaptive',learning_rate_init=0.002)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))

#4-5

arr1 = []
clf = MLPClassifier(hidden_layer_sizes=(3,3,3), activation='relu',max_iter=200, momentum
=0.8,learning_rate='adaptive',learning_rate_init=0.003)
clf.fit(X_train,Y_train)
pred = clf.predict(X_test)
arr1.append(accuracy_score(Y_test,pred))
print(arr1)

print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))
```

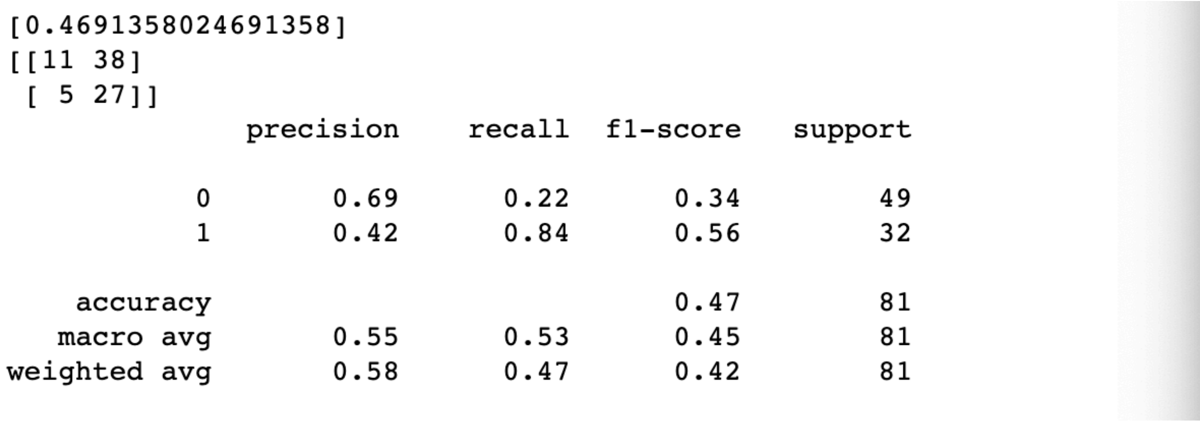
<Result>



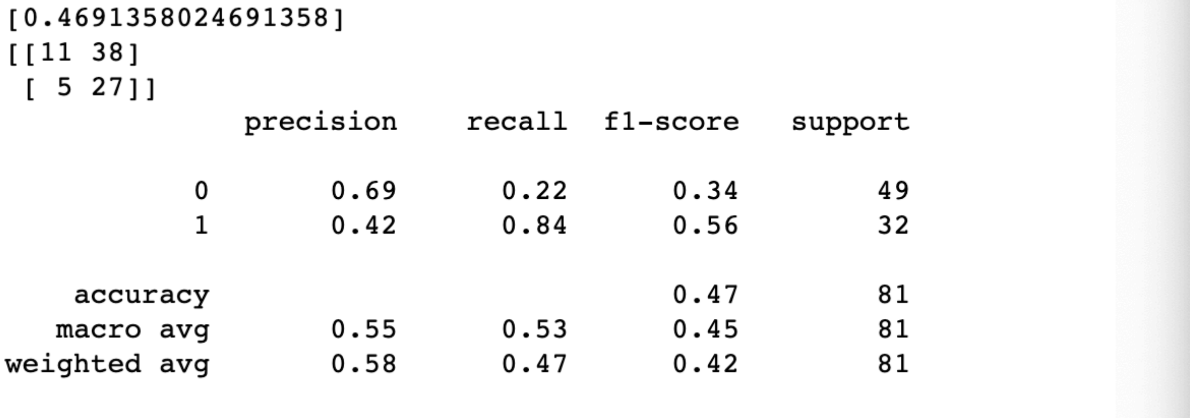
```
[0.7037037037037037]
[[38 11]
 [13 19]]
              precision    recall  f1-score   support

           0       0.75      0.78      0.76        49
           1       0.63      0.59      0.61        32

    accuracy                           0.70        81
   macro avg       0.69      0.68      0.69        81
weighted avg       0.70      0.70      0.70        81

