Machine Learning Homework 2\_Report

2017112167 이수진

1. 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>

스크린샷이(가) 표시된 사진

자동 생성된 설명

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>

스크린샷이(가) 표시된 사진

자동 생성된 설명

1. Normalize
2. MinMaxScaler

<Program Code>

#2-1

from sklearn.preprocessing import MinMaxScaler

list\_temp = a\_list\_enc

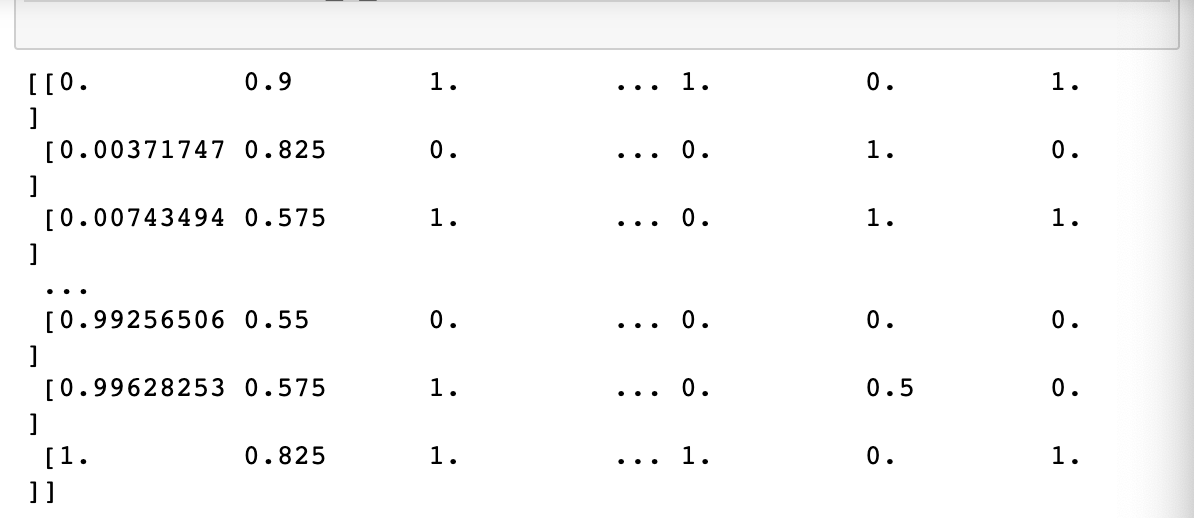
min\_max\_scaler = MinMaxScaler(feature\_range=(0, 1))

min\_max\_scaler.fit(list\_temp)

minmaxscaled\_a\_list = min\_max\_scaler.transform(a\_list)

print(minmaxscaled\_a\_list)

<Result>



1. 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)

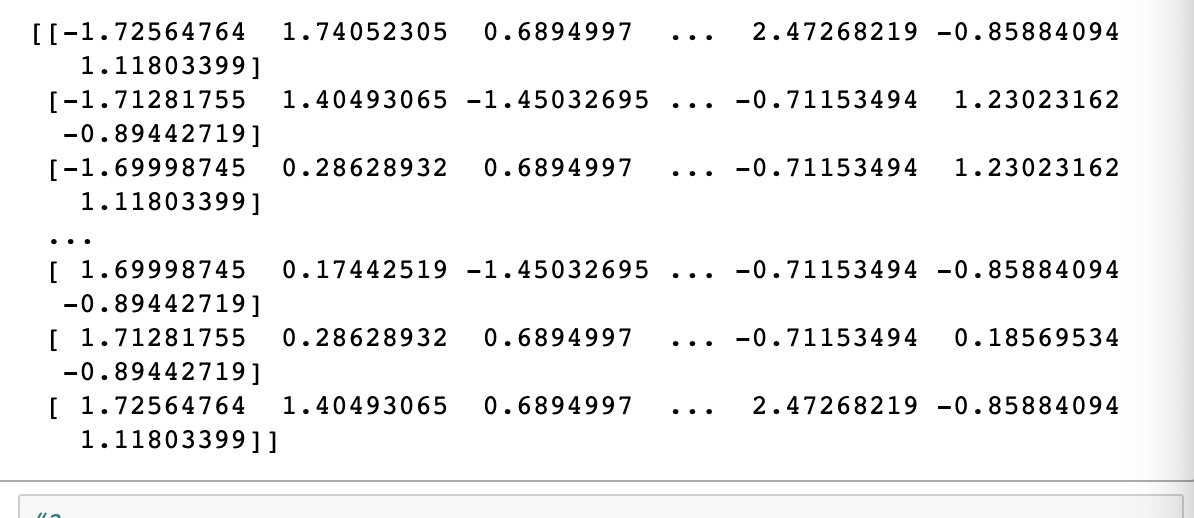
print(a\_list\_enc\_norm)

