Chapter 5: Resampling Method Homework

Bijesh Mishra

03/22/2021

Chapter 5: Resampling Method Homework

```
rm(list = ls())
setwd("~/Dropbox/OSU/PhD/SemVISp2021/STAT5063ML/Homeworks/hw5")
library(MASS)
library(boot)
library(ISLR)
library(class)
library(readxl) #read excel.
studentdata2019 = read_excel("//Users//bmishra//Dropbox//OSU//PhD//SemVISp2021//STAT5063ML//Data//Stude
hw5.data = setNames(studentdata2019,
                    tolower(names(studentdata2019))) #lower case names.
attach(hw5.data, pos = 2L,
       warn.conflicts = F)
# names(hw5.data)
as.data.frame(hw5.data[c(4,36),])
```

```
##   gender    class hsclass txtsent txtrec fbtime pinterest snapchat introvert
## 1      F STAT2023     123      18     28     45         N        Y         4
## 2      M STAT5063      36      10     20      5         Y        N         4
##   year
## 1 2016
## 2 2019
```

Sampling Procedure: LDA, QDA, KNN

Q1.A:

```
# Full LDA Model: Trianing Error
set.seed(1)
lda.fm = lda(pinterest ~ txtsent + txtrec +
               hsclass + fbtime + introvert,
             data = hw5.data) # LDA Full Model

# LDA FM Training error rate
ldafm.trr = mean((predict(lda.fm,
                          newdata = hw5.data))$class != pinterest)
ldafm.trr # LDA FM Training error rate
```

```
## [1] 0.2553191
```

```
# Full LDA Model: Test Error
set.seed(1)
lda.fmCV = lda(pinterest ~ txtsent +
                 txtrec + hsclass + fbtime + introvert,
```

```
             data = hw5.data,
             CV = TRUE) # LDA Full Model + LOOCV.
ldafm.CV = mean(lda.fmCV$class != pinterest) # LDA FM Test error rate.
ldafm.CV # LDA FM Test error rate.
```

```
## [1] 0.3191489
```

```
# Partial LDA Model: Training Error
set.seed(1)
lda.pm = lda(pinterest ~ txtsent,
             data = hw5.data) #LDA Partial Model
ldapm.trr = mean((predict(lda.pm,
                          newdata = hw5.data))$class != pinterest) # LDA PM Training error rate.
ldapm.trr # LDA PM Training error rate.
```

```
## [1] 0.3404255
```

```
# Partial LDA Model: Test Error
set.seed(1)
lda.pmCV = lda(pinterest ~ txtsent,
               data = hw5.data,
               CV = TRUE) #LDA Partial Model + LOOCV.
ldapm.CV = mean(lda.pmCV$class != pinterest) # LDA PM Test error rate.
ldapm.CV # LDA PM Test error rate.
```

```
## [1] 0.3404255
```

```
# Full QDA Model: Trianing Error
set.seed(1)
qda.fm = qda(pinterest ~ txtsent +
               txtrec + hsclass + fbtime + introvert,
             data = hw5.data) # QDA Full Model
qdafm.trr = mean((predict(qda.fm,
                          newdata = hw5.data))$class != pinterest) # QDA FM Training error rate
qdafm.trr # QDA FM Training error rate
```

```
## [1] 0.3617021
```

```
# Full QDA Model: Test Error
set.seed(1)
qda.fmCV = qda(pinterest ~ txtsent +
                 txtrec + hsclass + fbtime + introvert,
               data = hw5.data,
               CV = TRUE) # QDA Full Model + LOOCV.
qdafm.CV = mean(qda.fmCV$class != pinterest) # QDA FM Test error rate.
qdafm.CV # QDA FM Test error rate.
```

```
## [1] 0.5106383
```

```
# Partial QDA Model: Training Error
set.seed(1)
qda.pm = qda(pinterest ~ txtsent,
             data = hw5.data) # QDA Partial Model
qdapm.trr = mean((predict(qda.pm,
                          newdata = hw5.data))$class != pinterest) # QDA FM Training error rate
qdapm.trr # QDA PM Training error rate.
```

```
## [1] 0.3404255
```

```
# Partial QDA Model: Test Error
set.seed(1)
qda.pmCV = qda(pinterest ~ txtsent,
               data = hw5.data,
               CV = TRUE) # QDA Partial Model + LOOCV.
qdapm.CV = mean(qda.pmCV$class != pinterest) # QDA PM Test error rate.
qdapm.CV # QDA PM Test error rate.
```

```
## [1] 0.3404255
```

```
#Table from above results:
Predictors = c("Full Model", "Text Sent Only")
LDA.Train = round(c(ldafm.trr,
              ldapm.trr),3)
LDA.Test = round(c(ldafm.CV,
                   ldapm.CV),
                 3)
QDA.Train = round(c(qdafm.trr,
              qdapm.trr), 3)
QDA.Test = round(c(qdafm.CV,
                   qdapm.CV),
                 3)
#Table:
as.data.frame(cbind(Predictors,
                    LDA.Train,
                    LDA.Test,
                    QDA.Train,
                    QDA.Test))
```

```
##       Predictors LDA.Train LDA.Test QDA.Train QDA.Test
## 1     Full Model     0.255    0.319     0.362    0.511
## 2 Text Sent Only      0.34     0.34      0.34     0.34
```

