Jefrey Kerley
Jakcqc
2/7/2022

```
## Problem 1:
## A) The difference between test MSE and Training MSE
##    is that the test MSE is the MSE as it relates to more general data.
##    Meaning, that you would use the training MSE to gauge how well
##    the line of best fit is, only when dealing with the sample data.
##    So the difference between the two MSEs, is that one is relevant to the
##    sample data, and one is relevant to new general data, that was not in
##    the training set.
## B) The two curves show a different trend because as the complexity for
##    training set increases, the line over fits and is able to "correctly"
##    fit all training data. Yet, when this is applied to the testing data,
##    we see that the error sky rockets as the over fitted line does poorly
##    on data that does not match the training data.
## C) The large test MSE for the linear regression fit, is that the line
##    was too generalized, and because the "true" line of best fit was a
##    cubic line. As a result, more complexity needed to be added to prevent
##    a large testing MSE for this line.
## D) The spline with high degrees of freedom has a high test MSE, because it
##    is not generalized at all. This is a classic over fitted line, that was
##    trained "too much", on the training data. instead of learning, this line
##    memorized the data.
## E) The training MSE that are lower than the irreducible error, as this error
##    acts as the minimum lower bound for the test MSE. Which is why we see the
##    test MSE never drop below this. Over fitted, and "perfect" lines
##    for some data, it is possible for the test MSE to drop practically to 0.
##    Furthermore, this error simply acts as a representation of the real world
##    truth that is unknown, ie- known knowns, unknown knowns, etc...
##
##Problem 2
## a)
collegeData = read.csv('../College.csv', stringsAsFactors = TRUE)
View(collegeData)
##b)
rownames(collegeData) = collegeData[, 1]
View(collegeData)
collegeData = collegeData[, -1]
View(collegeData)
## c)
summary(collegeData)
pairs(collegeData[,1:10])
```

```r
plot(collegeData$Private, collegeData$Outstate)
Elite <- rep ("No", nrow (collegeData))
Elite[collegeData$Top10perc > 50] <- " Yes "
Elite <- as.factor(Elite)
collegeData <- data.frame (collegeData , Elite)
summary(Elite)
plot(Elite, collegeData$Outstate)
par(mfrow = c(2,2))
hist(collegeData$Top10perc)
hist(collegeData$Top25perc)
hist(collegeData$Enroll)
hist(collegeData$Apps)
##
## Problem 3
##
?data.frame
?runif
set.seed(80)
train.X <- data.frame(x1 = runif(n=100,-1,1), x2 = runif(n=100,-1,1))
# Randomly select 100 locations of (x1, x2)
prob <- ifelse( (-0.5 < train.X$x1) & (train.X$x1 < 0.5)
            & (-0.5 < train.X$x2) & (train.X$x2 < 0), 0.9, 0.1)
# If (x1, x2) is inside the rectangle, prob is 0.9, otherwise, prob is 0.1
train.Y <- as.factor(runif(n=100) < prob)
# Simulate class labels according to prob
colors <- c("skyblue","red")
plot(train.X$x1, train.X$x2, pch=20, col=colors[factor(train.Y)])
segments(-0.5, -0.5, 0.5, -0.5, col="orange", lwd=2)
segments(-0.5, -0.5, -0.5, 0, col="orange", lwd=2)
segments(-0.5, 0, 0.5, 0, col="orange", lwd=2)
segments(0.5, -0.5, 0.5, 0, col="orange", lwd=2)
## a) The Bayes error rate would be .10, as the irreducible error == bayes
##    error, thus are prediction rate would be 90% given a large set of data.
##    To further explain, Bayes error is the minimum error you can achieve.
##
## b) Below we can see that the distribution is a reasonable result.
set.seed(100)
train.X <- data.frame(x1 = sample(0:1, 100, replace = T))
train.Y <- as.factor(train.X$x1 > 0)
colors <- c("black","blue")
plot(train.Y, col=colors[factor(train.Y)], ylim = range(0,70))
##
## c) Below we can see that when k = 1, it does not satisfy the predictive
##    outcome. The classification "areas", are not relative to the right
```

```
##    classificiation predictive classes.
xGrid.1d <- seq(-1,1,0.02)
nx <- length(xGrid.1d)
xGrid.2d <- data.frame(x1=rep(xGrid.1d, each=nx), x2=rep(xGrid.1d,times=nx))
library(class)
set.seed(1)
runBlockKNN = function(knnSize)
{
  knn.pred <- knn(train=train.X, test=xGrid.2d, cl=train.Y, k=knnSize)
  plot(xGrid.2d$x1, xGrid.2d$x2, pch=3, col=colors[factor(knn.pred)])
  segments(-0.5, -0.5, 0.5, -0.5, col="orange", lwd=2)
  segments(-0.5, -0.5, -0.5, 0, col="orange", lwd=2)
  segments(-0.5, 0, 0.5, 0, col="orange", lwd=2)
  segments(0.5, -0.5, 0.5, 0, col="orange", lwd=2)
}
runBlockKNN(1)

## d) With increasing values of k, to an upper bound, we see the classification
##    becoming more true to the right predicitve classification.
runBlockKNN(2)
runBlockKNN(3)
runBlockKNN(4)
runBlockKNN(5)
runBlockKNN(6)
runBlockKNN(7)
runBlockKNN(8)
runBlockKNN(12)
runBlockKNN(15)
## e) The best k value here seems to be around k = 10, up to maybe around
##    k = 15. This is where the classification starts to be more accurate
##    of the expective predictive outcome.

##
##
##
##
##
##
##
##
##
##
##
```