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. 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\_state=42)

1. 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>

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명



텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

<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>

지도, 텍스트이(가) 표시된 사진

자동 생성된 설명

<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)

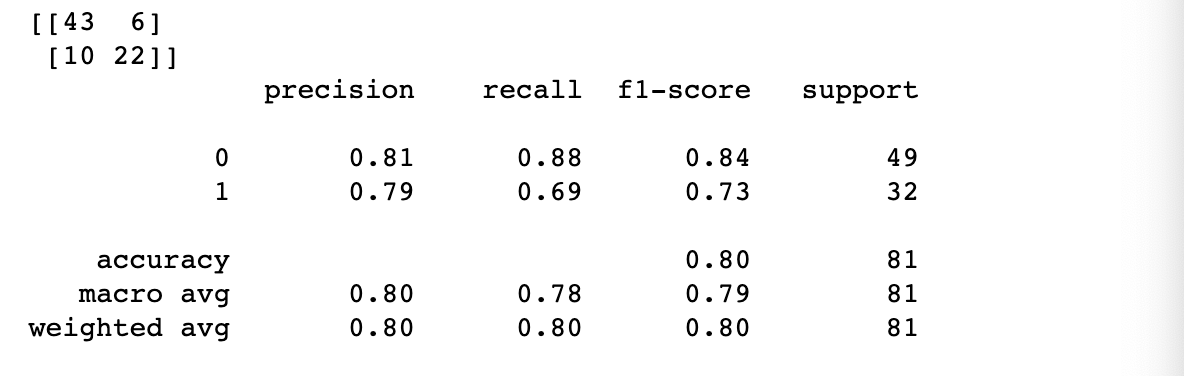
arr1.append(accuracy\_score(Y\_test,pred))

print(arr1)

print(confusion\_matrix(Y\_test,pred))

print(classification\_report(Y\_test,pred))