In full model, training error is less than test error for both LDA and QDA. Thus LDA and QDA Full moldes are overfitted. But in partial models, both test and training error are equal in all four models. So, they are neither underfitting nor overfitting ie. correct models. LDA full model has smaller Train errors and Test Errors compared to QDA Full Model.

Q1.B: KNN and KNN.CV

```
knn.data = (scale(hw5.data[,c(3,4,5,6,9)]))
# View(knn.data)

# KNN Full Model:
set.seed(1)
knnfm = knn(train = knn.data,
            test = knn.data,
            cl = pinterest,
            k = 1)
knnfm.err = mean(knnfm != pinterest)
knnfm.err # KNN Full Model Training Error Rate.
```

```
## [1] 0
```

```
# KNN Partial Model:
set.seed(1)
knnpm = knn(train = (scale(hw5.data[,4])),
```

```
            test = (scale(hw5.data[,4])),
            cl = pinterest,
            k = 1)
knnpm.err = mean(knnpm != pinterest)
knnpm.err # KNN Partial Model Training Error Rate.
```

```
## [1] 0.2765957
```

```
# Test Error Using KNN.CV:

# KNN CV Full Model:
set.seed(1)
knncvfm = knn.cv(train = knn.data,
                 cl = pinterest,
                 k = 1)
knncvfm.err = mean(knncvfm != pinterest)
knncvfm.err # KNN CV Full Model Test Error Rate.
```

```
## [1] 0.2978723
```

```
# KNN CV Partial Model:
set.seed(1)
knncvpm = knn.cv(train = (scale(hw5.data[,4])),
                 cl = pinterest,
                 k = 1)
knncvpm.err = mean(knncvpm != pinterest)
knncvpm.err # KNN CV Partial Model Test Error Rate.
```

```
## [1] 0.4680851
```

```
#Table:
KNN.Train = round(c(knnfm.err,
                    knnpm.err),
                  3)
KNN.Test = round(c(knncvfm.err,
                   knncvpm.err),
                 3)
#Table:
as.data.frame(cbind(Predictors,
                    KNN.Train,
                    KNN.Test))
```

```
##       Predictors KNN.Train KNN.Test
## 1     Full Model         0    0.298
## 2 Text Sent Only     0.277    0.468
```

Q1.C:

```
#Define two datasets, for full model and partial model respectively:
knn.q1c = (scale(hw5.data[,c(3,4,5,6,9)]))
textsnt = (scale(hw5.data[,4]))

# Define knn.train function as f(trainSet, TestSet, k).
# Write knn function in terms of trainSet, TestSet and k.
# Return training error (train.err).
# Round to three digits after decimal.
# Return and print the result.
```

```
set.seed(1)
knn.train = function(trainSet, testSet, k){
  knn.q1c = knn(train = trainSet, test = testSet,
                cl = hw5.data$pinterest, k = k)
  train.err = round(mean(knn.q1c != hw5.data$pinterest), 3)
  return(print(paste("Training error =" , train.err)))
  }

# Use knn.train function to input values of
# trainSet, testSet and k in above function and
# generate output.

knn.train(trainSet = knn.q1c,
          testSet = knn.q1c,
          k = 2) # Full Model
```

```
## [1] "Training error = 0.17"
```

```
knn.train(trainSet = textsnt,
          testSet = textsnt,
          k = 2) # Partial Model
```

```
## [1] "Training error = 0.277"
```

