1. (R and Python) Modify your program in Assignment #4 to do followings. For logistic regression, you can use any optimization function in R or Python. However, you should not use the logistic regression function directly.

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).

|  |
| --- |
| **(R code)**  HW5 = 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 HW5() :  print('Checking the working environment.')  mylib()  ans = int(input('Enter 1 to use Regression, Enter 2 to use Classification : '))  if ans == 1 :  regression()  elif ans == 2 :  # choose (i)LDA (ii)QDA (iii)RDA (iv)Logistic Regression  choice = int(input("Enter 1 for LDA, 2 for QDA, 3 for RDA or 4 for Logistic Regression."))  if choice <= 3 :  discriminant\_analysis()  elif choice == 4 :  logistic\_regression() |

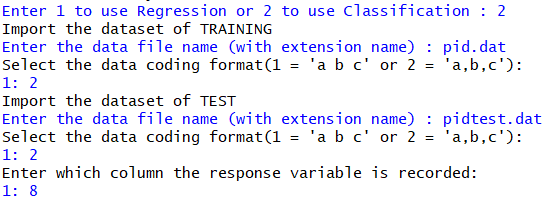
* (a) Prompt 창에 유저가 직접 Regression은 1을, Classification은 2를 입력하여 실행할 수 있도록 하였다. (b) Regression을 선택하면 Assignment1에서 만든 regression() 함수가 실행된다.

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.)

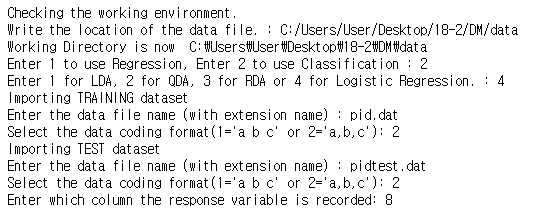
|  |
| --- |
| **(R code)**  # import training & test data file  cat('Import the dataset of TRAINING','\n')  train = read()  cat('Import the dataset of TEST','\n')  test = read()    # enter the Column number  cat("Enter which column the response variable is recorded: ")  num = scan(n=1, quiet=TRUE) |

|  |
| --- |
| **(Python code)**  print('Importing TRAINING dataset')  train = read()  print('Importing TEST dataset')  test = read()  num = int(input("Enter which column the response variable is recorded: "))-1 |

(R console)



(Python code)



* R과 Python에 read 함수를 정의하여, training dataset과 test dataset의 이름을 입력하여 직접 불러올 수 있게 하였다.

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

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

|  |
| --- |
| **(R code)**  # choose (i)LDA or (ii)QDA or (iii)RDA (iv)Logistic Regression  repeat{  cat("Enter 1 for LDA, 2 for QDA, 3 for RDA or 4 for Logistic Regression")  choice = scan(n=1, quiet=TRUE)  if (choice!=4|(choice==4&k==2)) {break} else {cat('If the data has more than two classes, do not implement Logistic Regression\n', 'Please choose other method.\n')}  }  ################################  ### (iv) Logistic Regression ###  ################################    if (choice == 4) {  # Basic matrix for Logistic  n = nrow(train)  n.t = nrow(test)  x = cbind(rep(1,n), as.matrix(train[-num]))  x.t = cbind(rep(1,n.t), as.matrix(test[-num]))  y = train[,num]  y.t = test[,num]  p = ncol(train)    library(maxLik)  B = matrix(0, nrow=(p))  loglik <- function(B) { t(y-1)%\*%x%\*%B - sum(log(1+exp(x%\*%B))) }  Lik <- maxLik(loglik, start=matrix(0, ncol=1, nrow=p))  b <- coef(Lik)  cat('The result that maximizes Log-Likelihood of Logistic Regression(MLE method).\n')  print(summary(Lik))    cat('Enter the cutoff value from 0 to 1.')  cutoff = scan(n=1, quiet=TRUE)    #Training data  prob <- round(exp(x%\*%b)/(1+exp(x%\*%b)), 3)  class <- c()  for(i in 1:n){if (prob[i]>=cutoff) {class[i] = 2} else {class[i] = 1}}    #Test data  prob.t <- round(exp(x.t%\*%b)/(1+exp(x.t%\*%b)), 3)  class.t <- c()  for (i in 1:n.t) {if(prob.t[i]>=cutoff) {class.t[i] = 2} else {class.t[i] = 1}}    # Output setting  predict = cbind(c(1:n), y, class, prob)  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)  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="")  } else warning ('Choose 1 for LDA, 2 for QDA, 3 for RDA or 4 for Logistic Regression.')  } |

|  |
| --- |
| **(Python code)**  # choose (i)LDA (ii)QDA (iii)RDA (iv)Logistic Regression  choice = int(input("Enter 1 for LDA, 2 for QDA, 3 for RDA or 4 for Logistic Regression."))  if choice <= 3 :  discriminant\_analysis()  elif choice == 4 :  logistic()  ################################  ### 3. Logistic Regression ###  ################################  def logistic\_regression() :  # import packages  import math  import pandas as pd  import numpy as np  from pandas import DataFrame as df  from scipy.optimize import minimize    # Basic matrix for Logistic  n = train.shape[0]  n\_t = test.shape[0]  ex = train.drop(num, axis=1)  x = pd.concat([pd.DataFrame(np.repeat(1,n)),ex], axis=1)  ex\_t = test.drop(num, axis=1)  x\_t = pd.concat([pd.DataFrame(np.repeat(1,n\_t)),ex\_t], axis=1)  y = train[num]  y\_t = test[num]  p = train.shape[1]    def loglike(B):  return - np.dot((y-1).T,x).dot(B) + sum(np.log(1+np.exp(np.dot(x,B))))  loglik = minimize(loglike, np.repeat(0,p))  b = loglik.x  print('The result that maximizes Log-Likelihood of Logistic Regression(MLE method).')  print(df(b))    cutoff = input('Enter the cutoff value from 0 to 1. : ')  cutoff = float(cutoff)  if not (0<=cutoff<=1) :  cutoff=0.5  print('The cutoff value must be in 0 to 1. So set 0.5 as the cutoff value.')    # Training data  prob = np.round(np.exp(np.dot(x,b))/(1+np.exp(np.dot(x,b))), 3)  c = np.repeat(0,n)  c[prob >= cutoff]=2  c[prob <= cutoff]=1  # Test data  prob\_t = np.round(np.exp(np.dot(x\_t,b))/(1+np.exp(np.dot(x\_t,b))), 3)  c\_t = np.repeat(0,n\_t)  c\_t[prob\_t >= cutoff]=2  c\_t[prob\_t <= cutoff]=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'  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], 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], 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)