<Result>

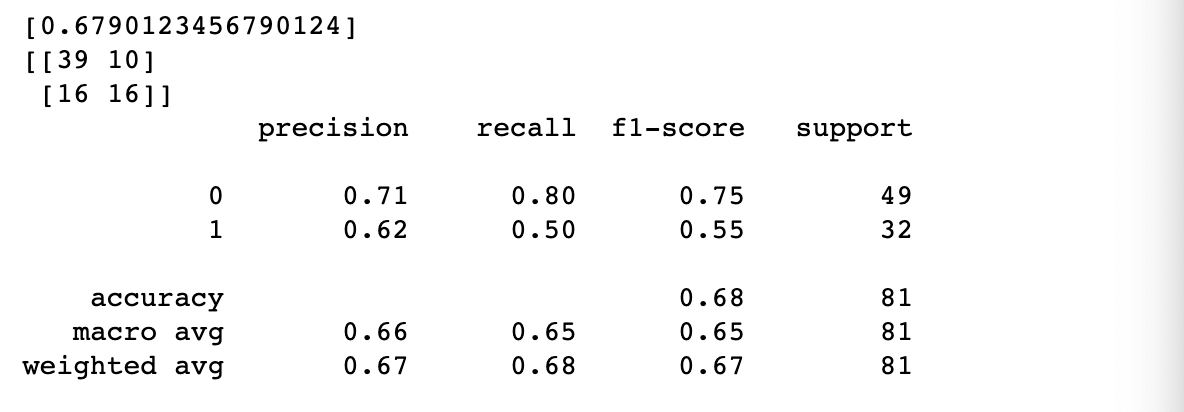


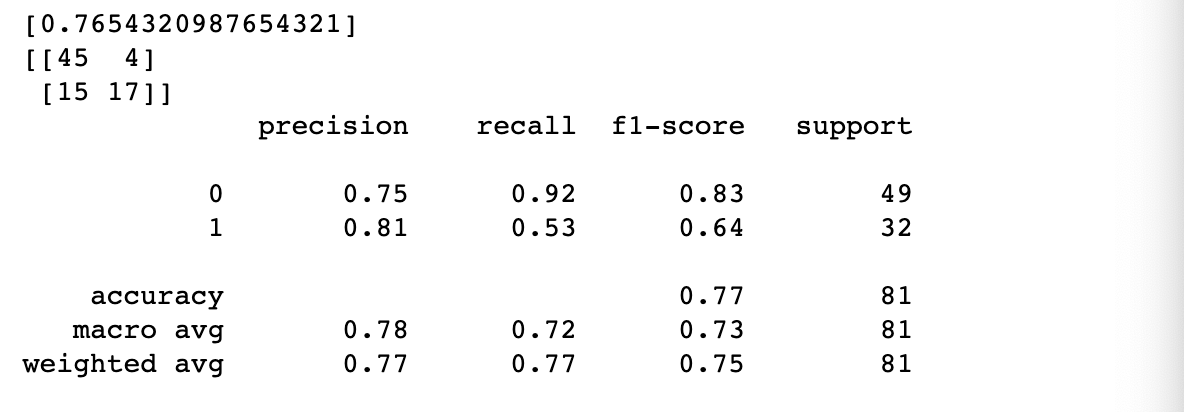
스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명





<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)

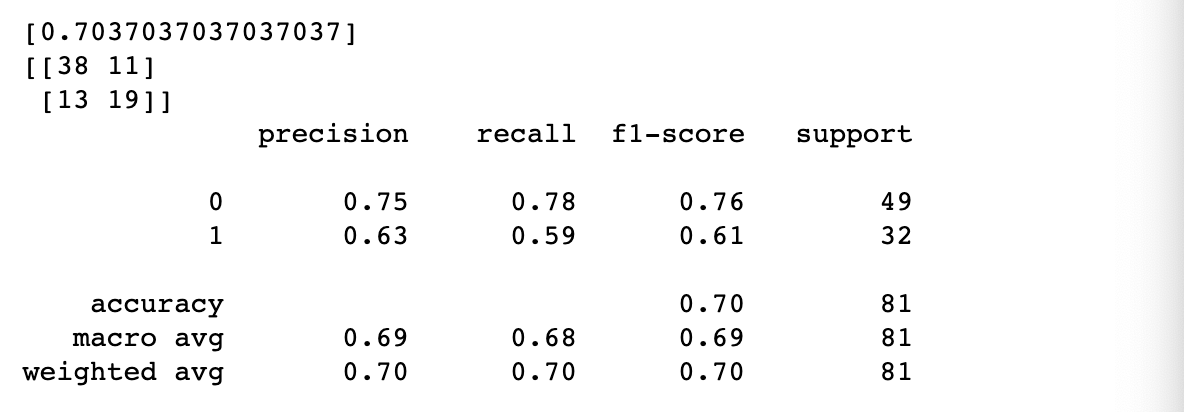
arr1.append(accuracy\_score(Y\_test,pred))

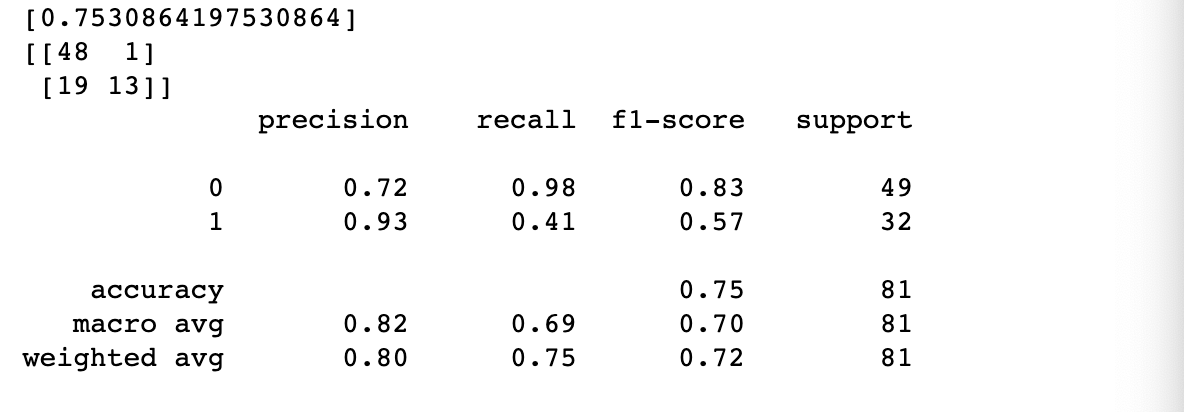
print(arr1)