Q1.D:

```
set.seed(1)
knn.test = function(trainSet, k){
  knncv.q1c = knn.cv(train = trainSet,
                cl = hw5.data$pinterest,
                k = k)
  test.err = round(mean(knncv.q1c != hw5.data$pinterest), 3)
  return(print(paste("Test error =" , test.err)))
  }

knn.test(trainSet = knn.q1c,
         k = 2) # Full Model
```

```
## [1] "Test error = 0.362"
```

```
knn.test(trainSet = textsnt,
         k = 2) # Partial Model
```

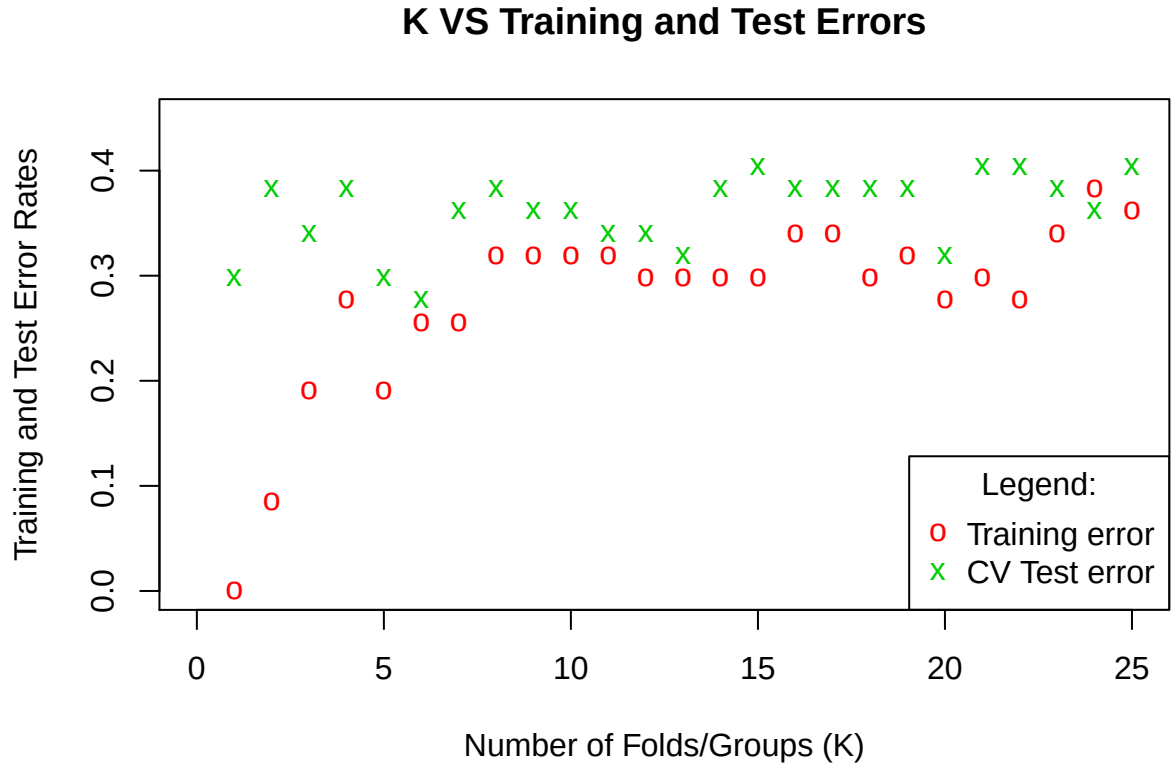
```
## [1] "Test error = 0.489"
```

Q1.E:

```
# Training Error (KNN):
Q1E.TrainError = rep(0, 25) # Stores Training Error from Q1E KNN
Q1E.TestError = rep(0, 25) # Stores Test Error from Q1E KNN.CV
set.seed(1)
for (i in 1:25) {
  q1e.knn = knn(train = knn.q1c,
                test = knn.q1c,
                cl = pinterest,
                k = i)
  Q1E.TrainError[i] = round(mean(q1e.knn != hw5.data$pinterest), 3)
```

```r
# Test Error (KNN.CV):
  q1e.knncv = knn.cv(train = knn.q1c,
                     cl = pinterest,
                     k = i)
  Q1E.TestError[i] = round(mean(q1e.knncv != hw5.data$pinterest), 3)
  }

k = seq(1:25) #K value from 1 to 25.
plot(x = k, y = Q1E.TrainError,
     lty = 2,
     col = 2,
     pch = "o",
     ylim = c(0, 0.45),
     xlim = c(0, 25),
     main = "K VS Training and Test Errors",
     ylab = "Training and Test Error Rates",
     xlab = "Number of Folds/Groups (K)")
points(x = k, y = Q1E.TestError,
       lty = 1,
       col = 3,
       pch = "x")
legend("bottomright",
       col = c(2, 3),
       pch = c("o", "x"),
       title = "Legend:",
       legend = c("Training error",
                  "CV Test error"))
```



Answer: When k = 1, the training error is low and test error is high and thus overfitting models. We want

less test error. Around K = 5, test and training error are starting to become equal. Visually, smallest distance between training error and test error is somewhere around k = 10. The training error is higher than test error around K = 25. So, the model might have overfitted when K = 1 and underfitted when K = 25. K between 10 to 15 (ideally 10, in this case) seems like a better fitting range.

Q1.F:

```
as.data.frame(cbind(Predictors,
                    LDA.Train,
                    LDA.Test,
                    QDA.Train,
                    QDA.Test,
                    KNN.Train,
                    KNN.Test))
```

```
##       Predictors LDA.Train LDA.Test QDA.Train QDA.Test KNN.Train KNN.Test
## 1     Full Model     0.255    0.319     0.362    0.511         0    0.298
## 2 Text Sent Only      0.34     0.34      0.34     0.34     0.277    0.468
```

