1. (R and Python) Modify your program in Assignment #5 to do followings

a. Prompt the user whether to run regression or classification.

b. If regression is chosen, perform the linear regression as you did in Assignment #3. (You have nothing to work on the regression algorithm in this assignment).

c. If classification is chosen, ask the user the filename of the training and test dataset. (Assume the column location of the class variable is the same for both training and test dataset.)

|  |
| --- |
| **(R code)**  HW6 = function(){  cat('Checking the working environment. \n')  mylib()  cat('Checking the packages required. \n')  is.install('rgl')  is.install('maxLik')  ans = readline('Enter 1 to use Regression or 2 to use Classification : ')  if (ans==1) { regression() }  if (ans==2) {classification()}  } |

|  |
| --- |
| **(Python code)**  def HW6() :  print('Checking the working environment.')  mylib()  ans = int(input('Enter 1 to use Regression, Enter 2 to use Classification : '))  if ans == 1 :  regression\_analysis()  elif ans == 2 :  classification() |

* 이전 과제와 동일하다. (a) Prompt 창에 유저가 직접 Regression은 1을, Classification은 2를 입력하여 실행할 수 있도록 하였다.

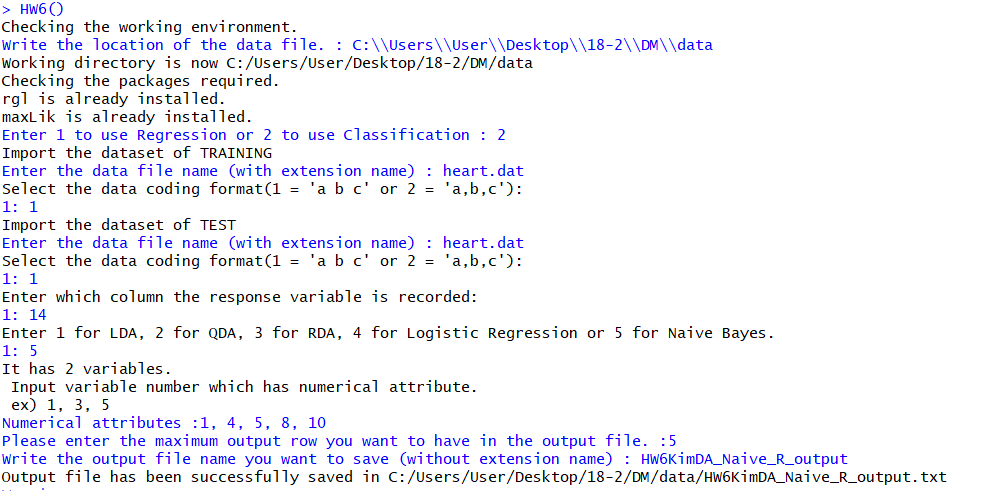
d. If classification is chosen, prompt the user to choose (i) LDA and (ii) QDA, (iii) RDA, (iv) Logistic regression, or (v) Naïve Bayes.

e. Perform (i) LDA and (ii) QDA, (iii) RDA, (iv) Logistic Regression, or (v) Naïve Bayes. depending on the choice by the user. However, if the data has more than two classes, do not prompt (iv) Logistic Regression and (v) Naïve Bayes. Use a file named “heart.dat” for the training in this assignment.

