**Machine Learning Homework 3\_Report**

**2017112167 이수진**

1. Read File

<PROGRAM CODE>

#0 READ FILE

#read your csv file into 2-dimenson list (a\_list)

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

<RESULT> : None

스크린샷이(가) 표시된 사진

자동 생성된 설명1. Preprocessing

<PROGRAM CODE>

#1-1 PREPROCESSING - LABEL ENCODING

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)

#1-2 PREPROCESSING - NOMALIZATION

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(a\_list\_enc)

a\_list\_enc\_norm = scaler.transform(a\_list\_enc)

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> : None

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명2. Divide into train & test

<PROGRAM CODE>

#2 Divide into train & test

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=4

<RESULT> : None

스크린샷이(가) 표시된 사진

자동 생성된 설명

3. Running IBL

<PROGRAM CODE>

#3 Running IBL (K nearest neighbor)

from sklearn.neighbors import NearestNeighbors

from sklearn.neighbors import KNeighborsClassifier

import sklearn

import matplotlib.pyplot as plt

<RESULT> : None

스크린샷이(가) 표시된 사진

자동 생성된 설명

3-1)

<PROGRAM CODE>

#3-1) run the algorithm by changing the value of n\_neighbors to 1, 5, 9, 15

# Draw a graph showing the relationship between accuracy and n\_neighbors.

nei\_1 = KNeighborsClassifier(n\_neighbors=1)

nei\_1.fit(X\_train, Y\_train)

predic\_1 = nei\_1.predict(X\_test)

cal\_nei\_1 = sklearn.metrics.accuracy\_score(Y\_test, predic\_1)

nei\_5 = KNeighborsClassifier(n\_neighbors=5)

nei\_5.fit(X\_train, Y\_train)

predic\_5 = nei\_5.predict(X\_test)

cal\_nei\_2 = sklearn.metrics.accuracy\_score(Y\_test, predic\_5)

nei\_9 = KNeighborsClassifier(n\_neighbors=9)

nei\_9.fit(X\_train, Y\_train)

predic\_9 = nei\_9.predict(X\_test)

cal\_nei\_3 = sklearn.metrics.accuracy\_score(Y\_test, predic\_9)

nei\_15 = KNeighborsClassifier(n\_neighbors=15)

nei\_15.fit(X\_train, Y\_train)

predic\_15 = nei\_15.predict(X\_test)

cal\_nei\_4 = sklearn.metrics.accuracy\_score(Y\_test, predic\_15)

x\_array = [1,5,9,15]

y\_array = [cal\_nei\_1,cal\_nei\_2,cal\_nei\_3,cal\_nei\_4]

print("IBL / accuracy when n\_neighbors=1 : ",cal\_nei\_1)

print("IBL / accuracy when n\_neighbors=5 : ",cal\_nei\_2)

print("IBL / accuracy when n\_neighbors=9 : ",cal\_nei\_3)

print("IBL / accuracy when n\_neighbors=15 : ",cal\_nei\_4)

plt.plot(x\_array, y\_array, color ="red")

plt.title('the relationship between accuracy and n\_neighbors')

plt.xlabel('n\_neighbors')

plt.ylabel('accuracy')

plt.show()

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

3-2)

<PROGRAM CODE>

#3-2) run the algorithm by changing the value of weights to

#‘uniform’ and ‘distance’, respectively.

# Compare the results and explain the meaning of the results.

nei\_uni = KNeighborsClassifier(weights='uniform')

nei\_uni.fit(X\_train, Y\_train)

predic\_uni = nei\_uni.predict(X\_test)

cal\_nei\_uni = sklearn.metrics.accuracy\_score(Y\_test, predic\_uni)

nei\_dis=KNeighborsClassifier(weights='distance')

nei\_dis.fit(X\_train, Y\_train)

predic\_dis = nei\_dis.predict(X\_test)

cal\_nei\_dis = sklearn.metrics.accuracy\_score(Y\_test, predic\_dis)

print("IBL / accuracy when weights='uniform' : ",cal\_nei\_uni)

print("IBL / accuracy weights='distance': ",cal\_nei\_dis)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<Explain>

Accuracy is the same for both weights=’uniform’ and weights=’distance’. so in this situation distance of each data is equal.

3-3)

<PROGRAM CODE>

#3-3)run the algorithm by changing the value of p to 1 (Manhattan) and 2(Euclidean).

