mt_v6

Oscar Monroy

11/7/2020

```r
testing <- read.csv("test.csv")

library(dplyr)

training <- read.csv("training.csv")

# Scale and center training data (except id and class)
sc_training <- training %>%
  select(-id, -class) %>%
  mutate_all(scale)

library(leaps)
library(caret)
library(MASS)
library(tidyverse)

training <- read_csv("training.csv")

# Select rows that contain Z scores less than -10 or greater than 10
outliers_inds <- unique(which(abs(sc_training) > 10, arr.ind = TRUE)[, "row"])
# Set outliers aside in separate data set
outliers <- training[outliers_inds,]
# Remove outliers from training data
training1 <- training[-outliers_inds,]

# Turn response variable into factor
training1 <- data.frame(training1[, -99], "class" = as.factor(training1$class))
# Remove duplicate row. id 3013 is just a duplicate of 520
training1 <- training1[-2935, ]
```

```r
library(ggplot2)

# Box plots of each variable (7 per plot) split by class
for (i in seq(1, ncol(training1) - 1, by = 7)) {
  # Select 7 columns plus class
  plot <- training1[, c(names(training)[seq(i, length.out = 7)], "class")] %>%
    # Scale and center numeric variables
    mutate_if(is.numeric, scale) %>%
    # Turns 7 columns into 2 columns: variable name and value
    gather(-class, key = "var", value = "value") %>%
    ggplot(aes(x = var, y = value, color = class)) +
      geom_boxplot(outlier.shape = 1) +
      theme(axis.text.x = element_text(angle = 20)) +
      coord_cartesian(ylim = c(-10, 10))

  print(plot)
}
```

```r
removal_kfold_qda <- function(train, kfold) {
  # List to store average error rates in a list where each element represents
  # a removed variable.
  lst <- list(0)
  errors <- rep(lst, dim(train)[2] - 1)

  # Here we'll run 3 for loops to run k-fold of 5 and calculate the mean error 100 times
  # per variable to determine if removing that variable imprives the model in any way.

  # The first loop with j as the index will be running through each variable in our current
  # training data
  for(j in 1:(dim(train)[2]-1)) {
    # Use "j" as index to remove a variable in each run
    rmv_sbst <- c(j)
    training_d <- train[, -rmv_sbst]

    # The second loop with k index will allow k-folds of 5 to ran 15 times at each iteration.
    # At the end of each iteration, the average of the error rate is taken and stored
    for(k in 1:15) {
      # Empty vector to store errors from qda and lda models
      error_qda <- error_lda <- rep(0, 5)
      # k-fold CV, k = 5
      kfolds <- createFolds(training_d[, dim(training_d)[2]], k = kfold)

      # The third loop with i index will run the k-folds through each of the 5 subsets created
      # using the createFolds function. At the end of each iteration, the average of the
      # folds is taken and stored in error qda
      for(i in 1:kfold) {
        # Here we subset the data by test and training sets
        # The current index will determine the current fold of subsets for the test set
        testset <- training_d[kfolds[[i]], ]
        trainingset <- training_d[-kfolds[[i]], ]

        # Subset with only the response variable
        test_class <- testset$class

        # qda model found
        qda_m <- qda(class ~ ., data = trainingset)

        # Predictions
        qda_pb <- predict(qda_m, testset)

        # Calculation of error rate
        error_qda[i] <- mean(qda_pb$class != test_class)
      }
      # The average of the errors of each of the folds is stored here in errors
      errors[[j]][k] <- mean(error_qda)
    }
  }

  # We'll take the overall averages of the average error rates
  lapply(errors, mean)
}

addition_kfold_qda <- function(main_t, train, kfold) {
  lst <- list(0)
  errors2 <- rep(lst, dim(main_t)[2])

  # Here we'll run 3 for loops to run k-fold of 5 and calculate the mean error 100 times
  # per variable to determine if -adding- a variable improves the model in any way.

  # The first loop with j as the index will run through each variable in the set t1 in order
  # to add that variable to our current training set
  for(j in 1:(dim(main_t)[2])) {
    # Use "j" as index to add a variable in each run
    add_sbst <- j
    training_d <- data.frame(main_t[, add_sbst], train)

    # Similarly to the loops where we remove a variable, this also run k-folds of five 15 times
    # At the end of each iteration, the average of the 5 folds is taken and stored in errors
    for(k in 1:15) {
      # Empty vector to store errors from qda and lda models
      error_qda <- error_lda <- rep(0, 5)
      # k-fold CV, k = 5
      kfolds <- createFolds(training_d[, dim(training_d)[2]], k = kfold)

