HW 08

TUAN BUI

11/16/2021

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✓ ggplot2 3.3.5 ✓ purrr 0.3.4  
## ✓ tibble 3.1.4 ✓ dplyr 1.0.7  
## ✓ tidyr 1.1.3 ✓ stringr 1.4.0  
## ✓ readr 2.0.1 ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(kernlab)

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':  
##   
## cross

## The following object is masked from 'package:ggplot2':  
##   
## alpha

spambase\_data <- read.table('~/OneDrive - Stony Brook University/SBU/MAT + AMS/Fall 2021/AMS 380/hw/08/spambase.data', sep = ",")  
  
spambase\_names <- read.delim('~/OneDrive - Stony Brook University/SBU/MAT + AMS/Fall 2021/AMS 380/hw/08/spambase.names')  
  
spambase\_names <- spambase\_names[-c(1:29),]  
spambase\_names <- as.data.frame(spambase\_names)  
spambase\_names <- spambase\_names %>%  
 separate(spambase\_names, c("Variable", "Type"), sep = ":")   
names(spambase\_data) <- spambase\_names$Variable  
names(spambase\_data)[58] <- 'class'  
  
# clean data  
spambase\_data <- na.omit(spambase\_data)  
  
spambase\_data$class <- as.factor(spambase\_data$class)

# Question 01:

# Use the random seed 123 to divide the cleaned data into 75% training and 25% testing  
set.seed(123)  
training.samples <- spambase\_data$class %>%   
 createDataPartition(p = 0.75, list = FALSE)  
train.data <- spambase\_data[training.samples, ]  
test.data <- spambase\_data[-training.samples, ]

# Question 02:

set.seed(123)  
model <- train(  
 class ~., data = train.data, method = "rf",  
 trControl = trainControl("cv", number = 10),  
 importance = TRUE  
 )  
  
predicted.classes\_1 <- model %>% predict(train.data)  
  
# Confusion matrix of the training data   
table(predicted.classes\_1, train.data$class)

##   
## predicted.classes\_1 0 1  
## 0 2091 2  
## 1 0 1358

mean(predicted.classes\_1 == train.data$class)

## [1] 0.9994205

# The overall accuracy of the training data is 0.9994205  
  
sum((train.data$class == 1)\*(predicted.classes\_1 == 1))/sum(train.data$class ==  
1)

## [1] 0.9985294

# The sensitivity of the test data is 0.9985294  
  
sum((train.data$class == 0)\*(predicted.classes\_1 == 0))/sum(train.data$class ==  
0)

## [1] 1

# The specificity of the test data is 1

# Question 03:

predicted.classes\_2 <- model %>% predict(test.data)  
  
# Confusion matrix of the testing data  
table(predicted.classes\_2, test.data$class)

##   
## predicted.classes\_2 0 1  
## 0 674 34  
## 1 23 419

mean(predicted.classes\_2 == test.data$class)

## [1] 0.9504348

# The overall accuracy of the testing data is 0.9504348  
  
sum((test.data$class == 1)\*(predicted.classes\_2 == 1))/sum(test.data$class ==  
1)

## [1] 0.9249448

# The sensitivity of the testing data is 0.9249448  
  
sum((test.data$class == 0)\*(predicted.classes\_2 == 0))/sum(test.data$class ==  
0)

## [1] 0.9670014

# The specificity of the testing data is 0.9670014

# Question 04:

# Plot MeanDecreaseAccuracy  
varImpPlot(model$finalModel, type = 1)

Chart, line chart

Description automatically generated

# Plot MeanDecreaseGini  
varImpPlot(model$finalModel, type = 2)

Graphical user interface, chart, table, Excel

Description automatically generated

# Question 05:

# The importance of each variable in percentage based on MeanDecreaseAccuracy  
varImp(model$finalModel, type = 1)

## Overall  
## word\_freq\_make 8.153946  
## word\_freq\_address 9.367384  
## word\_freq\_all 6.902195  
## word\_freq\_3d 11.807186  
## word\_freq\_our 36.103842  
## word\_freq\_over 22.307910  
## word\_freq\_remove 79.892670  
## word\_freq\_internet 16.843599  
## word\_freq\_order 12.148703  
## word\_freq\_mail 12.358148  
## word\_freq\_receive 12.781801  
## word\_freq\_will 17.058283  
## word\_freq\_people 8.937479  
## word\_freq\_report 8.684960  
## word\_freq\_addresses 6.577593  
## word\_freq\_free 56.136717  
## word\_freq\_business 20.560001  
## word\_freq\_email 13.563499  
## word\_freq\_you 17.255713  
## word\_freq\_credit 8.753119  
## word\_freq\_your 26.697597  
## word\_freq\_font 26.977270  
## word\_freq\_000 28.440091  
## word\_freq\_money 27.628663  
## word\_freq\_hp 57.245508  
## word\_freq\_hpl 21.207529  
## word\_freq\_george 36.460092  
## word\_freq\_650 17.305336  
## word\_freq\_lab 12.332069  
## word\_freq\_labs 10.002784  
## word\_freq\_telnet 5.097685  
## word\_freq\_857 6.688856  
## word\_freq\_data 5.389617  
## word\_freq\_415 5.611032  
## word\_freq\_85 11.228468  
## word\_freq\_technology 16.113064  
## word\_freq\_1999 22.059059  
## word\_freq\_parts 3.843755  
## word\_freq\_pm 12.666298  
## word\_freq\_direct 4.832203  
## word\_freq\_cs 5.346080  
## word\_freq\_meeting 20.748908  
## word\_freq\_original 7.560766  
## word\_freq\_project 6.885084  
## word\_freq\_re 26.956730  
## word\_freq\_edu 56.642240  
## word\_freq\_table 1.073460  
## word\_freq\_conference 10.188196  
## `char\_freq\_;` 16.495985  
## `char\_freq\_(` 16.824041  
## `char\_freq\_[` 6.488239  
## `char\_freq\_!` 73.894087  
## `char\_freq\_$` 51.642349  
## `char\_freq\_#` 7.792698  
## capital\_run\_length\_average 47.378136  
## capital\_run\_length\_longest 35.917454  
## capital\_run\_length\_total 26.784196

# Question 6:

mtry <- 26  
p <- mtry\*3  
p

## [1] 78

# The number of variables we should (as commonly recommended) to select, at random, to be considered for that node split are 78