stats101c_hw2

Takao

10/12/2022

Takao Oba

HW 2

Question 1

(a) Report dimension of your data and summary statistics of the variables in your data

```
setwd("/Users/takaooba/Downloads/")
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2

## Warning: package 'ggplot2' was built under R version 4.1.2

## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'readr' was built under R version 4.1.2

## Warning: package 'purrr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2

## Warning: package 'stringr' was built under R version 4.1.2

## Warning: package 'forcats' was built under R version 4.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
breast <- read_csv("BreastCancer.csv")
```

```
## New names:
## Rows: 569 Columns: 12
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): diagnosis dbl (11): ...1, radius_mean, texture_mean, perimeter_mean,
## area_mean, smooth...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

```r
breast <- breast[,-1]
head(breast)
```

```
## # A tibble: 6 x 11
##   diagnosis radius_mean textur~1 perim~2 area_~3 smoot~4 compa~5 conca~6 conca~7
##   <chr>           <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 M                18.0     10.4   123.     1001  0.118   0.278   0.300   0.147
## 2 M                20.6     17.8   133.     1326  0.0847  0.0786  0.0869  0.0702
## 3 M                19.7     21.2   130      1203  0.110   0.160   0.197   0.128
## 4 M                11.4     20.4    77.6     386. 0.142   0.284   0.241   0.105
## 5 M                20.3     14.3   135.     1297  0.100   0.133   0.198   0.104
## 6 M                12.4     15.7    82.6     477. 0.128   0.17    0.158   0.0809
## # ... with 2 more variables: symmetry_mean <dbl>, fractal_dimension_mean <dbl>,
## #   and abbreviated variable names 1: texture_mean, 2: perimeter_mean,
## #   3: area_mean, 4: smoothness_mean, 5: compactness_mean, 6: concavity_mean,
## #   7: concave.points_mean
```

```r
dim(breast)
```

```
## [1] 569  11
```

Upon removing the first column, we have that the dimension of the data is 569 by 11.

```r
summary(breast)
```

```
##   diagnosis          radius_mean      texture_mean   perimeter_mean  
##  Length:569         Min.   : 6.981   Min.   : 9.71   Min.   : 43.79  
##  Class :character   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17  
##  Mode  :character   Median :13.370   Median :18.84   Median : 86.24  
##                     Mean   :14.127   Mean   :19.29   Mean   : 91.97  
##                     3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10  
##                     Max.   :28.110   Max.   :39.28   Max.   :188.50  
##    area_mean      smoothness_mean   compactness_mean  concavity_mean   
##  Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000  
##  1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956  
##  Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154  
##  Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880  
##  3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070  
##  Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680  
```

```
##  concave.points_mean symmetry_mean    fractal_dimension_mean
##  Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770
##  Median :0.03350     Median :0.1792   Median :0.06154
##  Mean   :0.04892     Mean   :0.1812   Mean   :0.06280
##  3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612
##  Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
```

Shown above is the summary statistics of the variables in the data.

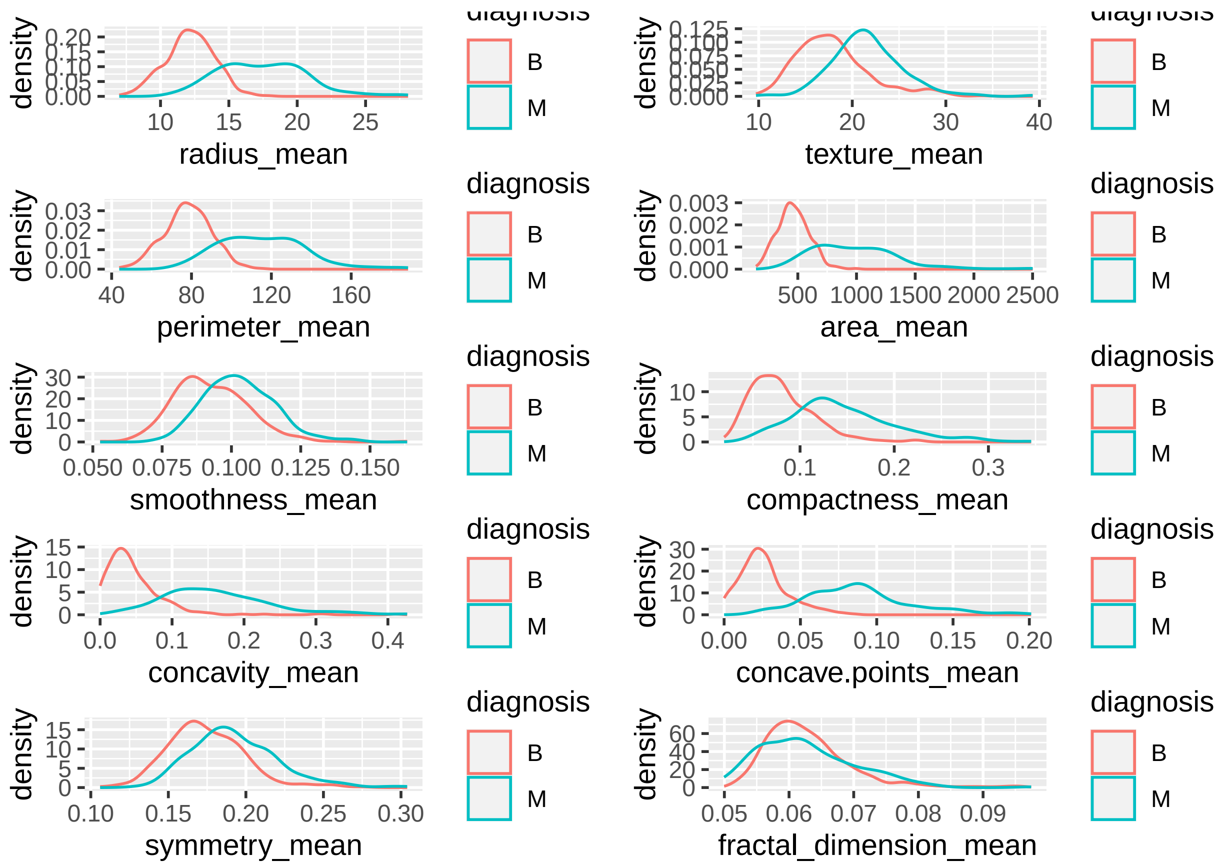
(b) Choose "best" three predictors based on density plots of Malignant and Benign categories.