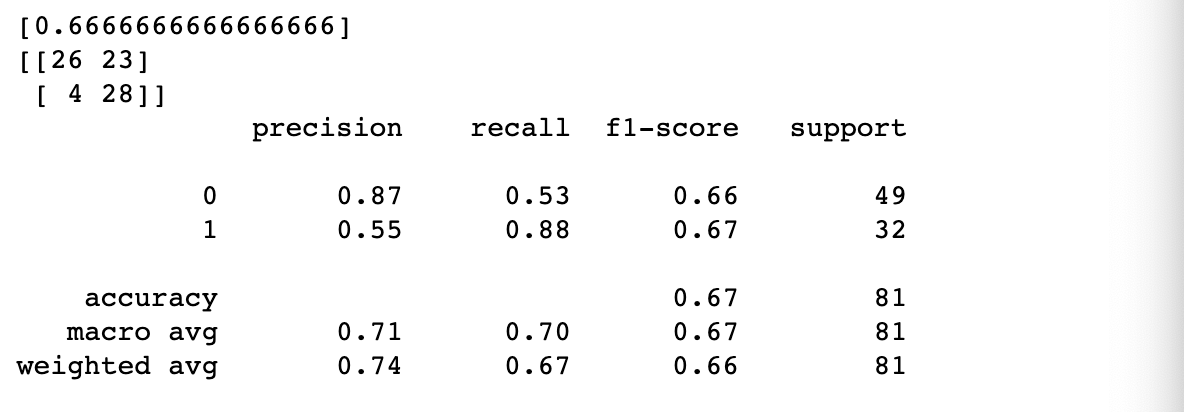
print(confusion\_matrix(Y\_test,pred))

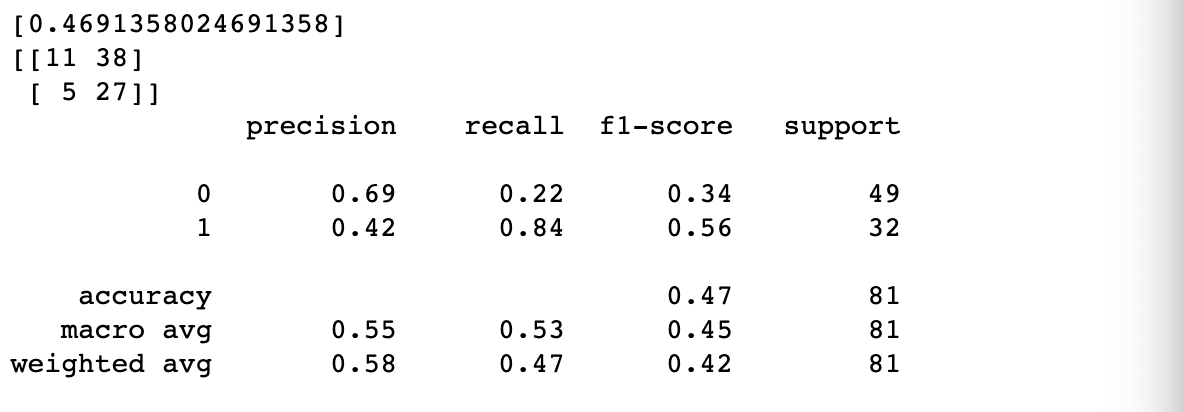
print(classification\_report(Y\_test,pred))