KNN full model is best for classification at K = 1. Because, the training error is zero and the test error is lowest among all six models.

BOOTSTRAPPING

Q2.A:

```
set.seed(1)
q2.data = (hw5.data[,c(3,4,5,6,9)]) #Data.
index = sample(seq(1:47))
q2afunction = function(data, index){
  q2a.pca = princomp(data[index,], cor = TRUE)
  IPC.stdev = round(q2a.pca$sdev[1],3)
  return(IPC.stdev)
}
Stdev = q2afunction(q2.data, index)
cat("Standard Deviation of Ist PC = ", Stdev)
```

```
## Standard Deviation of Ist PC =  1.425
```

Q2.B:

```
boots = boot(q2.data, q2afunction, 1000)
boots
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = q2.data, statistic = q2afunction, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    1.425 0.049162   0.0739273
```

```
CI = boot.ci(boot.out = boots, type = "perc")
CI
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boots, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 1.362,  1.643 )
## Calculations and Intervals on Original Scale
```

```
cat("95% Confidence interval of
    Standard Deviation of Ist PC = [",
    round(CI$percent[,4],3),",",
    round(CI$percent[,5],3),"]")
```

```
## 95% Confidence interval of
##     Standard Deviation of Ist PC = [ 1.362 , 1.643 ]
```

Q2.C:

We need to change "sdev[1]" to "loadings[1]". Other names can be changed to reflect the factor loading. However, these changes are not vital for analysis. Example is presented below:

```
q2.data = (hw5.data[,c(3,4,5,6,9)]) #Data.
index = sample(seq(1:47))
q2afunctionc = function(data, index){
  q2a.pcac = princomp(data[index,], cor = TRUE)
  IPC.loadingc = round(q2a.pcac$loadings[1],3)
  return(IPC.loadingc)
}
pcloadingc = q2afunctionc(q2.data, index)
cat("Standard Deviation of Ist PC = ", pcloadingc)
```

```
## Standard Deviation of Ist PC =  0.382
```

```
bootsc = boot(q2.data, q2afunctionc, 1000)
bootsc
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = q2.data, statistic = q2afunctionc, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*    0.382 -0.030513   0.1127838
```

```
CIc = boot.ci(boot.out = bootsc, type = "perc")
CIc
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
```

```
## boot.ci(boot.out = bootsc, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0641,  0.5060 )
## Calculations and Intervals on Original Scale
```

```
cat("95% Confidence interval of Ist PC loading = [", round(CIc$percent[,4],3),",",
    round(CIc$percent[,5],3),"]")
```

```
## 95% Confidence interval of Ist PC loading = [ 0.064 , 0.506 ]
```