```r
p1 <- ggplot(breast, aes(radius_mean, color = diagnosis)) + geom_density()
p2 <- ggplot(breast, aes(texture_mean, color = diagnosis)) + geom_density()
p3 <- ggplot(breast, aes(perimeter_mean, color = diagnosis)) + geom_density()
p4 <- ggplot(breast, aes(area_mean, color = diagnosis)) + geom_density()
p5 <- ggplot(breast, aes(smoothness_mean, color = diagnosis)) + geom_density()
p6 <- ggplot(breast, aes(compactness_mean, color = diagnosis)) + geom_density()
p7 <- ggplot(breast, aes(concavity_mean, color = diagnosis)) + geom_density()
p8 <- ggplot(breast, aes(concave.points_mean, color = diagnosis)) + geom_density()
p9 <- ggplot(breast, aes(symmetry_mean, color = diagnosis)) + geom_density()
p10 <- ggplot(breast, aes(fractal_dimension_mean, color = diagnosis)) + geom_density()
# install.packages("gridExtra")
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

```r
grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10, nrow = 5)
```



Now we will aim to choose the three best predictors. Through the density plots, I would say that the radius_mean, perimeter_mean, and the concave.points_mean are the three best predictors.

(c) Use the "best" three predictors to create knn classifiers (k=1,3,5,7,9 and11). Use 400 observations for the training data set and 169 observations for the testing data set. Use Set.seed(1234567)

```
set.seed(1234567)
S.test.i <- sample(1:569, 169, replace = F)

X.mat <- breast[,c(2,4,9)]
S.X.test<-X.mat[S.test.i,]
S.X.train<- X.mat[-S.test.i,]

S.Y.test <- breast$diagnosis[S.test.i]
S.Y.train <- breast$diagnosis[-S.test.i]

library(class)
```

```
## Warning: package 'class' was built under R version 4.1.2
```

```
S.out.1 <- knn(S.X.train, S.X.test, S.Y.train, k = 1)
table(S.out.1, S.Y.test)
```

```
##        S.Y.test
## S.out.1   B   M
##       B 106   7
##       M   6  50
```

```r
# mean(S.out.1 != S.Y.test)
# mean(S.out.1 == S.Y.test)

S.out.3 <- knn(S.X.train, S.X.test, S.Y.train, k = 3)
table(S.out.3, S.Y.test)
```

```
##        S.Y.test
## S.out.3   B   M
##       B 104   7
##       M   8  50
```

```r
# mean(S.out.3 != S.Y.test)
# mean(S.out.3 == S.Y.test)

S.out.5 <- knn(S.X.train, S.X.test, S.Y.train, k = 5)
table(S.out.5, S.Y.test)
```

```
##        S.Y.test
## S.out.5   B   M
##       B 105  10
##       M   7  47
```

```r
# mean(S.out.5 != S.Y.test)
# mean(S.out.5 == S.Y.test)

S.out.7 <- knn(S.X.train, S.X.test, S.Y.train, k = 7)
table(S.out.7, S.Y.test)
```

```
##        S.Y.test
## S.out.7   B   M
##       B 106   8
##       M   6  49
```

```r
# mean(S.out.7 != S.Y.test)
# mean(S.out.7 == S.Y.test)

S.out.11 <- knn(S.X.train, S.X.test, S.Y.train, k = 11)
table(S.out.11, S.Y.test)
```

```
##         S.Y.test
## S.out.11   B   M
##        B 105   7
##        M   7  50
```

```r
# mean(S.out.11 != S.Y.test)
# mean(S.out.11 == S.Y.test)
```

(d) Report the misclassification rate and report your best k.

```r
# Misclassification rate for k = 1
mean(S.out.1 != S.Y.test)
```

```
## [1] 0.07692308
```

```r
# Misclassification rate for k = 3
mean(S.out.3 != S.Y.test)
```

```
## [1] 0.0887574
```

```r
# Misclassification rate for k = 5
mean(S.out.5 != S.Y.test)
```

```
## [1] 0.1005917
```

```r
# Misclassification rate for k = 7
mean(S.out.7 != S.Y.test)
```

```
## [1] 0.08284024
```

```r
# Misclassification rate for k = 11
mean(S.out.11 != S.Y.test)
```

```
## [1] 0.08284024
```

The best k or the k with the lowest misclassification rate is when k = 1 with a misclassification rate of 0.07692308.