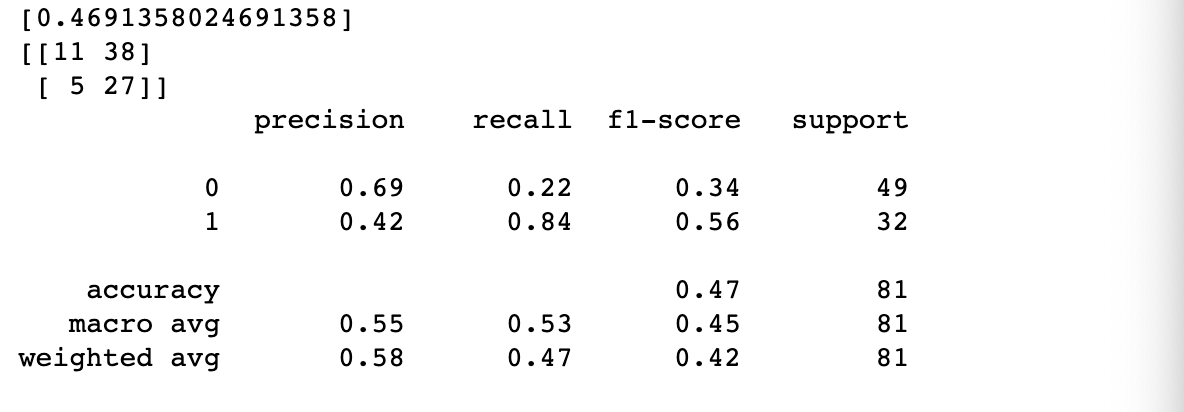
<Result>

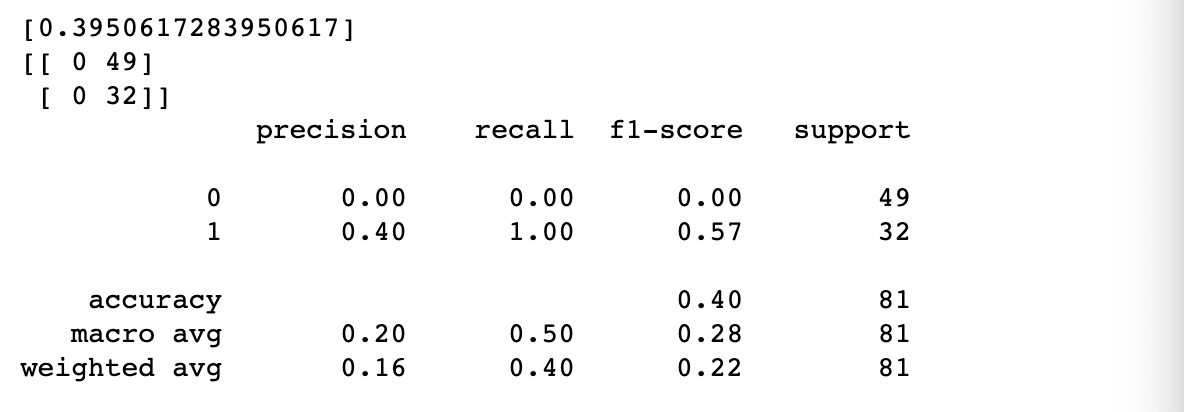


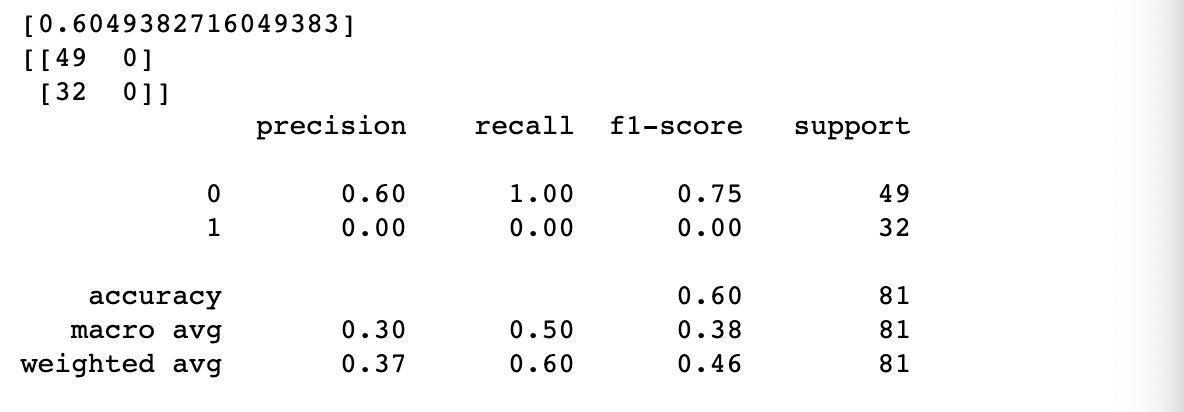


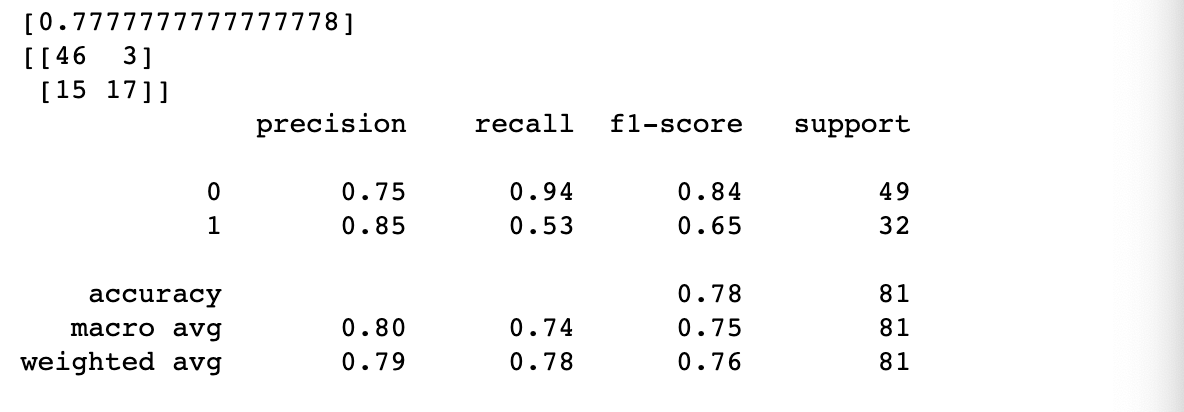


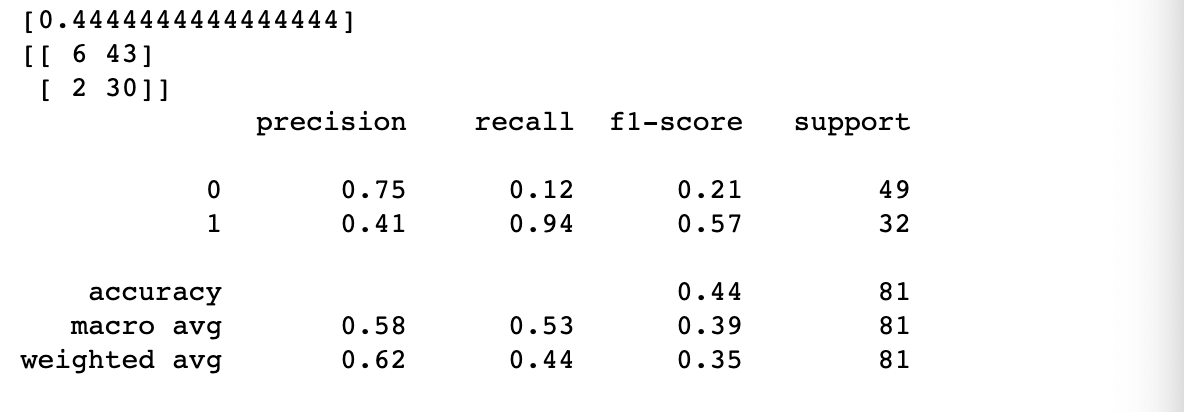


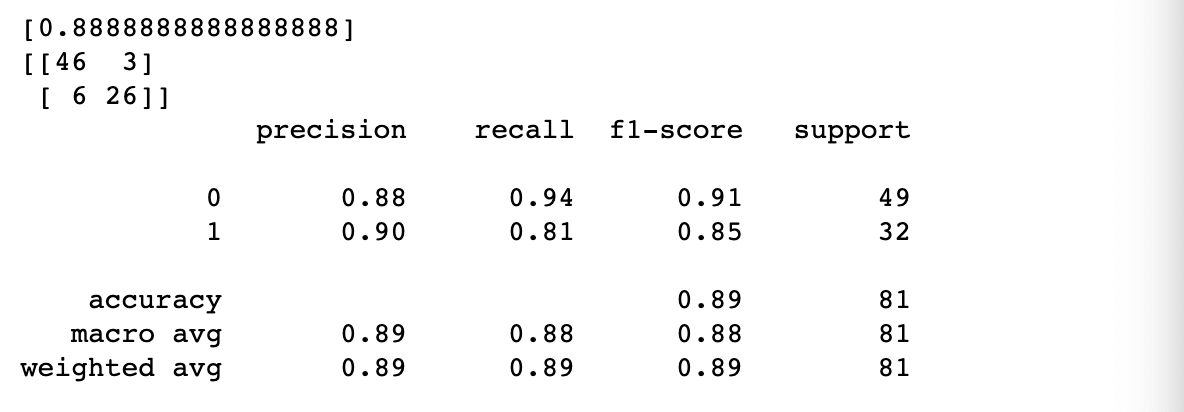