[0.7530864197530864]
[[48  1]
 [19 13]]
              precision    recall  f1-score   support

           0       0.72      0.98      0.83        49
           1       0.93      0.41      0.57        32

    accuracy                           0.75        81
   macro avg       0.82      0.69      0.70        81
weighted avg       0.80      0.75      0.72        81
```



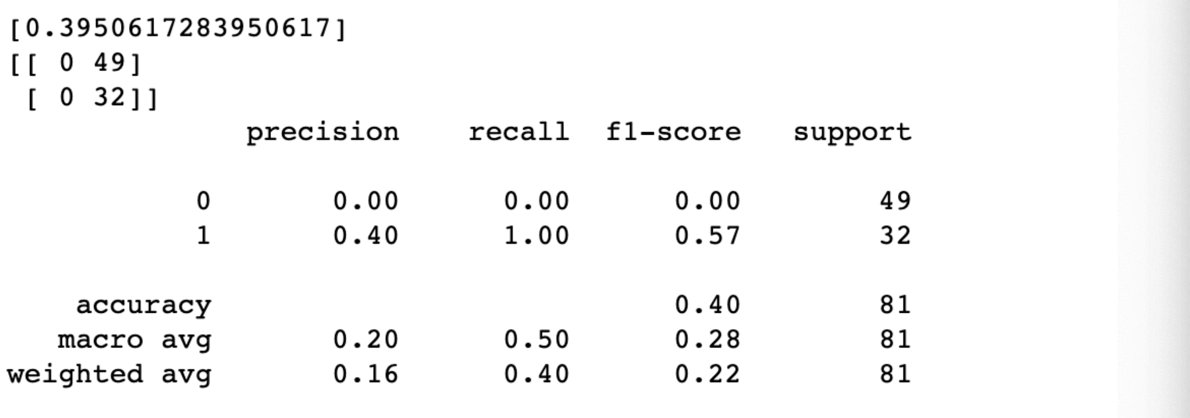
```
[0.6666666666666666]
[[26 23]
 [ 4 28]]
              precision    recall  f1-score   support

           0       0.87      0.53      0.66        49
           1       0.55      0.88      0.67        32

    accuracy                           0.67        81
   macro avg       0.71      0.70      0.67        81
weighted avg       0.74      0.67      0.66        81
```



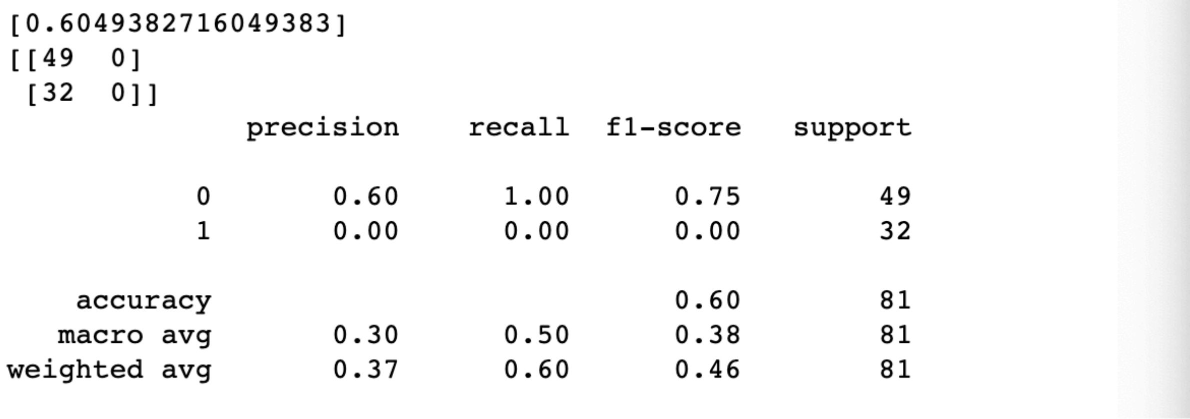
```
[0.4691358024691358]
[[11 38]
 [ 5 27]]
              precision    recall  f1-score   support