(e) Re-create your knn classifiers, but this time scale your variables first. (Use the same set of k values)

```r
set.seed(1234567)
# Using the scale function
X.mat1 <- scale(breast[,c(2,4,9)])

S.X.test<-X.mat1[S.test.i,]
S.X.train<- X.mat1[-S.test.i,]

S.Y.test <- (breast$diagnosis[S.test.i])
S.Y.train <- breast$diagnosis[-S.test.i]

S.out.1 <- knn(S.X.train, S.X.test, S.Y.train, k = 1)
table(S.out.1, S.Y.test)
```

```
##         S.Y.test
## S.out.1   B   M
##       B 102   5
##       M  10  52
```

```
S.out.3 <- knn(S.X.train, S.X.test, S.Y.train, k = 3)
table(S.out.3, S.Y.test)
```

```
##         S.Y.test
## S.out.3   B   M
##       B 104   4
##       M   8  53
```

```
S.out.5 <- knn(S.X.train, S.X.test, S.Y.train, k = 5)
table(S.out.5, S.Y.test)
```

```
##         S.Y.test
## S.out.5   B   M
##       B 105   5
##       M   7  52
```

```
S.out.7 <- knn(S.X.train, S.X.test, S.Y.train, k = 7)
table(S.out.7, S.Y.test)
```

```
##         S.Y.test
## S.out.7   B   M
##       B 104   5
##       M   8  52
```

```
S.out.11 <- knn(S.X.train, S.X.test, S.Y.train, k = 11)
table(S.out.11, S.Y.test)
```

```
##          S.Y.test
## S.out.11   B   M
##        B 107   4
##        M   5  53
```

(f) Report the misclassification rate and report your best k.

```
# Misclassification rate for k = 1
mean(S.out.1 != S.Y.test)
```

```
## [1] 0.0887574
```

```
# Misclassification rate for k = 3
mean(S.out.3 != S.Y.test)
```

```
## [1] 0.07100592
```

```r
# Misclassification rate for k = 5
mean(S.out.5 != S.Y.test)
```

```
## [1] 0.07100592
```

```r
# Misclassification rate for k = 7
mean(S.out.7 != S.Y.test)
```

```
## [1] 0.07692308
```

```r
# Misclassification rate for k = 11
mean(S.out.11 != S.Y.test)
```

```
## [1] 0.05325444
```

The best k or the k with the lowest misclassification rate is when k = 11 with a misclassification rate of 0.0532544.

Question 2

Re-do question 1, this time use all numerical predictors. Is it any better than the results in Question 2?

The numerical predictors will be everything besides the diagnosis column, thus we will have:

```r
breast_new <- breast[,-1]
```

No scaling

```r
set.seed(1234567)
S.test.i <- sample(1:569, 169, replace = F)

X.mat <- breast_new
S.X.test<-X.mat[S.test.i,]
S.X.train<- X.mat[-S.test.i,]

S.Y.test <- breast$diagnosis[S.test.i]
S.Y.train <- breast$diagnosis[-S.test.i]

S.out.1 <- knn(S.X.train, S.X.test, S.Y.train, k = 1)
table(S.out.1, S.Y.test)
```

```
##         S.Y.test
## S.out.1   B   M
##       B 100  11
##       M  12  46
```

```r
S.out.3 <- knn(S.X.train, S.X.test, S.Y.train, k = 3)
table(S.out.3, S.Y.test)
```

```
##        S.Y.test
## S.out.3   B   M
##       B 101   7
##       M  11  50
```

```r
S.out.5 <- knn(S.X.train, S.X.test, S.Y.train, k = 5)
table(S.out.5, S.Y.test)
```

```
##        S.Y.test
## S.out.5   B   M
##       B 103   9
##       M   9  48
```

```r
S.out.7 <- knn(S.X.train, S.X.test, S.Y.train, k = 7)
table(S.out.7, S.Y.test)
```

```
##        S.Y.test
## S.out.7   B   M
##       B 104   9
##       M   8  48
```

```r
S.out.11 <- knn(S.X.train, S.X.test, S.Y.train, k = 11)
table(S.out.11, S.Y.test)
```

```
##         S.Y.test
## S.out.11   B   M
##        B 107   9
##        M   5  48
```

Misclassification Rate

```r
# Misclassification rate for k = 1
mean(S.out.1 != S.Y.test)
```

```
## [1] 0.1360947
```

```r
# Misclassification rate for k = 3
mean(S.out.3 != S.Y.test)
```

```
## [1] 0.1065089
```

```r
# Misclassification rate for k = 5
mean(S.out.5 != S.Y.test)
```

```
## [1] 0.1065089
```

```r
# Misclassification rate for k = 7
mean(S.out.7 != S.Y.test)
```

```
## [1] 0.1005917
```

```r
# Misclassification rate for k = 11
mean(S.out.11 != S.Y.test)
```

```
## [1] 0.08284024
```

The best k or the k with the lowest misclassification rate is when k = 11 with a misclassification rate of 0.08284024.

