**데이터마이닝이론**

**S T A 6 6 0 0**

**김현중 교수님**

**Homework 8**

2018314030

응용통계학과 김단아

(R or Python) Modify your program for Assignment #7 to do followings. For this assignment, use ‘titanic.csv’ file for categorical variables.

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, or (vi) 1-level decision tree. However, if the data has more than two classes, do not prompt (iv), (v) and (vi).

3. Make your program to implement (vi) 1-level decision tree only for two classes:

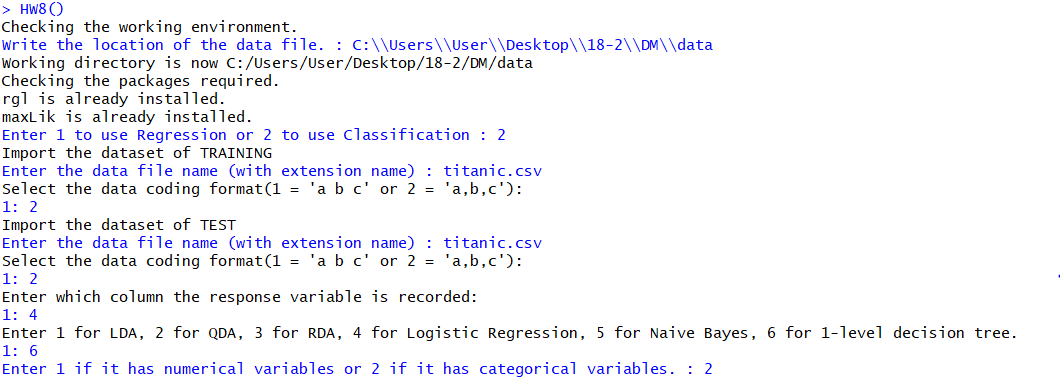
a. Find CART splitting rule, then split the current node into two subnodes. (Categorical variables should be considered in this assignment)

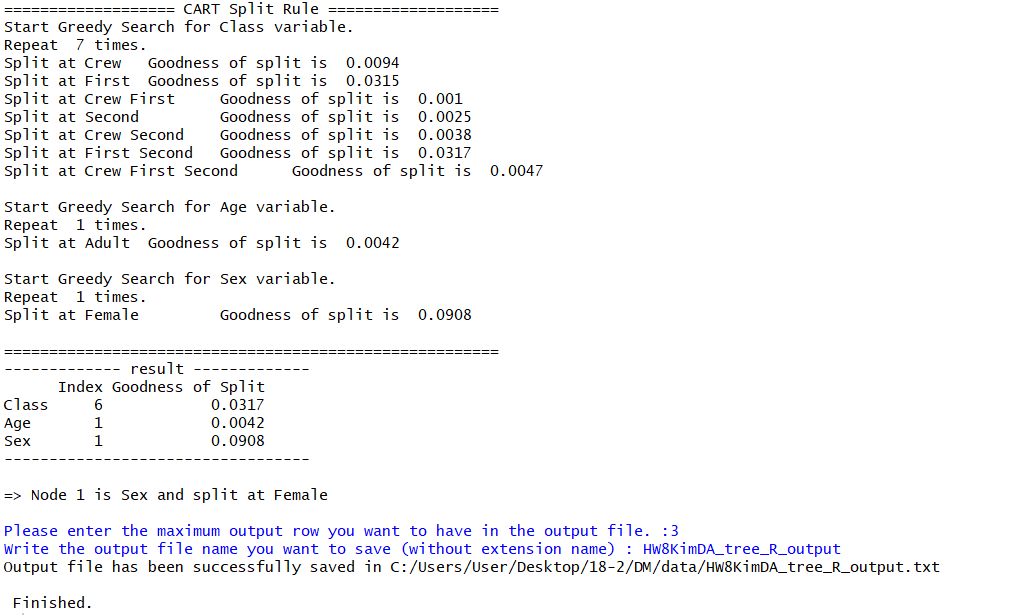
b. Print out the 1-level tree information and number of observations from each class.