      # Also similar to the variable removal loops, thise for loop runs through each fold's sebset
      # and then then calcaulate the error rate at the end and stores it.
      for(i in 1:kfold) {
        # Here we subset the data by test and training sets
        testset <- training_d[kfolds[[i]], ]
        trainingset <- training_d[-kfolds[[i]], ]

        # Subset with only the response variable
        test_class <- testset$class

        # Regression methods on first data set
        qda_m <- qda(class ~ ., data = trainingset)

        # Predictions
        qda_pb <- predict(qda_m, testset)

        # Calculation of error rate
        error_qda[i] <- mean(qda_pb$class != test_class)
      }
      # Storage of the mean of the average error rate of each whole k-fold loop
      errors2[[j]][k] <- mean(error_qda)
    }
    errors2[j]
  }

  names(errors2) <- names(main_t)
  lapply(errors2, mean)
}
```

```
# We'll add new variables that can hopefully predict the classes better.
# THe variables were picked from the boxplots where I determine if they appear
# to be good predictors,
sbst2 <- c(2, 4, 5, 8, 9, 13, 15, 18, 24, 26, 30, 32, 38, 39, 42, 44, 45, 46, 47, 51, 55, 56, 57, 63, 6
           68, 69, 70, 71, 72,  73, 74, 75, 76, 77, 78, 80, 93, 95, 96, 98, 99)
training3 <- training1[, sbst2]

set.seed(10)
error_qda <- error_lda <- rep(0, 5)
# k-fold CV, k = 5
kfolds <- createFolds(training3$class, k = 5)

# Single run of 5-folds to calculate error rate of current model
for(i in 1:5) {
  # Here we subset the data by test and training sets
  testset <- training3[kfolds[[i]], ]
  trainingset <- training3[-kfolds[[i]], ]

  train_class <- trainingset$class
  test_class <- testset$class

  # Regression methods on first data set
  qda_m <- qda(class ~ ., data = trainingset)
  lda_m <- lda(class ~ ., data = trainingset)

  # Predictions
  qda_pb <- predict(qda_m, testset)
  lda_pb <- predict(lda_m, testset)

  # Calculation of error rate
  error_qda[i] <- mean(qda_pb$class != test_class)
  error_lda[i] <- mean(lda_pb$class != test_class)
}

# We can now see a model with some strong potential, particularly with the qda model
error_qda
error_lda
mean(error_qda)
```

```
# We'll run some code where we'll remove variables from the current
# training set iteratively and determine which removed variable
# improves the model with a lower avaerage error rate

set.seed(8)

# Here we'll run the removal_kfold_qda function to see if we should remove variables
```

```r
a1 <- removal_kfold_qda(train = training3, kfold = 5)
# Turns list of average error rates into a vector
a1 <- unlist(a1)
# Assigns the names of the variables that were removed to the correct place
names(a1) <- names(training3[-dim(training3)[2]])
# Create boxplot to see the range of average error rates
boxplot(a1, main = "Range of Error Rates - Removal", ylab = "Average Error Rates")
# Generate summary statistics about the average error rates
summary(a1)
# Shows the index of the minimum value as well as the value
print(c(which.min(a1), min(a1)))
# Shoes the three most minimal values
head(sort(a1), 3)

# After running the for loop several times, we can see removing a few
# variables would improve the model by a bit
```

```r
# Here I'll pick the 3 variables with the lowest error rates when said variables are removed
# For example: variable 28 (Broad_H3K4me2_percentage), when removed,
# gives an error rate of 0.08091007, an improvemnt
# over most other removed variables. Likewise, variable 39 (H3K79me2_height) and variable 32 (H3K4me1_h
# Now we remove those variables mentioned above.

sbst3 <- sbst2[-c(28, 32, 39)]
training3_5 <- training1[, sbst3]

error_qda <- error_lda <- rep(0, 5)
# k-fold CV, k = 5
kfolds <- createFolds(training3_5$class, k = 5)

# Single run of 5-folds to calculate error rate of current model
for(i in 1:5) {
  # Here we subset the data by test and training sets
  testset <- training3_5[kfolds[[i]], ]
  trainingset <- training3_5[-kfolds[[i]], ]

  test_class <- testset$class

  # Regression methods on first data set
  qda_m <- qda(class ~ ., data = trainingset)

  # Predictions
  qda_pb <- predict(qda_m, testset)

  # Calculation of error rate
  error_qda[i] <- mean(qda_pb$class != test_class)
  print(table("true class" = test_class, "predicted" = as.factor(qda_pb$class)))
}
error_qda
error_lda
mean(error_qda)
# Although not perfect, we can see an improvement using this confusion matrix
# table("true class" = test_class, "predicted" = as.factor(qda_pb$class))
set.seed(777) # seed of 777 because that's a lucky number]
test4 <- testing[, sbst3]

qda_m <- qda(class ~ ., data = training3_5)

qda_p <- predict(qda_m, test4)

resultqda <- data.frame("id" = testing$id, "class" = qda_p$class)

# Code used to create pred_version8_5
```