After scaling

```r
set.seed(1234567)
# Using the scale function
X.mat1 <- scale(breast_new)

S.X.test<-X.mat1[S.test.i,]
S.X.train<- X.mat1[-S.test.i,]

S.Y.test <- breast$diagnosis[S.test.i]
S.Y.train <- breast$diagnosis[-S.test.i]

S.out.1 <- knn(S.X.train, S.X.test, S.Y.train, k = 1)
table(S.out.1, S.Y.test)
```

```
##        S.Y.test
## S.out.1   B   M
##       B 105   3
##       M   7  54
```

```r
S.out.3 <- knn(S.X.train, S.X.test, S.Y.train, k = 3)
table(S.out.3, S.Y.test)
```

```
##        S.Y.test
## S.out.3   B   M
##       B 106   2
##       M   6  55
```

```r
S.out.5 <- knn(S.X.train, S.X.test, S.Y.train, k = 5)
table(S.out.5, S.Y.test)
```

```
##        S.Y.test
## S.out.5   B   M
##       B 107   2
##       M   5  55
```

```r
S.out.7 <- knn(S.X.train, S.X.test, S.Y.train, k = 7)
table(S.out.7, S.Y.test)
```

```
##        S.Y.test
## S.out.7   B   M
##       B 105   2
##       M   7  55
```

```r
S.out.11 <- knn(S.X.train, S.X.test, S.Y.train, k = 11)
table(S.out.11, S.Y.test)
```

```
##         S.Y.test
## S.out.11   B   M
##        B 109   2
##        M   3  55
```

Misclassification rate after scaling

```r
# Misclassification rate for k = 1
mean(S.out.1 != S.Y.test)
```

```
## [1] 0.0591716
```

```r
# Misclassification rate for k = 3
mean(S.out.3 != S.Y.test)
```

```
## [1] 0.04733728
```

```r
# Misclassification rate for k = 5
mean(S.out.5 != S.Y.test)
```

```
## [1] 0.04142012
```

```r
# Misclassification rate for k = 7
mean(S.out.7 != S.Y.test)
```

```
## [1] 0.05325444
```

```r
# Misclassification rate for k = 11
mean(S.out.11 != S.Y.test)
```

```
## [1] 0.0295858
```

The best k or the k with the lowest misclassification rate is when k = 11 with a misclassification rate of 0.0295858.

Comparing to the parts in Question 1, we have that the misclassification rate is lower with numerical predictors, so it is in fact better than the results in question 1.

Question 3

```
breast <- read_csv("/Users/takaooba/Downloads/BreastCancer.csv")
```

```
## New names:
## Rows: 569 Columns: 12
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): diagnosis dbl (11): ...1, radius_mean, texture_mean, perimeter_mean,
## area_mean, smooth...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

```
breast <- breast[,-1]
# head(breast)
```

(a) Apply logistic regression model using all predictors. Report confusion matrices of both the training and the test data sets. (Same training and testing data sets created for question 2.

```
lr.model <- glm(factor(diagnosis) ~ . , data = breast, family = binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
breast2 <- breast[,-1]

breast.test <- breast2[S.test.i,]

pred.test <- predict(lr.model, breast.test)
# breast.test

breast.glm.pred = rep("B", length(pred.test))
breast.glm.pred[pred.test >= 0] = "M"

# Confusion matrix of the testing data sets
table1 <- table(breast.glm.pred, S.Y.test)
table1
```

```
##                 S.Y.test
## breast.glm.pred   B   M
##               B 112   3
##               M   0  54
```

Now we will assess the training data

```r
breast.train <- breast2[-S.test.i,]
pred.test.2 <- predict(lr.model, breast.train)

breast.glm.pred.2 = rep("B", length(pred.test.2))
breast.glm.pred.2[pred.test.2 >= 0] = "M"

# Confusion matrix of the training data
table2 <- table(breast.glm.pred.2,S.Y.train)
table2
```

```
##                  S.Y.train
## breast.glm.pred.2   B   M
##                 B 235  16
##                 M  10 139
```

```r
summary(lr.model)
```

```
##
## Call:
## glm(formula = factor(diagnosis) ~ ., family = binomial(), data = breast)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.95590  -0.14839  -0.03943   0.00429   2.91690
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -7.35952   12.85259  -0.573   0.5669
## radius_mean             -2.04930    3.71588  -0.551   0.5813
## texture_mean             0.38473    0.06454   5.961  2.5e-09 ***
## perimeter_mean          -0.07151    0.50516  -0.142   0.8874
## area_mean                0.03980    0.01674   2.377   0.0174 *
## smoothness_mean         76.43227   31.95492   2.392   0.0168 *
## compactness_mean        -1.46242   20.34249  -0.072   0.9427
## concavity_mean           8.46870    8.12003   1.043   0.2970
## concave.points_mean     66.82176   28.52910   2.342   0.0192 *
## symmetry_mean           16.27824   10.63059   1.531   0.1257
## fractal_dimension_mean -68.33703   85.55666  -0.799   0.4244
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 751.44  on 568  degrees of freedom
## Residual deviance: 146.13  on 558  degrees of freedom
## AIC: 168.13
##
## Number of Fisher Scoring iterations: 9
```

Above is the summary output of the generated logistic regression. We assess that there is a great difference between the null deviance which closely resembles the SSTotal and the residual deviance which closely resembles the SSE. We like the gap that we notice and is a good thing to see.

(b) Scale all variables, the create another logistic regression model using all predictors.