# Compare the results and explain the meaning of the results.

nei\_p1=KNeighborsClassifier(p=1)

nei\_p1.fit(X\_train, Y\_train)

predic\_p1 = nei\_p1.predict(X\_test)

cal\_nei\_p1 = sklearn.metrics.accuracy\_score(Y\_test, predic\_p1)

nei\_p2=KNeighborsClassifier(p=2)

nei\_p2.fit(X\_train, Y\_train)

predic\_p2 = nei\_p2.predict(X\_test)

cal\_nei\_p2 = sklearn.metrics.accuracy\_score(Y\_test, predic\_p2)

print("IBL / accuracy when p=1 : ",cal\_nei\_p1)

print("IBL / accuracy p=2 : ",cal\_nei\_p2)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<Explain>

- accuracy is greater when we use Euclidean Distance. Manhattan Distance is preferred over the Euclidean distance metric as the dimension of the data increases. In high dimensional data, it is better to use Manhtten Distance. in this case, However, our data is low dimensional data. So, it is better to use the Euclidean Distance

4. Running AdaBoost

<PROGRAM CODE>

#4 Running AdaBoost

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

<RESULT> : None

스크린샷, 조류이(가) 표시된 사진

자동 생성된 설명

4-1)

<PROGRAM CODE>

#4-1) run by changing n\_estimators = 3, 7, 10, 50, 100,

#and show the accuracies of each run.

# Explain the effect of the n\_estimators.

ada\_3 = AdaBoostClassifier(n\_estimators=3)

ada\_3.fit(X\_train, Y\_train)

predic\_ada\_3 = ada\_3.predict(X\_test)

cal\_ada\_3 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ada\_3)

print("AdaBoost / accuracy when n\_estimators=3 : ",cal\_ada\_3)

ada\_7 = AdaBoostClassifier(n\_estimators=7)

ada\_7.fit(X\_train, Y\_train)

predic\_ada\_7 = ada\_7.predict(X\_test)

cal\_ada\_7 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ada\_7)

print("AdaBoost / accuracy when n\_estimators=7 : ",cal\_ada\_7)

ada\_10 = AdaBoostClassifier(n\_estimators=10)

ada\_10.fit(X\_train, Y\_train)

predic\_ada\_10 = ada\_10.predict(X\_test)

cal\_ada\_10 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ada\_10)

print("AdaBoost / accuracy when n\_estimators=10 : ",cal\_ada\_10)

ada\_50 = AdaBoostClassifier(n\_estimators=50)

ada\_50.fit(X\_train, Y\_train)

predic\_ada\_50 = ada\_50.predict(X\_test)

cal\_ada\_50 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ada\_50)

print("AdaBoost / accuracy when n\_estimators=50 : ",cal\_ada\_50)

ada\_100 = AdaBoostClassifier(n\_estimators=100)

ada\_100.fit(X\_train, Y\_train)

predic\_ada\_100 = ada\_100.predict(X\_test)

cal\_ada\_100 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ada\_100)

print("AdaBoost / accuracy when n\_estimators=100 : ",cal\_ada\_100)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

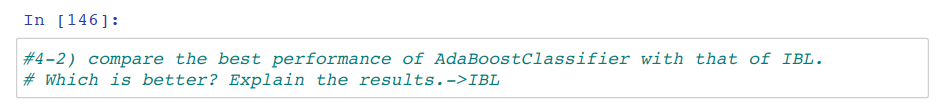
<Explain>

n\_estimators is the maximum number of estimators at which boosting is terminated. By the result, we can see the accuracy is high when n\_estimators=7, or 50. If we use 7, 50 for n\_estimators, the learning procedure is stopped early.

4-2)

<PROGRAM CODE> : None

<RESULT> : None



<Explain>

- maximum of IBL that I tested [IBL / accuracy when n\_neighbors=15 : 0.8641975308641975]

- maximum of AdaBoostClassifier that I tested [AdaBoost / accuracy when n\_estimators=50 : 0.8024691358024691]

So, IBL shows better performance of accuracy

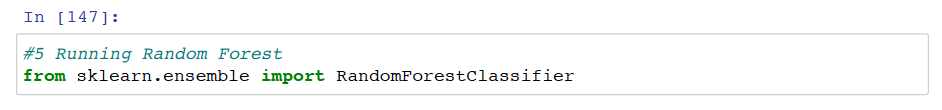
5. Running Random Forest

<PROGRAM CODE>

#5 Running Random Forest

from sklearn.ensemble import RandomForestClassifier

<RESULT> : None



5-1)