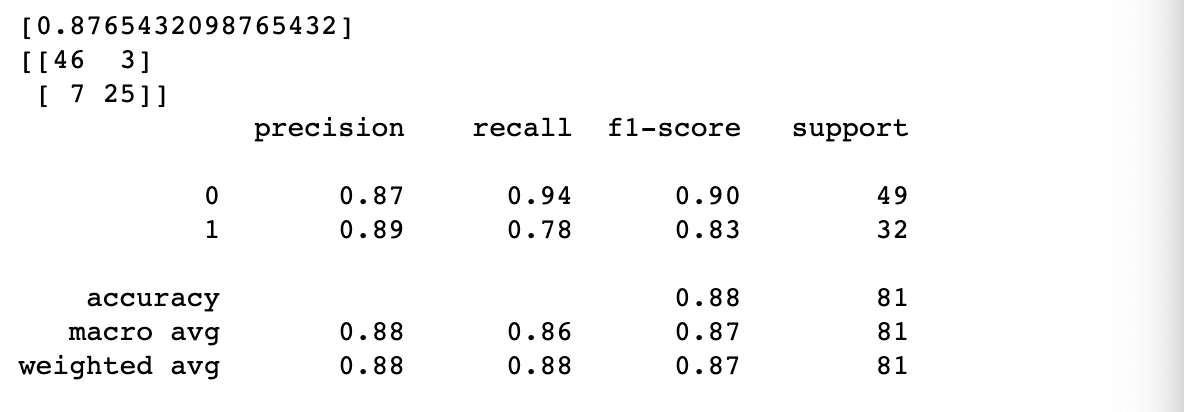












1. Discretization

<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>



1. 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>

텍스트, 표지판, 시계이(가) 표시된 사진

자동 생성된 설명