```
breast3 <- data.frame(scale(breast[,-1]))
lr.model <- glm(factor(diagnosis) ~ . , data = breast, family = binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
breast.test.1 <- breast3[S.test.i,]

pred.test.3 <- predict(lr.model, breast.test.1)
# breast.test

breast.glm.pred.3 = rep("B", length(pred.test.3))
breast.glm.pred.3[pred.test.3 >= 0] = "M"

# Confusion matrix of the testing data sets
table3 <- table(breast.glm.pred.3, S.Y.test)
table3
```

```
##                  S.Y.test
## breast.glm.pred.3  B  M
##                 B 97  4
##                 M 15 53
```

Now we will assess the training data

```
breast.train.2 <- breast3[-S.test.i,]
pred.test.4 <- predict(lr.model, breast.train.2)
breast.glm.pred.4 = rep("B", length(pred.test.4))
breast.glm.pred.4[pred.test.4 >= 0] = "M"

# Confusion matrix of the training data
table4 <- table(breast.glm.pred.4,S.Y.train)
table4
```

```
##                  S.Y.train
## breast.glm.pred.4   B   M
##                 B 206  18
##                 M  39 137
```

(c) Report your confusion matrices and misclassification rates for both the training and testing data sets.

For the unscaled data

```
table1
```

```
##                S.Y.test
## breast.glm.pred   B   M
##               B 112   3
##               M   0  54
```

```r
# Misclassification rate
mean(breast.glm.pred != S.Y.test)
```

```
## [1] 0.01775148
```

```r
table2
```

```
##                   S.Y.train
## breast.glm.pred.2   B   M
##                   B 235  16
##                   M  10 139
```

```r
# Misclassification rate
mean(breast.glm.pred.2 != S.Y.train)
```

```
## [1] 0.065
```

For scaled data

```r
table3
```

```
##                   S.Y.test
## breast.glm.pred.3  B  M
##                   B 97  4
##                   M 15 53
```

```r
# Misclassification rate
mean(breast.glm.pred.3 != S.Y.test)
```

```
## [1] 0.112426
```

```r
table4
```

```
##                   S.Y.train
## breast.glm.pred.4   B   M
##                   B 206  18
##                   M  39 137
```

```r
# Misclassification rate
mean(breast.glm.pred.4 != S.Y.train)
```

```
## [1] 0.1425
```

Question 4

Compare and contrast the results of your KNN models and your logistic regression models. Which one of those models is your hero model? Why?

We assess the misclassification rate for all the table that we have created thus far. In question 2, we have concluded that the best k or the k with the lowest misclassification rate is when k = 11 with a misclassification rate of 0.0295858. However, in question 3, we see that the best model (out of all the models that we have generated) is the unscaled glm model utilizing logistic regression. When using this model, we were able to achieve the lowest misclassification rate of 0.01775148.

Question 5

Split the Boston data posted on bruinlearn week 3 into 80% training and 20% testing data: use set.seed(1128)

```r
bostonData <- read_csv("/Users/takaooba/Downloads/boston.csv")
```

```
## Rows: 506 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): crim, zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
dim(bostonData)
```

```
## [1] 506  14
```

```r
506*0.2
```

```
## [1] 101.2
```

```r
# training 405, testing 101
set.seed(1128)
test.i <- sample(1:506, 101, replace = F)
x.test <- bostonData[test.i,]
x.train <- bostonData[-test.i,]


# the median of the crime rate can be determined through the summary function
summary(x.train$crim)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.01096  0.08265  0.26363  3.74667  3.53501 88.97620
```

```r
# median is 0.26363
crMedian <- 0.26363


# We will determine which observations are greater than the median and will observe that through assign
crimrate <- rep(0, length(bostonData$crim))
for (i in 1:length(bostonData$crim)){
  if(bostonData$crim[i] > crMedian){
    crimrate[i] <- 1
  }
}

bostonData1 <- data.frame(bostonData, crimrate)
head(bostonData1)
```

```
##      crim   zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18.0  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0.0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.06905  0.0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 4 0.02985  0.0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
## 5 0.08829 12.5  7.87    0 0.524 6.012 66.6 5.5605   5 311    15.2 395.60 12.43
## 6 0.14455 12.5  7.87    0 0.524 6.172 96.1 5.9505   5 311    15.2 396.90 19.15
##   medv crimrate
## 1 24.0        0
## 2 21.6        0
## 3 36.2        0
## 4 28.7        0
## 5 22.9        0
## 6 27.1        0
```

```
# We will look at the new train and testing data frame
boston.test <- bostonData1[test.i,]
boston.train <- bostonData1[-test.i,]
dim(boston.test)
```

```
## [1] 101  15
```

We will then investigate which predictors are the most significant