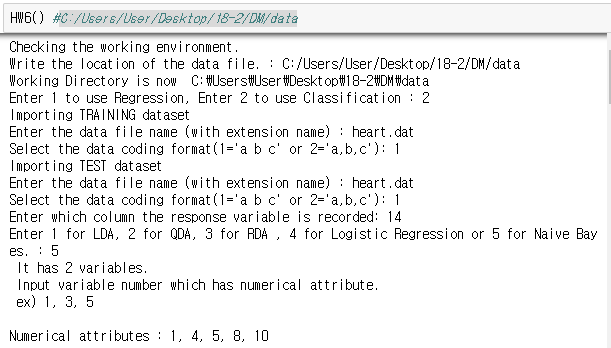
|  |
| --- |
| **(R code)**  classification <- function() {    ###################### General background #######################    # import training & test data file  cat('Import the dataset of TRAINING','\n')  train = read()  cat('Import the dataset of TEST','\n')  test = read()    # enter the Column number  cat("Enter which column the response variable is recorded: ")  num = scan(n=1, quiet=TRUE)    # nclass of response variable  k = length(unique(train[,num])) # Assume that values of the class variable are integers starting with 1    # choose (i)LDA (ii)QDA (iii)RDA (iv)Logistic Regression (v) Naive Bayes  repeat{  cat("Enter 1 for LDA, 2 for QDA, 3 for RDA, 4 for Logistic Regression or 5 for Naive Bayes.")  choice = scan(n=1, quiet=TRUE)  if (choice!=4|choice!=5|(choice==4&k==2)|(choice==5&k==2)) {break} else  {cat('If the data has more than two classes, do not implement Logistic Regression and Naive Bayes. \n', 'Please choose other method.\n')}  }  #######################  ### (v) Naive Bayes ###  #######################    if (choice == 5) {  # set variable attributes  cat('It has',k,'variables.','\n',  'Input variable number which has numerical attribute.\n',  'ex) 1, 3, 5 \n')  numer = as.numeric(strsplit(readline('Numerical attributes :'),split=",")[[1]])    # Basic vectors  n = nrow(train)  x = train[-num]  y = train[,num]  p = ncol(train)  n.t = nrow(test)  x.t = test[,-num]  y.t = test[,num]  # probability of training set  t1 = train[train[,num]==1,]  t2 = train[train[,num]==2,]  prob\_1 = matrix(1, nrow=n, ncol=p)  prob\_2 = matrix(1, nrow=n, ncol=p)    for(i in 1:n) {  for(j in c(1:p)[-num]) {  if (train[i,j]=='?') {} else {  if (any(j==numer)) {  prob\_1[i,j] <- dnorm(train[i,j], mean(t1[,j]), sd(t1[,j]))  prob\_2[i,j] <- dnorm(train[i,j], mean(t2[,j]), sd(t2[,j]))  } else {  t <- prop.table(table(y, x[[j]]), 1)  prob\_1[i,j] <- t[1,][colnames(t)==train[i,j]]  prob\_2[i,j] <- t[2,][colnames(t)==train[i,j]]  }}}}    prob\_temp <- cbind(apply(prob\_1,1,prod),apply(prob\_2,1,prod))  prob <- round(prob\_temp/rowSums(prob\_temp),3)  class <- as.matrix(apply(prob, 1, function(x) which(x==max(x))))  # probability of test set  prob\_1.t = matrix(1, nrow=n.t, ncol=p)  prob\_2.t = matrix(1, nrow=n.t, ncol=p)    for(i in 1:n.t) {  for(j in c(1:p)[-num]) {  if (test[i,j]=='?') {} else {  if (any(j==numer)) {  prob\_1.t[i,j] <- dnorm(test[i,j], mean(t1[,j]), sd(t1[,j]))  prob\_2.t[i,j] <- dnorm(test[i,j], mean(t2[,j]), sd(t2[,j]))  } else {  t <- prop.table(table(y, x[[j]]), 1)  prob\_1.t[i,j] <- t[1,][colnames(t)==test[i,j]]  prob\_2.t[i,j] <- t[2,][colnames(t)==test[i,j]]  }}}}    prob\_temp.t <- cbind(apply(prob\_1,1,prod),apply(prob\_2,1,prod))  prob.t <- round(prob\_temp.t/rowSums(prob\_temp.t),3)  class.t <- as.matrix(apply(prob.t, 1, function(x) which(x==max(x))))    # Output setting  predict = cbind(c(1:n), y, class, prob[class])  table = table(y, class, dnn=c("Actual Class","Predicted Class"))  accuracy = sum(diag(table))/sum(table)  sensi = table[2,2]/sum(table[2,])  speci = table[1,1]/sum(table[1,])    Predict.t = cbind(c(1:n.t), y.t, class.t, prob.t[class.t])  table.t = table(y.t, class.t, dnn=c("Actual Class","Predicted Class"))  accuracy.t = sum(diag(table.t))/sum(table.t)  sensi.t = table.t[2,2]/sum(table.t[2,])  speci.t = table.t[1,1]/sum(table.t[1,])  # make output file  out\_num = as.numeric(readline('Please enter the maximum output row you want to have in the output file. :' ))    outputname = readline("Write the output file name you want to save (without extension name) : ")  outputname = paste(outputname,".txt",sep="")    cat("ID, Actual class, Resub pred, Pred Prob", "\n", "-----------------------------", "\n", file = outputname, sep="")  write.table(head(predict, out\_num), outputname, sep= ", ", row.names=FALSE, col.names=FALSE, append=TRUE, quote=FALSE)  cat('(continue)','\n','\n', file = outputname, sep="", append = TRUE)  cat('Confusion Matrix (Resubstitution)', "\n", "----------------------------------", "\n",file = outputname,sep="", append=TRUE)  capture.output(print(table), file=outputname, append=TRUE)  cat("\n", "Model Summary (Resubstitution)", "\n", "------------------------------", "\n",file = outputname, sep="", append=TRUE)  cat("Overall accuracy = ", round(accuracy, 3), "\n",file = outputname, sep="", append=TRUE)  cat("Sensitivity = ", round(sensi, 3), "\n",file = outputname, sep="", append=TRUE)  cat("Specificity =", round(speci, 3), "\n\n",file = outputname, sep="", append=TRUE)    cat("ID, Actual class, Test pred, Pred Prob", "\n", "-----------------------------", "\n",file = outputname,sep="", append=TRUE)  write.table(head(Predict.t, out\_num), file=outputname, sep= ", ", row.names=FALSE, col.names=FALSE, append=TRUE, quote=FALSE)  cat('(continue)',"\n",'\n', file = outputname, sep="", append = TRUE)  cat('Confusion Matrix (Test)', "\n", "----------------------------------", "\n",file = outputname,sep="", append=TRUE)  capture.output(print(table.t), file=outputname,append=TRUE)  cat("\n", "Model Summary (Test)", "\n", "------------------------------", "\n",file = outputname,sep="", append=TRUE)  cat("Overall accuracy = ", round(accuracy.t, 3), "\n" ,file = outputname,sep="", append=TRUE)  cat("Sensitivity = ", round(sensi.t, 3), "\n",file = outputname, sep="", append=TRUE)  cat("Specificity =", round(speci.t, 3), "\n\n",file = outputname, sep="", append=TRUE)  cat("Output file has been successfully saved in ",getwd(),"/",outputname,sep="")  }  } |