4. Perform (i)-(vi) methods depending on the choice by the user.

|  |
| --- |
| **(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')}  }  ############################  ### **(vi) 1-level decision tree** ###  ############################    if (choice == 6) {    # Basic vectors  classes = sort(unique(train[,num]))  n = nrow(train)  n.t = nrow(test)  x = train[-num]  x.t = test[,-num]  y = train[,num]  y.t = test[,num]  p = ncol(train)-1    # Data type  type = readline("Enter 1 if it has numerical variables or 2 if it has categorical variables. : ")      **# Categorical variables**  else if (type == 2) {  # Greedy search for categorical variable  GoS = matrix(ncol=2, nrow=p)  s = c()    cat('=================== CART Split Rule ===================', '\n')  for (j in 1:p) {  # Xi variable  value = sort(unique(x[,j]))  c = length(value)  combi = expand.grid(rep(list(0:1), c))  t = nrow(combi)/2  goodness <- c()  cat('Start Greedy Search for', colnames(train)[-num][j], 'variable.','\n')  cat('Repeat ',t-1 ,'times.','\n')    for (i in 1:(t-1)) {  # 2^(c-1)-1 time repetition  s = value[as.numeric(combi[i+1,])\*c(1:c)]  cat('Split at', as.character(s), '\t')  # split  t1 = subset(y, x[,j] %in% s) ; t2 = subset(y, !(x[,j] %in% s))  n1 = length(t1) ; n2 = length(t2)    #Gini impurity and Goodness of split  imp\_t1 <- 1-(sum(t1==classes[1])/n1)^2-(sum(t1==classes[2])/n1)^2  imp\_t2 <- 1-(sum(t2==classes[1])/n2)^2-(sum(t2==classes[2])/n2)^2  imp\_t <- 1-(sum(y==classes[1])/n)^2-(sum(y==classes[2])/n)^2  goodness[i] <- imp\_t - n1/n\*imp\_t1 - n2/n\*imp\_t2  cat('Goodness of split is ', round(goodness[i], 4), '\n')  }  GoS[j,] = c(which.max(goodness), round(max(goodness),4))  cat('\n')  }  cat('=======================================================', '\n')    # result of split point and spliting  result <- GoS  rownames(result) <- c(colnames(train)[-num])  colnames(result) <- c('Index','Goodness of Split')  cat('------------- result -------------', '\n')  print(result)  cat('----------------------------------', '\n', '\n')    j = which.max(result[,2])  var\_1 = rownames(result)[j]  t = result[j,1]    value = sort(unique(x[,j]))  c = length(value)  combi = expand.grid(rep(list(0:1), c))  split\_1 = as.character(value[as.numeric(combi[t+1,])\*c(1:c)])  cat('=> Node 1 is', var\_1, 'and split at',split\_1, '\n', '\n')    # training set  t1 = subset(y, x[,j] %in% split\_1); t2 = subset(y, !(x[,j] %in% split\_1))  n1 = length(t1) ; n2 = length(t2)  c1 = as.character(classes[which.max(table(t1))]) ; c2 = as.character(classes[which.max(table(t2))])  class = rep(0,n)  class[x[,j] %in% split\_1] <- c1  class[!(x[,j] %in% split\_1)] <- c2    # test set  class.t = rep(0,n.t)  class.t[x.t[,j] %in% split\_1] <- c1  class.t[!(x.t[,j] %in% split\_1)] <- c2    # Output setting  predict = cbind(c(1:n), as.character(y), 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), as.character(y.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("Tree Structure", "\n", file = outputname, sep="")  cat("\t", 'Node 1: ', var\_1, ' in {', split\_1, '} (', table(y)[1], ',', table(y)[2], ')', '\n', file = outputname, sep="", append=TRUE)  cat("\t", 'Node 2: ', c1, ' (', table(t1)[1], ',', table(t1)[2], ')', '\n', file = outputname, sep="", append=TRUE)  cat("\t", 'Node 3: ', c2, ' (', table(t2)[1], ',', table(t2)[2], ')', '\n','\n', file = outputname, sep="", append=TRUE)    cat("ID, Actual class, Resub pred", "\n", "-----------------------------", "\n", file = outputname, sep="", append=TRUE)  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", "\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="") }  } |

**(R console)**





|  |
| --- |
| **(HW8KimDA\_Tree\_R\_output)**  Tree Structure  Node 1: Sex in {Female} (1490,711)  Node 2: Yes (126,344)  Node 3: No (1364,367)  ID, Actual class, Resub pred  -----------------------------  1, Yes, No  2, Yes, No  3, Yes, No  (continue)  Confusion Matrix (Resubstitution)  ----------------------------------  Predicted Class  Actual Class No Yes  No 1364 126  Yes 367 344  Model Summary (Resubstitution)  ------------------------------  Overall accuracy = 0.776  Sensitivity = 0.484  Specificity = 0.915  ID, Actual class, Test pred  -----------------------------  1, Yes, No  2, Yes, No  3, Yes, No  (continue)  Confusion Matrix (Test)  ----------------------------------  Predicted Class  Actual Class No Yes  No 1364 126  Yes 367 344  Model Summary (Test)  ------------------------------  Overall accuracy = 0.776  Sensitivity = 0.484  Specificity = 0.915 |

* Prompt 창에 유저가 직접 Regression은 1을, Classification은 2를 입력하여 실행할 수 있도록 하였다.
* 1-level decision tree를 위해 6을 입력하고, categorical variable 2 를 입력해준다.
* CART split rule을 사용하여 node 1과 spilt point를 찾아주었다. Greedy Search의 결과를 R console창에서 확인할 수 있다.
* Titanic.csv를 사용한 1-level decition tree의 output은 위와 같다.