           0       0.69      0.22      0.34        49
           1       0.42      0.84      0.56        32

    accuracy                           0.47        81
   macro avg       0.55      0.53      0.45        81
weighted avg       0.58      0.47      0.42        81
```



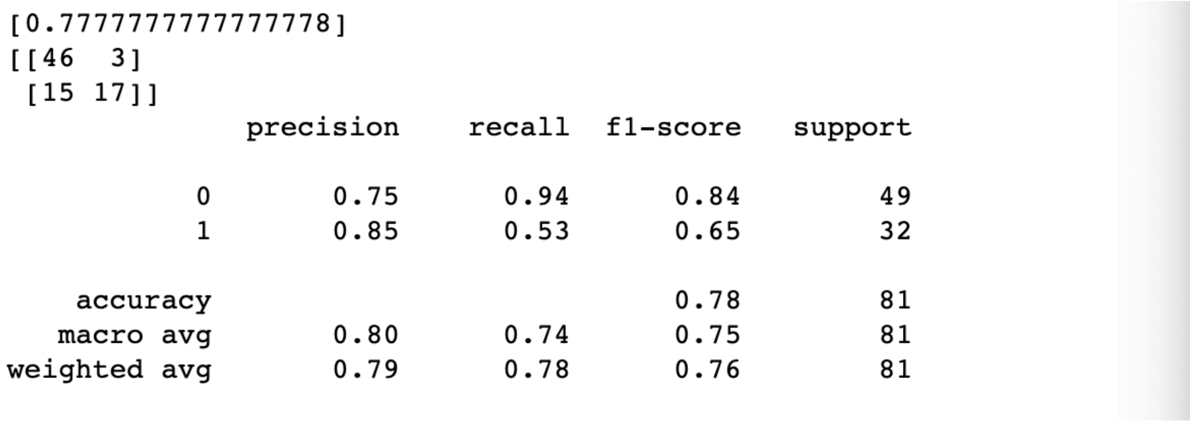
```
[0.4691358024691358]
[[11 38]
 [ 5 27]]
              precision    recall  f1-score   support

           0       0.69      0.22      0.34        49
           1       0.42      0.84      0.56        32

    accuracy                           0.47        81
   macro avg       0.55      0.53      0.45        81
weighted avg       0.58      0.47      0.42        81
```



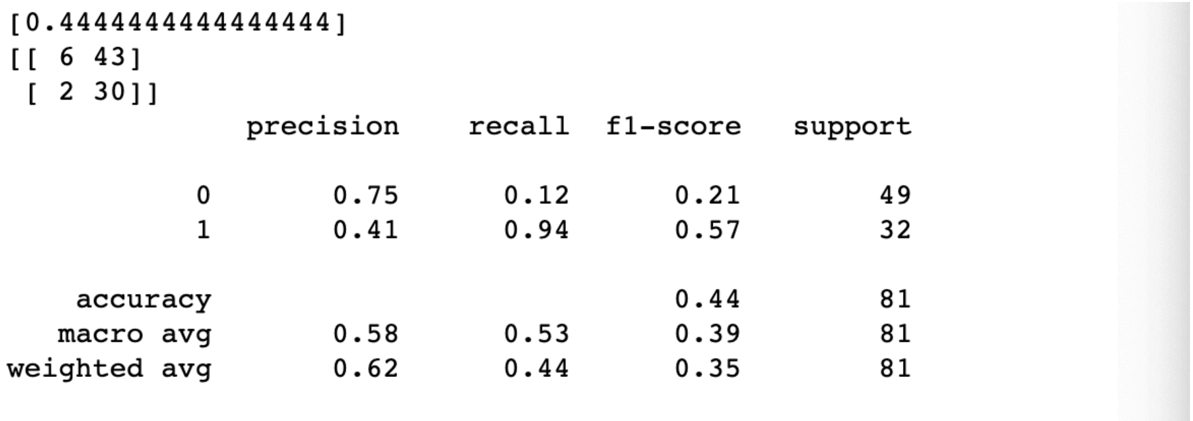
```
[0.3950617283950617]
[[ 0 49]
 [ 0 32]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        49
           1       0.40      1.00      0.57        32