```
cor(boston.train)
```

```
##                 crim          zn       indus        chas        nox          rm
## crim      1.00000000 -0.19070657  0.38373023 -0.05254358  0.3977812 -0.21919186
## zn       -0.19070657  1.00000000 -0.53113966 -0.06529596 -0.5117771  0.27932552
## indus     0.38373023 -0.53113966  1.00000000  0.09114113  0.7583027 -0.38987859
## chas     -0.05254358 -0.06529596  0.09114113  1.00000000  0.1346566  0.09417681
## nox       0.39778120 -0.51177714  0.75830269  0.13465655  1.0000000 -0.30901187
## rm       -0.21919186  0.27932552 -0.38987859  0.09417681 -0.3090119  1.00000000
## age       0.32740799 -0.57740371  0.63335651  0.10848152  0.7152417 -0.24959396
## dis      -0.36154263  0.66308903 -0.70169262 -0.12212935 -0.7660507  0.19924917
## rad       0.60826275 -0.30623278  0.57904506  0.01442969  0.5969556 -0.21319404
## tax       0.56114318 -0.30826032  0.72463307 -0.01337514  0.6594471 -0.30350177
## ptratio   0.27838084 -0.34244294  0.35621506 -0.11970174  0.1629552 -0.31786365
## black    -0.40830350  0.17240463 -0.36522203  0.04726198 -0.3735093  0.13660919
## lstat     0.43522786 -0.40475564  0.58807216 -0.05818729  0.5686407 -0.61984095
## medv     -0.38011046  0.31545633 -0.47642577  0.17862496 -0.4108785  0.70548406
## crimrate  0.39633280 -0.42694313  0.58902415  0.10287004  0.7157084 -0.14441198
##                age        dis         rad         tax    ptratio       black
## crim      0.3274080 -0.3615426  0.60826275  0.56114318  0.2783808 -0.40830350
## zn       -0.5774037  0.6630890 -0.30623278 -0.30826032 -0.3424429  0.17240463
## indus     0.6333565 -0.7016926  0.57904506  0.72463307  0.3562151 -0.36522203
## chas      0.1084815 -0.1221294  0.01442969 -0.01337514 -0.1197017  0.04726198
## nox       0.7152417 -0.7660507  0.59695561  0.65944710  0.1629552 -0.37350930
## rm       -0.2495940  0.1992492 -0.21319404 -0.30350177 -0.3178637  0.13660919
## age       1.0000000 -0.7361661  0.42798485  0.48629042  0.2353040 -0.26271856
## dis      -0.7361661  1.0000000 -0.49088987 -0.53180866 -0.2123928  0.29336190
## rad       0.4279848 -0.4908899  1.00000000  0.90130157  0.4530058 -0.47694193
## tax       0.4862904 -0.5318087  0.90130157  1.00000000  0.4567533 -0.46523579
```

```
## ptratio   0.2353040 -0.2123928  0.45300580  0.45675331  1.0000000 -0.17926832
## black    -0.2627186  0.2933619 -0.47694193 -0.46523579 -0.1792683  1.00000000
## lstat     0.6115116 -0.4748234  0.47452560  0.53580165  0.3616868 -0.35877727
## medv     -0.3644717  0.2244874 -0.37102742 -0.47004902 -0.4776864  0.34193385
## crimrate  0.5918706 -0.6089804  0.62175721  0.59845376  0.2209815 -0.36319529
##                lstat       medv   crimrate
## crim      0.43522786 -0.3801105  0.3963328
## zn       -0.40475564  0.3154563 -0.4269431
## indus     0.58807216 -0.4764258  0.5890241
## chas     -0.05818729  0.1786250  0.1028700
## nox       0.56864073 -0.4108785  0.7157084
## rm       -0.61984095  0.7054841 -0.1444120
## age       0.61151165 -0.3644717  0.5918706
## dis      -0.47482339  0.2244874 -0.6089804
## rad       0.47452560 -0.3710274  0.6217572
## tax       0.53580165 -0.4700490  0.5984538
## ptratio   0.36168681 -0.4776864  0.2209815
## black    -0.35877727  0.3419339 -0.3631953
## lstat     1.00000000 -0.7501347  0.4184869
## medv     -0.75013472  1.0000000 -0.2428595
## crimrate  0.41848686 -0.2428595  1.0000000
```

From above, we can see that the best predictors are (from best going down): nox, rad, dis, tax, age. Now we can further investigate by 3 predictors, 4 predictors, and 5 predictors.

Using the training data fit classification models in order to predict whether a given suburb has a crime rate above or below the median (create a new response variable for crime rate). Explore and report the performance of logistic regression and KNN models using various subsets of the predictors (Best 3 predictors, Best 4 predictors, and Best 5 predictors). Describe your findings.

3 Predictors

```r
lr.model3 <- glm(factor(crimrate) ~ nox + rad + dis, data = bostonData1, family = binomial())
summary(lr.model3)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + rad + dis, family = binomial(),
##     data = bostonData1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.91461  -0.32369  -0.06538   0.00648   2.64083
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.7905     3.0667  -6.779 1.21e-11 ***
## nox          32.0836     4.7029   6.822 8.97e-12 ***
## rad           0.5166     0.1005   5.138 2.77e-07 ***
```

```
## dis           0.2028     0.1362   1.489    0.136
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 264.13  on 502  degrees of freedom
## AIC: 272.13
##
## Number of Fisher Scoring iterations: 8
```

This results is pretty interesting, in fact we see that there is a gap between the null deviance and the residual deviance. However, "dis" is an insignificant predictor as the z-value is 1.489.

```
boston.pred <- predict(lr.model3, data = boston.train, newdata = boston.test)
boston.glm.pred <- rep(0, length(boston.pred))
boston.glm.pred[boston.pred >= 0] <- 1

length(boston.glm.pred)
```

```
## [1] 101
```

```
# boston.test
table(boston.glm.pred, boston.test[,15])
```

```
##
## boston.glm.pred  0  1
##               0 51 11
##               1  2 37
```

```
#The misclassification rate is going to be calculated as
mean(boston.glm.pred != boston.test[,15])
```

```
## [1] 0.1287129
```

We will try to assess the next best predictor, and simply going off the next in the list, we will try to assess nox, rad, and tax