<PROGRAM CODE>

#5-1) Run by changing n\_estimators = 5, 10, 30, 50, 100, respectively,

#and show the accuracies of each run. Explain the effect of the n\_estimators.

ranf\_5 = RandomForestClassifier(n\_estimators=5,max\_depth=2, random\_state=0)

ranf\_5.fit(X\_train, Y\_train)

predic\_ranf\_5 = ranf\_5.predict(X\_test)

cal\_ranf\_5 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_5)

print("Random Forest / accuracy when n\_estimators=5 : ",cal\_ranf\_5)

ranf\_10 = RandomForestClassifier(n\_estimators=10,max\_depth=2, random\_state=0)

ranf\_10.fit(X\_train, Y\_train)

predic\_ranf\_10 = ranf\_10.predict(X\_test)

cal\_ranf\_10 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_10)

print("Random Forest / accuracy when n\_estimators=10 : ",cal\_ranf\_10)

ranf\_30 = RandomForestClassifier(n\_estimators=30,max\_depth=2, random\_state=0)

ranf\_30.fit(X\_train, Y\_train)

predic\_ranf\_30 = ranf\_30.predict(X\_test)

cal\_ranf\_30 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_30)

print("Random Forest / accuracy when n\_estimators=30 : ",cal\_ranf\_30)

ranf\_100 = RandomForestClassifier(n\_estimators=100,max\_depth=2, random\_state=0)

ranf\_100.fit(X\_train, Y\_train)

predic\_ranf\_100 = ranf\_100.predict(X\_test)

cal\_ranf\_100 = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_100)

print("Random Forest / accuracy when n\_estimators=100 : ",cal\_ranf\_100)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<Explain>

the accuracy at n\_estimators=10 shows best performance in RandomForestClassifier

5-2)

<PROGRAM CODE>

#5-2) Choose the optimal n\_estimators from q. 1),

#and run the model by changing oob\_score = True/False. respectively.

#Show the accuracies of each run, and explain the effect of the oob\_score.

#optimal n\_estimators=100

ranf\_f = RandomForestClassifier(n\_estimators=100,oob\_score=False,max\_depth=2, random\_state=0)

ranf\_f.fit(X\_train, Y\_train)

predic\_ranf\_f = ranf\_f.predict(X\_test)

cal\_ranf\_f = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_f)

print("Random Forest / accuracy when oob\_score = False : ",cal\_ranf\_f)

ranf\_t = RandomForestClassifier(n\_estimators=100,oob\_score=True,max\_depth=2, random\_state=0)

ranf\_t.fit(X\_train, Y\_train)

predic\_ranf\_t = ranf\_t.predict(X\_test)

cal\_ranf\_t = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_t)

print("Random Forest / accuracy when oob\_score = True : ",cal\_ranf\_t)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<Explain>

oob\_score is whether to use out-of-bag samples to estimate the generalization accuracy. Looking at the result, the accuracy of the data is same both oob\_score is used and not used.

5-3)

<PROGRAM CODE>

#5-3) Choose the optimal n\_estimators from q. 1),

#and run the model by changing max\_features = “auto”, “sqrt”, “log2”, respectively.

#Show the accuracies of each run, and explain the effect of the max\_features.

ranf\_a = RandomForestClassifier(n\_estimators=100, max\_features='auto',max\_depth=2, random\_state=0)

ranf\_a.fit(X\_train, Y\_train)

predic\_ranf\_a = ranf\_a.predict(X\_test)

cal\_ranf\_a = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_a)

print("Random Forest / accuracy when max\_features = “auto” : ",cal\_ranf\_a)

ranf\_s = RandomForestClassifier(n\_estimators=100, max\_features='sqrt',max\_depth=2, random\_state=0)

ranf\_s.fit(X\_train, Y\_train)

predic\_ranf\_s = ranf\_s.predict(X\_test)

cal\_ranf\_s = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_s)

print("Random Forest / accuracy when max\_features = “sqrt” : ",cal\_ranf\_s)

ranf\_l = RandomForestClassifier(n\_estimators=100, max\_features='log2',max\_depth=2, random\_state=0)

ranf\_l.fit(X\_train, Y\_train)

predic\_ranf\_l = ranf\_l.predict(X\_test)

cal\_ranf\_l = sklearn.metrics.accuracy\_score(Y\_test, predic\_ranf\_l)

print("Random Forest / accuracy when max\_features = “log2” : ",cal\_ranf\_l)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<Explain>

- max\_features is the size of the random subsets of features to consider when splitting a node. From the results, In RandomForestClassifier, the accuracy is same when we use auto, log2, and sqrt for max\_features. (Use ‘auto’ is same as use ‘sqrt’. )

If “auto”, then max\_features=sqrt(n\_features).