    accuracy                           0.40        81
   macro avg       0.20      0.50      0.28        81
weighted avg       0.16      0.40      0.22        81
```



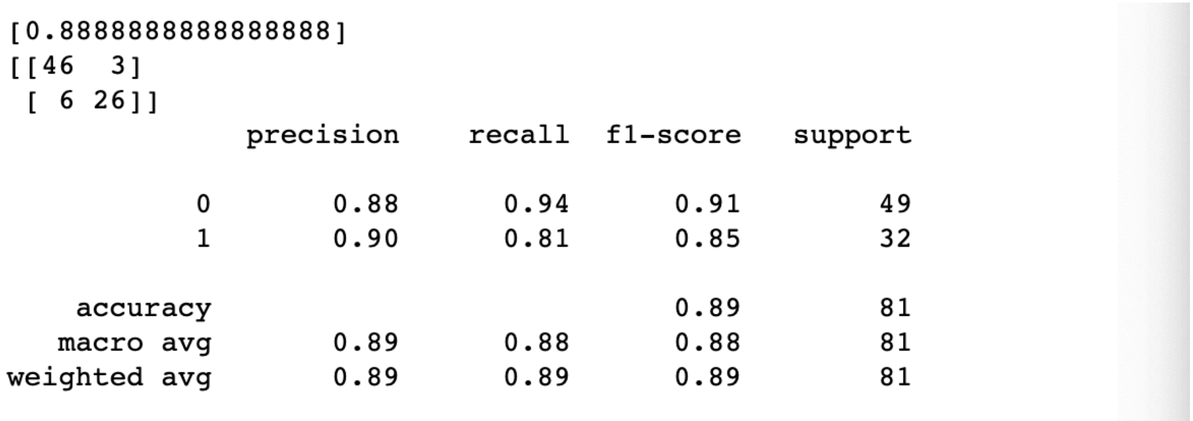
```
[0.6049382716049383]
[[49  0]
 [32  0]]
              precision    recall  f1-score   support

           0       0.60      1.00      0.75        49
           1       0.00      0.00      0.00        32

    accuracy                           0.60        81
   macro avg       0.30      0.50      0.38        81
weighted avg       0.37      0.60      0.46        81
```



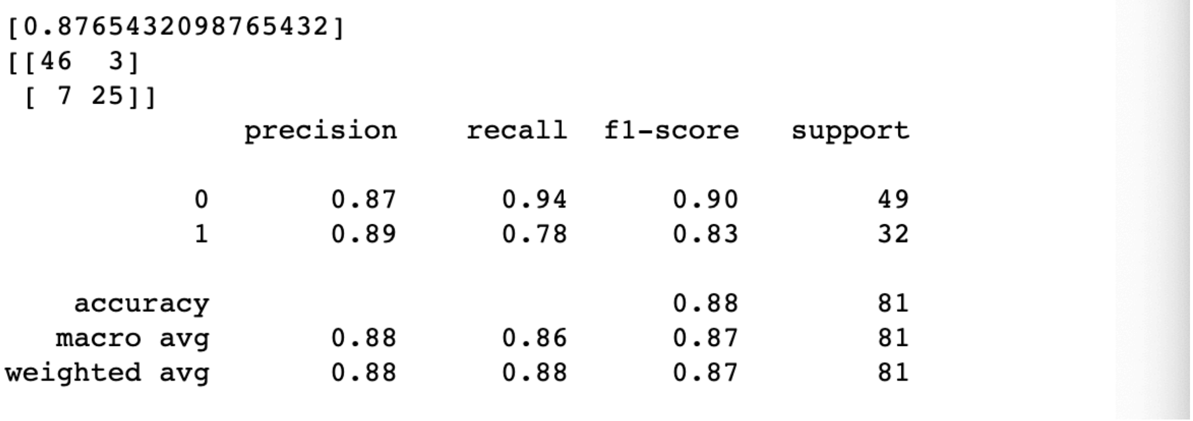
```
[0.7777777777777778]
[[46  3]
 [15 17]]
              precision    recall  f1-score   support

           0       0.75      0.94      0.84        49
           1       0.85      0.53      0.65        32

    accuracy                           0.78        81
   macro avg       0.80      0.74      0.75        81
weighted avg       0.79      0.78      0.76        81
```