```
lr.model3 <- glm(factor(crimrate) ~ nox + rad + age, family = binomial(), data = bostonData1)
summary(lr.model3)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + rad + age, family = binomial(),
##     data = bostonData1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.89196  -0.35145  -0.06124   0.00747   2.59298
##
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -16.913418   1.952868  -8.661  < 2e-16 ***
## nox          25.404884   3.776367   6.727 1.73e-11 ***
## rad           0.509933   0.099923   5.103 3.34e-07 ***
## age           0.006396   0.008104   0.789     0.43    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 265.69  on 502  degrees of freedom
## AIC: 273.69
##
## Number of Fisher Scoring iterations: 8
```

```
lr.model3 <- glm(factor(crimrate) ~ nox + dis + tax, family = binomial(), data = bostonData1)
summary(lr.model3)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + dis + tax, family = binomial(),
##     data = bostonData1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max  
## -2.3193  -0.4086  -0.1955   0.3745   2.2814  
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -17.526842   2.437219  -7.191 6.42e-13 ***
## nox          29.523641   4.061820   7.269 3.63e-13 ***
## dis           0.179678   0.126542   1.420   0.1556    
## tax           0.002513   0.001254   2.004   0.0451 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 318.11  on 502  degrees of freedom
## AIC: 326.11
##
## Number of Fisher Scoring iterations: 6
```

```
lr.model3 <- glm(factor(crimrate) ~ nox + rad + tax, family = binomial(), data = bostonData1)
summary(lr.model3)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + rad + tax, family = binomial(),
##     data = bostonData1)
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.89749  -0.30647  -0.03087   0.00565   2.55953
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -19.837117   2.271908  -8.731  < 2e-16 ***
## nox          35.556283   4.317479   8.235  < 2e-16 ***
## rad           0.645415   0.112523   5.736  9.7e-09 ***
## tax          -0.008226   0.002261  -3.638 0.000275 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 249.99  on 502  degrees of freedom
## AIC: 257.99
##
## Number of Fisher Scoring iterations: 8
```

We see that when doing so we have that all the predictors show significance. Thus, for 3 predictors, the subset will include "nox" "rad" "tax"

4 predictors

We will try to examine the top 4 predictors first

```
lr.model4 <- glm(factor(crimrate) ~ nox + rad + tax + age, data = bostonData1, family = binomial())
summary(lr.model4)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + rad + tax + age, family = binomial(),
##     data = bostonData1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.87083  -0.29846  -0.02991   0.00686   2.60339
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -19.311621   2.325108  -8.306  < 2e-16 ***
## nox          33.430182   4.744348   7.046 1.84e-12 ***
## rad           0.649977   0.113335   5.735 9.75e-09 ***
## tax          -0.008413   0.002295  -3.666 0.000247 ***
## age           0.008830   0.008468   1.043 0.297053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 248.90  on 501  degrees of freedom
## AIC: 258.9
##
## Number of Fisher Scoring iterations: 8
```

```r
lr.model4 <- glm(factor(crimrate) ~ nox + rad + dis + age, data = bostonData1, family = binomial())
summary(lr.model4)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + rad + dis + age, family = binomial(),
##     data = bostonData1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.91089  -0.32118  -0.06026   0.00685   2.68470
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.549617   3.070757  -6.692 2.20e-11 ***
## nox          30.288586   4.963789   6.102 1.05e-09 ***
## rad           0.515626   0.100552   5.128 2.93e-07 ***
## dis           0.227752   0.138256   1.647   0.0995 .
## age           0.008644   0.008202   1.054   0.2919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 263.02  on 501  degrees of freedom
## AIC: 273.02
##
## Number of Fisher Scoring iterations: 8
```

```r
lr.model4 <- glm(factor(crimrate) ~ nox + rad + dis + tax, data = bostonData1, family = binomial())
summary(lr.model4)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + rad + dis + tax, family = binomial(),
##     data = bostonData1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.92984  -0.30227  -0.03637   0.00487   2.62973
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -23.187324   3.371179  -6.878 6.07e-12 ***
## nox          40.373851   5.620450   7.183 6.80e-13 ***
```

```
## rad           0.649013   0.112812   5.753 8.77e-09 ***
## dis           0.205545   0.143236   1.435 0.151286
## tax          -0.008203   0.002263  -3.625 0.000289 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 247.95  on 501  degrees of freedom
## AIC: 257.95
##
## Number of Fisher Scoring iterations: 8
```

```
boston.pred.2 <- predict(lr.model3, data = boston.train, newdata = boston.test)
boston.glm.pred.2 <- rep(0, length(boston.pred))
boston.glm.pred.2[boston.pred >= 0] <- 1

length(boston.glm.pred.2)
```

```
## [1] 101
```

```
# boston.test
table(boston.glm.pred.2, boston.test[,15])
```

```
##
## boston.glm.pred.2  0  1
##                 0 51 11
##                 1  2 37
```

```
#The misclassification rate is going to be calculated as
mean(boston.glm.pred.2 != boston.test[,15])
```

```
## [1] 0.1287129
```

The model with the predictors "nox" "rad" "tax" "age" performs the best with the lowest misclassification rate as well as only one of the predictors does not show statistical significance.

5 Predictors

```
lr.model5 <- glm(factor(crimrate) ~ nox + rad + tax + age + dis, data = bostonData1, family = binomial()
summary(lr.model5)
```