If “sqrt”, then max\_features=sqrt(n\_features) (same as “auto”).

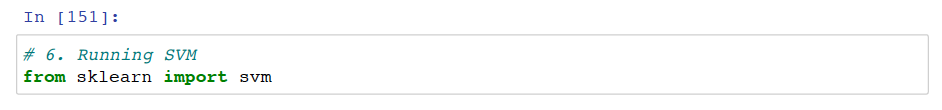
If “log2”, then max\_features=log2(n\_features).

6. Running SVM

<PROGRAM CODE>

From sklearn import svm

<RESULT> : None



6-1)

<PROGRAM CODE>

#6-1) calculate the accuracy of SVC.

svm\_ = svm.SVC()

svm\_.fit(X\_train, Y\_train)

predic\_svm\_ = svm\_.predict(X\_test)

cal\_svm\_ = sklearn.metrics.accuracy\_score(Y\_test, predic\_svm\_)

print("SVM / accuracy of SVM : ",cal\_svm\_)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

6-2)

<PROGRAM CODE>

#6-2) run SVC by changing kernel to

#‘linear’, ‘poly’, ‘rbf’, and ‘sigmoid’, and show the accuracies of each.

# Which kernel function shows the best accuracy ? => sigmoid

svm\_linear = svm.SVC(kernel='linear')

svm\_linear.fit(X\_train, Y\_train)

predic\_svm\_linear = svm\_linear.predict(X\_test)

cal\_svm\_linear = sklearn.metrics.accuracy\_score(Y\_test, predic\_svm\_linear)

print("SVM / accuracy when kernel='linear' : ",cal\_svm\_linear)

svm\_rbf = svm.SVC(kernel='rbf')

svm\_rbf.fit(X\_train, Y\_train)

predic\_svm\_rbf = svm\_rbf.predict(X\_test)

cal\_svm\_rbf = sklearn.metrics.accuracy\_score(Y\_test, predic\_svm\_)

print("SVM / accuracy when kernel='rbf' : ",cal\_svm\_rbf)

svm\_poly = svm.SVC(kernel='poly')

svm\_poly.fit(X\_train, Y\_train)

predic\_svm\_poly = svm\_.predict(X\_test)

cal\_svm\_poly = sklearn.metrics.accuracy\_score(Y\_test, predic\_svm\_)

print("SVM / accuracy when kernel='poly' : ",cal\_svm\_poly)

svm\_sigmoid = svm.SVC(kernel='sigmoid')

svm\_sigmoid.fit(X\_train, Y\_train)

predic\_svm\_sigmoid = svm\_sigmoid.predict(X\_test)

cal\_svm\_sigmoid = sklearn.metrics.accuracy\_score(Y\_test, predic\_svm\_sigmoid)

print("SVM / accuracy when kernel='sigmoid' : ",cal\_svm\_sigmoid)

<RESULT>

스크린샷이(가) 표시된 사진

자동 생성된 설명

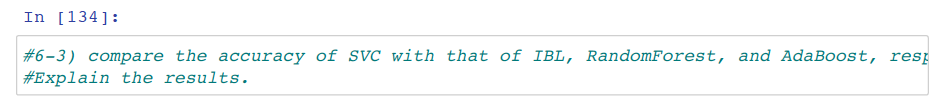
<Explain>

sigmoid kernel function shows the best accuracy.

6-3)

<PROGRAM CODE> : None

<RESULT> : None



<Explain>

- maximum of IBL that I tested [IBL / accuracy when n\_neighbors=15 : 0.8641975308641975]

- maximum of AdaBoostClassifier that I tested [AdaBoost / accuracy when n\_estimators=50 : 0.8024691358024691]

- maximum of RandomForest that I tested [Random Forest / accuracy when n\_estimators=10 : 0.8271604938271605]

- maximum of SVC that I tested [SVM / accuracy when kernel='sigmoid' : 0.8641975308641975]

So, accuracy of SVC and IBL show best performance compare to RandomForest and AdaBoostClassifier