```
[0.4444444444444444]
[[ 6 43]
 [ 2 30]]
              precision    recall  f1-score   support

           0       0.75      0.12      0.21        49
           1       0.41      0.94      0.57        32

    accuracy                           0.44        81
   macro avg       0.58      0.53      0.39        81
weighted avg       0.62      0.44      0.35        81
```



```
[0.8888888888888888]
[[46  3]
 [ 6 26]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91        49
           1       0.90      0.81      0.85        32

    accuracy                           0.89        81
   macro avg       0.89      0.88      0.88        81
weighted avg       0.89      0.89      0.89        81
```



```
[0.8765432098765432]
[[46  3]
 [ 7 25]]
              precision    recall  f1-score   support

           0       0.87      0.94      0.90        49
           1       0.89      0.78      0.83        32

    accuracy                           0.88        81
   macro avg       0.88      0.86      0.87        81
weighted avg       0.88      0.88      0.87        81
```

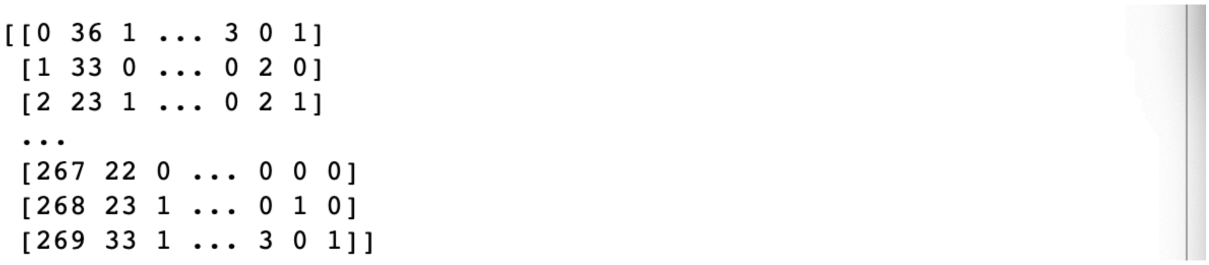
5. Discretization

<Program Code>

```
#5
from sklearn.preprocessing import KBinsDiscretizer
list_temp=a_list_enc

disc = KBinsDiscretizer(n_bins=4,encode='ordinal', strategy='uniform')
disc.fit_transform(list_temp)

a_list_enc_disc = list_temp
```

```
print(a_list_enc_disc)

df=pd.DataFrame(data=a_list_enc_disc)
df.to_csv('a_list_enc_disc.csv',index=False,header=False)
```

<Result>