```r
# Now, I'll be adding one variable (from the main dataset) or removing one variable via for loops and
# depending which method improves the error rate, I'll choose that.

# We create a subset from the original training set without the id and class variables, as well as
# without variables already being used by our current training data
t1 <- training1[, -c(1, sbst3, dim(training1)[2])]

# We remove these variables since they create a "rank deficiency" error
t2 <- t1[, -c(15, 24, 27, 30, 42)]

set.seed(777)

a2 <- addition_kfold_qda(main_t = t2, train = training3_5, kfold = 5)
boxplot(unlist(a2), main = "Range of Error Rates - Addition", ylab = "Average Error Rates")
summary(unlist(a2))
print(c(which.min(unlist(a2)), min(unlist(a2))))
head(sort(unlist(a2)), 3)

# We can see adding the variable MGAentropy improves the model the most
```

```r
# We now add MGAentropy
sbst4 <- sort(c(36, sbst3))
training4 <- training1[, sbst4]

# We can now run another variable removal loop to test which variable we don't need

set.seed(1)
a3 <- removal_kfold_qda(training4, kfold = 5)
# Turns list of average error rates into a vector
a3 <- unlist(a3)
# Assigns the names of the variables that were removed to the correct place
names(a3) <- names(training4[, -dim(training4)[2]])
boxplot(a3, main = "Range of Error Rates - Removal", ylab = "Average Error Rates")
summary(a3)
print(c(which.min(a3), min(a3)))
head(sort(unlist(a3)), 3)

# Variable 17 (Super_Enhancer_percentage) can be removed
```

```r
training4_5 <- training4[, -17]

set.seed(80)
```

```r
test5 <- testing[, sbst4]
test5 <- test5[, -17]

qda_m <- qda(class ~ ., data = training4_5)

qda_p <- predict(qda_m, test5)

resultqda <- data.frame("id" = testing$id, "class" = qda_p$class)

# Code used to create pred_version9
```

```r
# Now, I'll be removing one variable or adding one variable (from the main dataset) via for loops and
# depending which method improves the error rate, I'll choose that.

t2 <- training1[, -c(1, sbst4, dim(training1)[2])]
# We remove these indices since they create a "rank deficiency" error
t2_5 <- t2[, -c(15, 23, 26, 29, 40)]

set.seed(7)
a4 <- addition_kfold_qda(main_t = t2_5, train = training4_5, kfold = 5)
boxplot(unlist(a4), main = "Range of Error Rates - Addition", ylab = "Average Error Rates")
summary(unlist(a4))
print(c(which.min(unlist(a4)), min(unlist(a4))))
head(sort(unlist(a4)), 3)

# We can add Missense_TO_Silent_Ratio
```

```r
sbst5 <- sort(c(11, sbst4))

training5 <- training1[, sbst5]

test6 <- testing[, sbst5]

qda_m <- qda(class ~ ., data = training5)

qda_p <- predict(qda_m, test6)

resultqda <- data.frame("id" = testing$id, "class" = qda_p$class)

# Code used to create pred_version10
```

```r
# We can now run another variable removal loop to test which variable we don't need

set.seed(777)
a5 <- removal_kfold_qda(train = training5, kfold = 5)
a5 <- unlist(a5)
names(a5) <- names(training5[, -dim(training5)[2]])
boxplot(a5, main = "Range of Error Rates - Removal", ylab = "Average Error Rates")
summary(a5)
print(c(which.min(a5), min(a5)))
head(sort(a5), 3)

# We can remove the 28th variable as seen with the score
# We'll try out the model without the 28th variable now
# sbst5_5 <- c(2, 4, 5, 8, 9, 11, 13, 15, 18, 24, 26, 30, 32, 36, 38, 39, 42,
# 44, 45, 46, 47, 51, 55, 56, 57, 63, 66, 69, 71, 72, 73, 75, 76, 77, 78, 80, 93, 96, 98, 99)
sbst5_5 <- sbst5[-28]
training5_5 <- training1[, sbst5_5]

test6_5 <- testing[, sbst5_5]

qda_m <- qda(class ~ ., data = training5_5)

qda_p <- predict(qda_m, test6_5)

resultqda <- data.frame("id" = testing$id, "class" = qda_p$class)

# Use your own directory
write.csv(resultqda, "pred_version12.csv", row.names = FALSE) # Last prediction
```