|  |
| --- |
| **(Python code)**  # method choice  choice = int(input("Enter 1 for LDA, 2 for QDA, 3 for RDA , 4 for Logistic Regression or 5 for Naive Bayes. : "))  while ((choice==4&k!=2) | (choice==5$k!=2)) :  choice = int(input("Enter 1 for LDA, 2 for QDA, 3 for RDA , 4 for Logistic Regression or 5 for Naive Bayes. : "))  if ((choice != 4)|(choice!=5)) :  break  ###########################  ##### (v) Naive Bayes #####  ###########################  if choice == 5 :  import pandas as pd  from pandas import DataFrame as df  import numpy as np  from scipy.stats import norm  # import data  print('Importing TRAINING dataset')  train = read()  print('Importing TEST dataset')  test = read()  num = int(input("Enter which column the response variable is recorded: "))-1  # Basic matrixs  n = train.shape[0]  n\_t = test.shape[0]  x = train.drop(num, axis=1)  x\_t = test.drop(num, axis=1)  y = train[num]  y\_t = test[num]  p = train.shape[1]  yclass = train[num].unique()  k = len(yclass) # Assume that values of the class variable are integers starting with 1  # set variable attributes  print(' It has',k,'variables.','\n',  'Input variable number which has numerical attribute.','\n',  'ex) 1, 3, 5','\n')  numer = input('Numerical attributes : ').split(',')  numer = (np.array(numer).astype('int')-1).tolist()  numer = np.array(numer)+1  # probabiliy of training set  t1 = train[train.loc[:,num]==1]  t2 = train[train.loc[:,num]==2]  prob\_1 = np.ones((n,p))  prob\_2 = np.ones((n,p))  nnn=list(range(0,p))  nnn.remove(num)  for i in range(0,n) :  for j in nnn :  if(train.loc[i,j] != '?') :  if(j in numer) :  prob\_1[i,j] = norm.pdf(train.loc[i,j], loc=np.mean(t1.loc[:,j]), scale=np.std(t1.loc[:,j]) )  prob\_2[i,j] = norm.pdf(train.loc[i,j], loc=np.mean(t2.loc[:,j]), scale=np.std(t2.loc[:,j]) )  else :  t = pd.crosstab(y, x.loc[:,j]).apply(lambda r: r/r.sum(), axis=1)  prob\_1[i,j] = t.loc[1,train.loc[i,j]]  prob\_2[i,j] = t.loc[2,train.loc[i,j]]  prob\_temp = np.c\_[np.prod(prob\_1, axis=1), np.prod(prob\_2, axis=1) ]  prob = np.zeros((n,2))  prob[:,0] = np.round(prob\_temp[:,0]/prob\_temp.sum(axis=1),3)  prob[:,1] = np.round(prob\_temp[:,1]/prob\_temp.sum(axis=1),3)  c = df(prob, columns=[1,2]).idxmax(axis=1)  # probability of test set  prob\_1\_t = np.ones((n\_t,p))  prob\_2\_t = np.ones((n\_t,p))  for i in range(0,n\_t) :  for j in nnn :  if(test.loc[i,j] != '?') :  if(j in numer) :  prob\_1\_t[i,j] = norm.pdf(test.loc[i,j], loc=np.mean(t1.loc[:,j]), scale=np.std(t1.loc[:,j]) )  prob\_2\_t[i,j] = norm.pdf(test.loc[i,j], loc=np.mean(t2.loc[:,j]), scale=np.std(t2.loc[:,j]) )  else :  t = pd.crosstab(y, x.loc[:,j]).apply(lambda r: r/r.sum(), axis=1)  prob\_1\_t[i,j] = t.loc[1,test.loc[i,j]]  prob\_2\_t[i,j] = t.loc[2,test.loc[i,j]]  prob\_temp\_t = np.c\_[np.prod(prob\_1\_t, axis=1), np.prod(prob\_2\_t, axis=1) ]  prob\_t = np.zeros((n\_t,2))  prob\_t[:,0] = np.round(prob\_temp\_t[:,0]/prob\_temp\_t.sum(axis=1),3)  prob\_t[:,1] = np.round(prob\_temp\_t[:,1]/prob\_temp\_t.sum(axis=1),3)  prob\_t  c\_t = df(prob\_t, columns=[1,2]).idxmax(axis=1)  # Output setting  out\_num = int(input('Please enter the maximum output row you want to have in the output file. :' ))  # Crosstable  con = df({'Actual Class':y, 'Predicted Class':c})  table = pd.crosstab(con['Actual Class'],con['Predicted Class'], colnames=[''])  accuracy = np.trace(table)/n  sensi = table.loc[2,2]/sum(table.loc[2,])  speci = table.loc[1,1]/sum(table.loc[1,])  con\_t = df({'Actual Class':y\_t, 'Predicted Class':c\_t})  table\_t = pd.crosstab(con\_t['Actual Class'],con\_t['Predicted Class'], colnames=[''])  accuracy\_t = np.trace(table\_t)/n\_t  sensi\_t = table\_t.loc[2,2]/sum(table\_t.loc[2,])  speci\_t = table\_t.loc[1,1]/sum(table\_t.loc[1,])  # output file  outputname = input("Write the output file name you want to save (without extension name) : ")  outputname = outputname+'.txt'  import os as os  with open(outputname,"w") as text\_file:  print('ID, Actual class, Resub pred, Pred Prob', file=text\_file)  print('-----------------------------', file=text\_file)  for i in range(0, out\_num):  print(i+1, y[i], c[i], prob[i][c[i]-1], sep=', ', file=text\_file)  print('(continue)',file=text\_file)  print('',file=text\_file)  print('Confusion Matrix (Resubstitution)', file=text\_file)  print('----------------------------------','\n',' Predicted Class', file=text\_file)  print(table, file=text\_file)  print("",file=text\_file)  print("Model Summary (Resubstitution)", file=text\_file)  print('------------------------------', file=text\_file)  print("Overall accuracy = ", accuracy.round(3), sep='', file=text\_file)  print("Sensitivity = ", sensi.round(3), sep='', file=text\_file)  print("Specificity = ", speci.round(3), sep='', file=text\_file)  print('', file=text\_file)  print('ID, Actual class, Test pred, Pred Prob', file=text\_file)  print('-----------------------------', file=text\_file)  for i in range(0, out\_num):  print(i+1, y\_t[i], c\_t[i], prob[i][c[i]-1], sep=', ', file=text\_file)  print('(continue)',file=text\_file)  print('',file=text\_file)  print('Confusion Matrix (Test)', file=text\_file)  print('----------------------------------','\n',' Predicted Class', file=text\_file)  print(table\_t, file=text\_file)  print("",file=text\_file)  print("Model Summary (Test)", file=text\_file)  print('------------------------------', file=text\_file)  print("Overall accuracy = ", accuracy\_t.round(3), sep='', file=text\_file)  print("Sensitivity = ", sensi\_t.round(3), sep='', file=text\_file)  print("Specificity = ", speci\_t.round(3), sep='', file=text\_file)  print('', file=text\_file)  print("Output file has been successfully saved in ",os.getcwd(),"/",outputname,sep="") |