```
[[0 36 1 ... 3 0 1]
 [1 33 0 ... 0 2 0]
 [2 23 1 ... 0 2 1]
 ...
 [267 22 0 ... 0 0 0]
 [268 23 1 ... 0 1 0]
 [269 33 1 ... 3 0 1]]
```

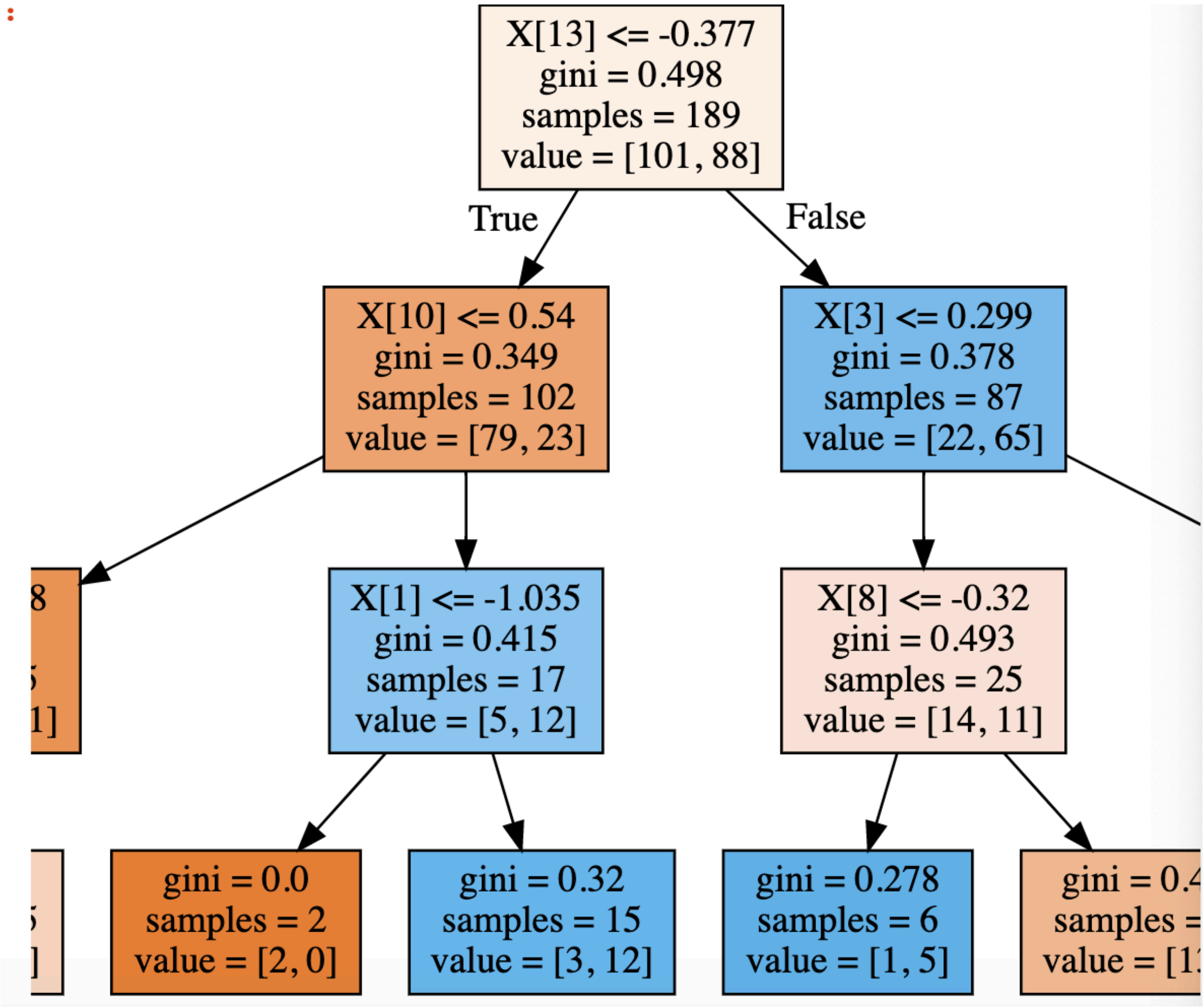
6. Running Decision Tree

<Program Code>

```
#6
from sklearn import tree

X_data2=[]
for i in range(len(a_list_enc_disc)):
      X_data2.append(a_list_enc_disc[i][:len(a_list_enc_disc[0])-1])
Y_data2=[]
for i in range(len(a_list_enc)):
      Y_data2.append(a_list_enc[i][-1])

X_train, X_test, Y_train, Y_test = train_test_split(X_data2, Y_data2, test_size=0.3, random_
state=42)


scaler.fit(a_list_enc)

a_list_enc_norm = scaler.transform(a_list_enc)
print(a_list_enc_norm)


scaler.fit(X_train)
scaled_X_train = scaler.transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

```
#entropy
clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
clf.fit(scaled_X_train,Y_train)
Y_pr = clf.predict(scaled_X_test)
print(accuracy_score(Y_test, Y_pr))


#gini
clf = tree.DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)
clf.fit(scaled_X_train,Y_train)
Y_pr = clf.predict(scaled_X_test)
print(accuracy_score(Y_test, Y_pr))

#6-2
import graphviz

togradata = tree.export_graphviz(clf,out_file=None, filled=True)
graph =graphviz.Source(togradata)

graph
```