<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))

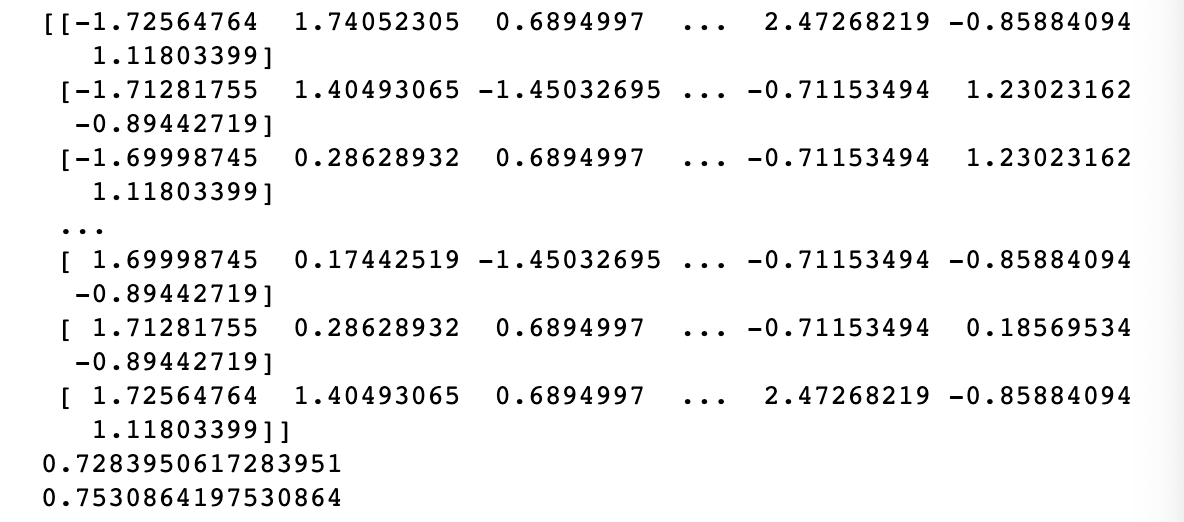
#6-3-2

togradata = tree.export\_graphviz(clf,out\_file=None, filled=True)

graph =graphviz.Source(togradata)

graph

<Result>



텍스트, 표지판이(가) 표시된 사진

자동 생성된 설명