**데이터마이닝이론**

**S T A 6 6 0 0**

**김현중 교수님**

**Homework 10**

2018314030

응용통계학과 김단아

Modify your program for Assignment #8 to do followings. You may use one of two (R or Python) language for this assignment.

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

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

regression, (v) Naïve Bayes, (vi) 1-level decision tree, or (vii) Bagging Ensemble. However, if the

data has more than two classes, do not prompt (iv), (v) and (vi).

3. For Bagging Ensemble method, use LDA as the classifier and 51 bootstraps as the number of resampled data.

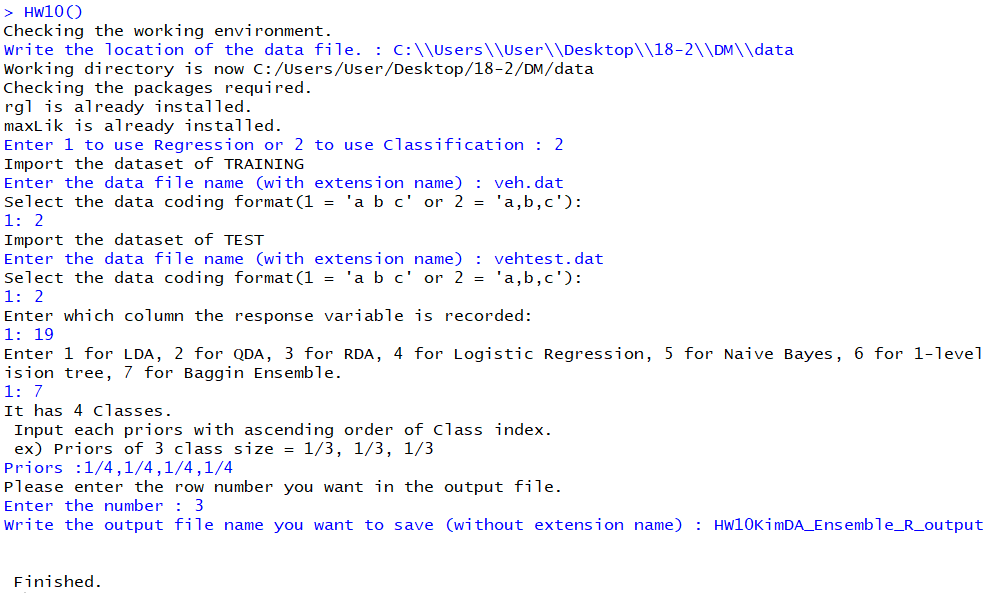
4. Use a file named “veh.dat” for the training and ‘vehtest.dat’ as the test data in this assignment.

The last column is the class variable.