<Result>

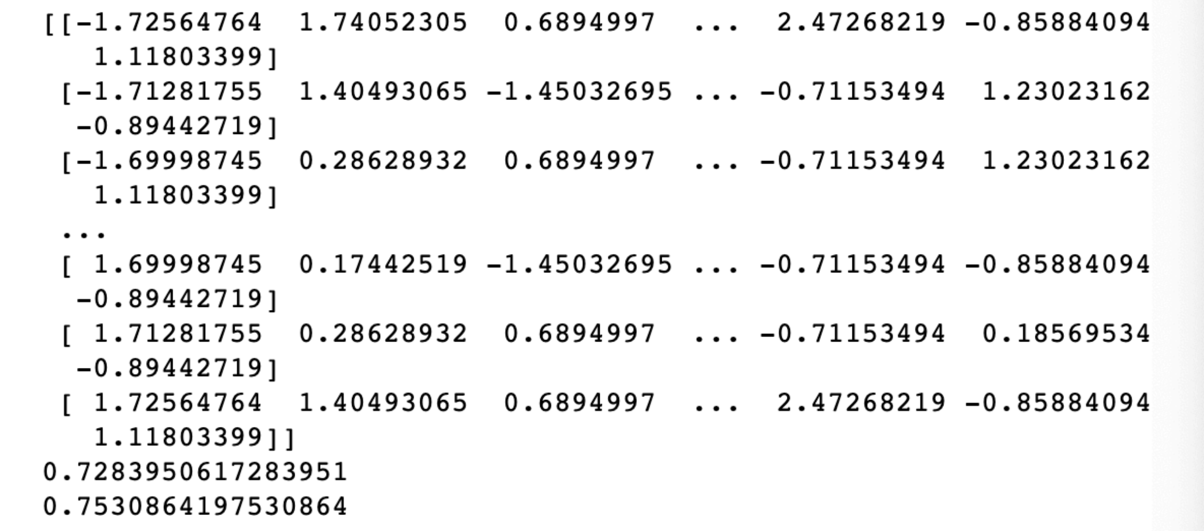


<Program Code>

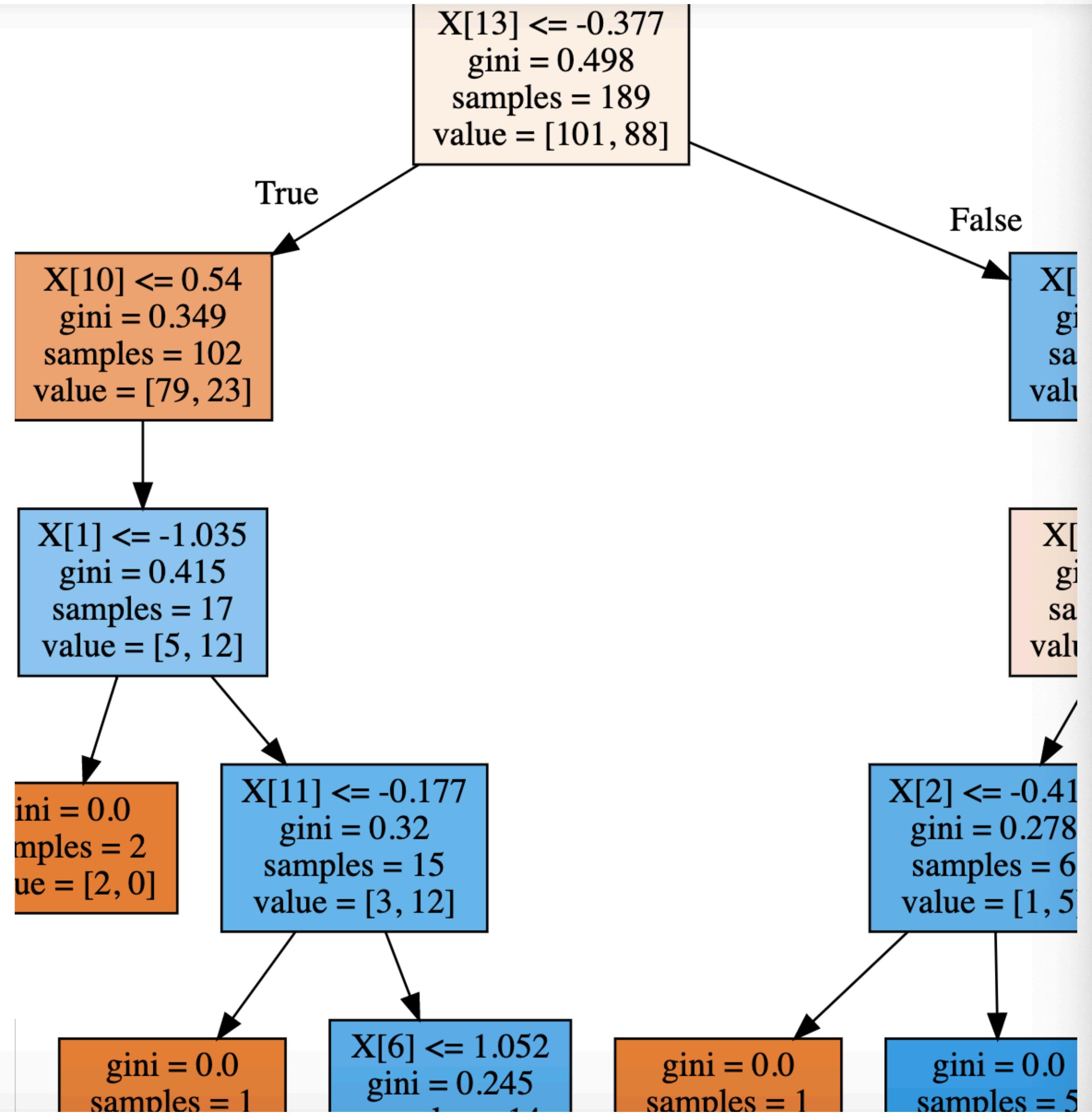
```
#6-3


X_data2=[]
for i in range(len(a_list_enc_disc)):
    X_data2.append(a_list_enc_disc[i][:len(a_list_enc_disc[0])-1])
Y_data2=[]
for i in range(len(a_list_enc)):
    Y_data2.append(a_list_enc[i][-1])

X_train, X_test, Y_train, Y_test = train_test_split(X_data2, Y_data2, test_size=0.3, random_state=42)
```

```
scaler.fit(a_list_enc)

a_list_enc_norm = scaler.transform(a_list_enc)
print(a_list_enc_norm)


scaler.fit(X_train)
scaled_X_train = scaler.transform(X_train)
scaled_X_test = scaler.transform(X_test)

#entropy
clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=42)
clf.fit(scaled_X_train,Y_train)
Y_pr = clf.predict(scaled_X_test)
print(accuracy_score(Y_test, Y_pr))


#gini
clf = tree.DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)
clf.fit(scaled_X_train,Y_train)
Y_pr = clf.predict(scaled_X_test)
print(accuracy_score(Y_test, Y_pr))

#6-3-2


togradata = tree.export_graphviz(clf,out_file=None, filled=True)