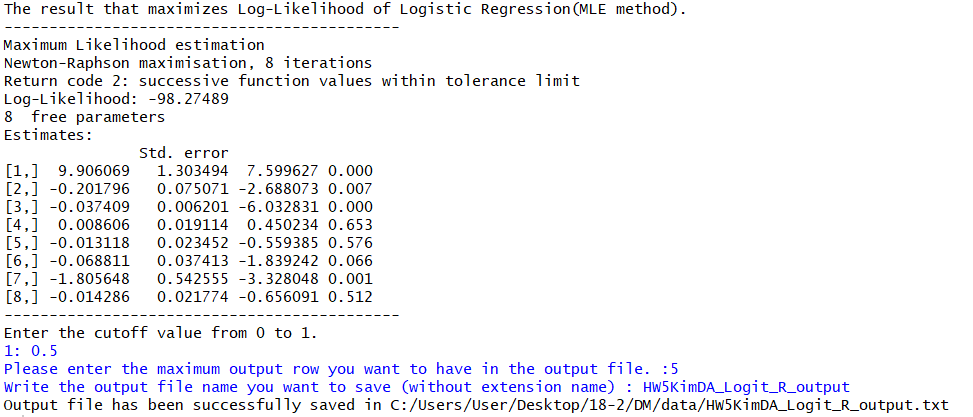
(Python console)



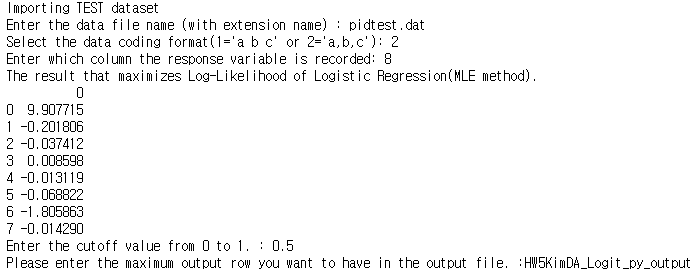
* (d) Classification을 선택했다면, 그 후에 LDA(1), QDA(2), RDA(3) 그리고 Logistic Regression(4)를 선택하여 실행하도록 하였다. (e) 그러나 class의 수가 3개 이상이면 Logistic Regression을 실행하지 않고, 다른 방법을 선택하도록 repeat과 while함수를 이용하였다.

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

**(R console)**



**(Python console)**



|  |
| --- |
| **(HW5KimDA\_Logit\_R\_output)**  ID, Actual class, Resub pred, Pred Prob  -----------------------------  1, 2, 2, 0.968  2, 2, 2, 0.889  3, 2, 2, 0.957  4, 2, 2, 0.805  5, 2, 2, 0.786  (continue)  Confusion Matrix (Resubstitution)  ----------------------------------  Predicted Class  Actual Class 1 2  1 41 31  2 14 166  Model Summary (Resubstitution)  ------------------------------  Overall accuracy = 0.821  Sensitivity = 0.922  Specificity =0.569  ID, Actual class, Test pred, Pred Prob  -----------------------------  1, 1, 2, 0.639  2, 1, 1, 0.082  3, 1, 1, 0.072  4, 1, 1, 0.238  5, 1, 2, 0.507  (continue)  Confusion Matrix (Test)  ----------------------------------  Predicted Class  Actual Class 1 2  1 43 44  2 16 123  Model Summary (Test)  ------------------------------  Overall accuracy = 0.735  Sensitivity = 0.885  Specificity =0.494 |

|  |
| --- |
| **(HW5KimDA\_Logit\_py\_output)**  ID, Actual class, Resub pred, Pred Prob  -----------------------------  1, 2, 2, 0.968  2, 2, 2, 0.889  3, 2, 2, 0.957  4, 2, 2, 0.805  5, 2, 2, 0.786  (continue)  Confusion Matrix (Resubstitution)  ----------------------------------  Predicted Class  1 2  Actual Class  1 41 31  2 14 166  Model Summary (Resubstitution)  ------------------------------  Overall accuracy = 0.821  Sensitivity = 0.922  Specificity = 0.569  ID, Actual class, Test pred, Pred Prob  -----------------------------  1, 1, 2, 0.968  2, 1, 1, 0.889  3, 1, 1, 0.957  4, 1, 1, 0.805  5, 1, 2, 0.786  (continue)  Confusion Matrix (Test)  ----------------------------------  Predicted Class  1 2  Actual Class  1 43 44  2 16 123  Model Summary (Test)  ------------------------------  Overall accuracy = 0.735  Sensitivity = 0.885  Specificity = 0.494 |

* R과 Python모두 동일한 beta 추정값과 동일한 output result가 나왔음을 확인할 수 있다.