5. Perform (i)-(vii) methods depending on the choice by the user. Only the output of the **test data** is necessary for 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, 5 for Naive Bayes, 6 for 1-level decision tree, 7 for Baggin Ensemble..")  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')}  }  ###################################  ### (vii) Bagging Ensemble using LDA ###  ###################################    if(choice==7) {  # set prior  cat('It has',k,'Classes.','\n',  'Input each priors with ascending order of Class index.','\n',  'ex) Priors of 3 class size = 1/3, 1/3, 1/3','\n')  prior = strsplit(readline('Priors :'),split=",")[[1]]    # Error type 1  if (length(prior)!= k) {cat('Your Prior input does not correspond with class size.','\n')  ; prior = paste(rep(1, k),'/',k,sep="")  ; cat('So equal prior is given such as', prior)}    prior = as.vector(sapply(prior, function(x){eval(parse(text=prior))}))  # Error type 2  if (length(sum(prior))!= 1) {cat('Sum of your prior input is not equal to 1','\n')  ; prior = paste(rep(1, k),'/',k,sep="")  ; cat('So equal prior is given such as', prior)}      ############## LDA - no bagging ##############  # Basic vectors  n = nrow(train)  n.t = nrow(test)  x = t(train[-num])  x.t = t((test[,-num]))  y = train[,num]  y.t = test[,num]  p = ncol(train)-1  k = length(unique(train[,num]))    means = t(as.matrix(aggregate(train[-num], train[num], mean)[,-1]))  sk = lapply(lapply(split(train,train[,num]), function(x){x[,-num]}), cov)  sp\_nonsum = lapply( sk, function(x){ (nrow(x)-1)\*x/(n-k) })  sp = Reduce('+', sp\_nonsum)  # No bagging with test data  d.nobag = t(means)%\*%solve(sp)%\*%x.t + matrix(rep(diag(-(1/2)\*t(means)%\*%solve(sp)%\*%means),each=n.t),ncol=n.t,byrow=TRUE) + matrix(rep(log(prior),each=n.t),ncol=n.t, byrow=TRUE)  c.nobag = as.matrix(apply(d.nobag, 2, function(x) which(x==max(x))))    # output  cat('Please enter the row number you want in the output file.')  out\_num = as.numeric(readline('Enter the number : '))  predict.nobag = head(cbind(c(1:n.t), y.t, c.nobag), n=out\_num)  table.nobag = table(y.t, c.nobag, dnn=c("Actual Class","Predicted Class"))  accuracy.nobag = sum(diag(table.nobag))/sum(table.nobag)    ############## LDA - bagging (n=51) ##############  id = c(1:n)  result = matrix(nrow=n.t, ncol=51)  bootstrap = c()  for(i in 1:51) {  set.seed(i\*100)  bootstrap = train[sample(id, n, replace=TRUE),]  # Basic vectors  y.b = bootstrap[,num]  means = t(as.matrix(aggregate(bootstrap[-num], bootstrap[num], mean)[,-1]))  sk = lapply(lapply(split(bootstrap, bootstrap[,num]), function(x){x[,-num]}), cov)  sp\_nonsum = lapply( sk, function(x){ (nrow(x)-1)\*x/(n-k) })  sp = Reduce('+', sp\_nonsum)  # LDA  d.temp = t(means)%\*%solve(sp)%\*%x.t + matrix(rep(diag(-(1/2)\*t(means)%\*%solve(sp)%\*%means),each=n.t),ncol=n.t,byrow=TRUE) + matrix(rep(log(prior),each=n.t), ncol=n.t, byrow=TRUE)  c.temp = as.matrix(apply(d.temp, 2, function(x) which(x==max(x))))  result[,i] = c.temp  }  c.bag = apply( result, 1, function(x) names(which.max(table(x))) )  # output  predict.bag = head(cbind(c(1:n.t), y.t, c.bag), n=out\_num)  table.bag = table(y.t, c.bag, dnn=c("Actual Class","Predicted Class"))  accuracy.bag = sum(diag(table.bag))/sum(table.bag)    # make 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, LDA-nobagging pred", "\n", "-----------------------------", "\n", file = outputname, sep="")  write.table(predict.nobag, outputname, sep= ", ", row.names=FALSE, col.names=FALSE, append=TRUE, quote=FALSE)  cat('(continue)','\n','\n', file = outputname, sep="", append = TRUE)  cat('Confusion Matrix (LDA - no bagging)', "\n", "----------------------------------", "\n",file = outputname,sep="", append=TRUE)  capture.output(print(table.nobag), file=outputname, append=TRUE)  cat("\n", "Model Summary (LDA - no bagging)", "\n", "------------------------------", "\n",file = outputname, sep="", append=TRUE)  cat("Overall accuracy: ", round(accuracy.nobag, 3), "\n\n",file = outputname, sep="", append=TRUE)  cat("ID, Actual class, LDA-bagging pred", "\n", "-----------------------------", "\n",file = outputname,sep="", append=TRUE)  write.table(predict.bag, 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 (LDA - bagging)', "\n", "----------------------------------", "\n",file = outputname,sep="", append=TRUE)  capture.output(print(table.bag), file=outputname,append=TRUE)  cat("\n", "Model Summary (LDA - bagging)", "\n", "------------------------------", "\n",file = outputname,sep="", append=TRUE)  cat("Overall accuracy: ", round(accuracy.bag, 3), "\n" ,file = outputname,sep="", append=TRUE)  } |

**(R console)**



|  |
| --- |
| **(HW10KimDA\_Ensemble(LDA)\_R\_output)**  ID, Actual class, LDA-nobagging pred  -----------------------------  1, 1, 1  2, 1, 2  3, 1, 4  (continue)  Confusion Matrix (LDA - no bagging)  ----------------------------------  Predicted Class  Actual Class 1 2 3 4  1 49 28 4 4  2 26 42 9 8  3 0 0 85 1  4 0 0 2 77  Model Summary (LDA - no bagging)  ------------------------------  Overall accuracy: 0.755  ID, Actual class, LDA-bagging pred  -----------------------------  1, 1, 1  2, 1, 2  3, 1, 4  (continue)  Confusion Matrix (LDA - bagging)  ----------------------------------  Predicted Class  Actual Class 1 2 3 4  1 50 27 4 4  2 23 44 10 8  3 0 0 85 1  4 0 1 2 76  Model Summary (LDA - bagging)  ------------------------------  Overall accuracy: 0.761 |

* Prompt 창에 유저가 직접 Regression은 1을, Classification은 2를 입력하여 실행할 수 있도록 하였다.
* LDA를 이용한 Bagging Ensemble의 결과는 위와 같다.
* Bootstrap은 51번 복원추출로 실행하였으며, 그 결과 Bagging Ensemble의 정확도가 Bagging을 하지 않은 LDA보다 조금 더 올라간 것을 확인할 수 있다.