graph =graphviz.Source(togradata)

graph
```

<Result>



```
[[-1.72564764  1.74052305  0.6894997  ...  2.47268219 -0.85884094
   1.11803399]
 [-1.71281755  1.40493065 -1.45032695 ... -0.71153494  1.23023162
  -0.89442719]
 [-1.69998745  0.28628932  0.6894997  ... -0.71153494  1.23023162
   1.11803399]
 ...
 [ 1.69998745  0.17442519 -1.45032695 ... -0.71153494 -0.85884094
  -0.89442719]
 [ 1.71281755  0.28628932  0.6894997  ... -0.71153494  0.18569534
  -0.89442719]
 [ 1.72564764  1.40493065  0.6894997  ...  2.47268219 -0.85884094
   1.11803399]]
0.7283950617283951
0.7530864197530864
```


X[13] <= -0.377
gini = 0.498
samples = 189
value = [101, 88]
True
False
X[10] <= 0.54
gini = 0.349
samples = 102
value = [79, 23]
X[
gi
sa
valu
X[1] <= -1.035
gini = 0.415
samples = 17
value = [5, 12]
X[
gi
sa
valu
ini = 0.0
mples = 2
ue = [2, 0]
X[11] <= -0.177
gini = 0.32
samples = 15
value = [3, 12]
X[2] <= -0.41
gini = 0.278
samples = 6
value = [1, 5]
gini = 0.0
samples = 1
X[6] <= 1.052
gini = 0.245
gini = 0.0
samples = 1
gini = 0.0
samples = 5