(R console)



(Python console)



* **(d)** Classification을 선택했다면, 그 후에 LDA(1), QDA(2), RDA(3), Logistic Regression(4) 그리고 Naïve Bayes(5)를 선택하여 실행하도록 하였다.
* **(e)** 그러나 class의 수가 3개 이상이면 Logistic Regression과 Naïve bayes를 실행하지 않고, 다른 방법을 선택하도록 repeat과 while함수를 이용하였다.

f. The output file for classification generated by the program must look like below.

|  |
| --- |
| **(HW5KimDA\_Logit\_R\_output)**  ID, Actual class, Resub pred, Pred Prob  -----------------------------  1, 1, 1, 0.934  2, 1, 1, 0.999  3, 1, 1, 1  4, 1, 1, 0.523  5, 1, 2, 0.956  (continue)  Confusion Matrix (Resubstitution)  ----------------------------------  Predicted Class  Actual Class 1 2  1 115 24  2 21 143  Model Summary (Resubstitution)  ------------------------------  Overall accuracy = 0.851  Sensitivity = 0.872  Specificity =0.827  ID, Actual class, Test pred, Pred Prob  -----------------------------  1, 1, 1, 0.934  2, 1, 1, 0.999  3, 1, 1, 1  4, 1, 1, 0.523  5, 1, 2, 0.956  (continue)  Confusion Matrix (Test)  ----------------------------------  Predicted Class  Actual Class 1 2  1 115 24  2 21 143  Model Summary (Test)  ------------------------------  Overall accuracy = 0.851  Sensitivity = 0.872  Specificity =0.827 |

|  |
| --- |
| **(HW5KimDA\_Logit\_py\_output)**  ID, Actual class, Resub pred, Pred Prob  -----------------------------  1, 1, 1, 0.934  2, 1, 1, 0.999  3, 1, 1, 1.0  4, 1, 1, 0.524  5, 1, 2, 0.957  (continue)  Confusion Matrix (Resubstitution)  ----------------------------------  Predicted Class  1 2  Actual Class  1 115 24  2 21 143  Model Summary (Resubstitution)  ------------------------------  Overall accuracy = 0.851  Sensitivity = 0.872  Specificity = 0.827  ID, Actual class, Test pred, Pred Prob  -----------------------------  1, 1, 1, 0.934  2, 1, 1, 0.999  3, 1, 1, 1.0  4, 1, 1, 0.524  5, 1, 2, 0.957  (continue)  Confusion Matrix (Test)  ----------------------------------  Predicted Class  1 2  Actual Class  1 115 24  2 21 143  Model Summary (Test)  ------------------------------  Overall accuracy = 0.851  Sensitivity = 0.872  Specificity = 0.827 |

* 미세한 소수점 차이를 제외하고 R과 Python 모두 동일한 output result가 나왔음을 확인할 수 있다.