```
##
## Call:
## glm(formula = factor(crimrate) ~ nox + rad + tax + age + dis,
##     family = binomial(), data = bostonData1)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -1.86953  -0.29755  -0.03264   0.00457   2.69152
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -22.986629   3.374368  -6.812 9.62e-12 ***
## nox          38.367984   5.796181   6.620 3.60e-11 ***
## rad           0.653928   0.113718   5.750 8.90e-09 ***
## tax          -0.008408   0.002296  -3.661 0.000251 ***
## age           0.010977   0.008566   1.281 0.200057
## dis           0.234786   0.144945   1.620 0.105269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 246.30  on 500  degrees of freedom
## AIC: 258.3
##
## Number of Fisher Scoring iterations: 8
```

```
boston.pred3 <- predict(lr.model3, data = boston.train, newdata = boston.test)
boston.glm.pred3 <- rep(0, length(boston.pred3))
boston.glm.pred3[boston.pred >= 0] <- 1

length(boston.glm.pred3)
```

```
## [1] 101
```

```
# boston.test
table(boston.glm.pred3, boston.test[,15])
```

```
##
## boston.glm.pred3  0  1
##                0 51 11
##                1  2 37
```

```
#The misclassification rate is going to be calculated as
mean(boston.glm.pred3 != boston.test[,15])
```

```
## [1] 0.1287129
```

Above result with the five predictors nox, rad, dis, tax, age will generate the best model with five predictors. However, the numbers that are outputted seems very familiar. Yes, although the preditors itself was added to the model, the confusion matrix and the misclassification rate remains the same. Thus, it suffices to say that the additional predictors that are added will generate the same result and thus is only making it more difficult for the researchers as the model will become more complex.

knn models

3 predictors

k = 1

```
pred3 <- c(5,8,9)
boston.train.3.1 <- boston.train[, pred3]
boston.test.3.1 <- boston.test[,pred3]
boston.knn.3.1 <- knn(boston.train.3.1, boston.test.3.1, boston.train[,15],k = 1)
table(boston.knn.3.1, boston.test[,15])
```

```
##
## boston.knn.3.1  0  1
##              0 51  4
##              1  2 44
```

```
mean(boston.knn.3.1 != boston.test[,15])
```

```
## [1] 0.05940594
```

k = 3

```
boston.train.3.3 <- boston.train[, pred3]
boston.test.3.3 <- boston.test[,pred3]
boston.knn.3.3 <- knn(boston.train.3.3, boston.test.3.3, boston.train[,15], k = 3)
table(boston.knn.3.3, boston.test[,15])
```

```
##
## boston.knn.3.3  0  1
##              0 51  4
##              1  2 44
```

```
mean(boston.knn.3.3 != boston.test[,15])
```

```
## [1] 0.05940594
```

k = 5

```
boston.train.3.5 <- boston.train[, pred3]
boston.test.3.5 <- boston.test[,pred3]
boston.knn.3.5 <- knn(boston.train.3.5, boston.test.3.5, boston.train[,15],k = 5)
table(boston.knn.3.5, boston.test[,15])
```

```
##
## boston.knn.3.5  0  1
##              0 50  3
##              1  3 45
```

```r
mean(boston.knn.3.5 != boston.test[,15])
```

```
## [1] 0.05940594
```

k = 7

```r
boston.train.3.7 <- boston.train[, pred3]
boston.test.3.7 <- boston.test[,pred3]
boston.knn.3.7 <- knn(boston.train.3.7, boston.test.3.7, boston.train[,15],k = 7)
table(boston.knn.3.7, boston.test[,15])
```

```
##
## boston.knn.3.7  0  1
##              0 49  4
##              1  4 44
```

```r
mean(boston.knn.3.7 != boston.test[,15])
```

```
## [1] 0.07920792
```

k = 9

```r
boston.train.3.9 <- boston.train[, pred3]
boston.test.3.9 <- boston.test[,pred3]
boston.knn.3.9 <- knn(boston.train.3.9, boston.test.3.9, boston.train[,15],k = 9)
table(boston.knn.3.9, boston.test[,15])
```

```
##
## boston.knn.3.9  0  1
##              0 50  6
##              1  3 42
```

```r
mean(boston.knn.3.9 != boston.test[,15])
```

```
## [1] 0.08910891
```

k = 11

```r
boston.train.3.11 <- boston.train[, pred3]
boston.test.3.11 <- boston.test[,pred3]
boston.knn.3.11 <- knn(boston.train.3.11, boston.test.3.11, boston.train[,15],k = 11)
table(boston.knn.3.11, boston.test[,15])
```

```
##
## boston.knn.3.11  0  1
##               0 48  4
##               1  5 44
```

```r
mean(boston.knn.3.11 != boston.test[,15])
```

```
## [1] 0.08910891
```

The lowest misclassification rate occurs when k = 1, 3, and 5 for when there are 3 predictors.

4 predictors

k = 1

```r
pred4 <- c(5,8,9,10)
boston.train.4.1 <- boston.train[, pred4]
boston.test.4.1 <- boston.test[,pred4]
boston.knn.4.1 <- knn(boston.train.4.1, boston.test.4.1, boston.train[,15],k = 1)
table(boston.knn.4.1, boston.test[,15])
```

```
##
## boston.knn.4.1  0  1
##              0 51  1
##              1  2 47
```

```r
mean(boston.knn.4.1 != boston.test[,15])
```

```
## [1] 0.02970297
```

k = 3

```r
pred4 <- c(5,8,9,10)
boston.train.4.3 <- boston.train[, pred4]
boston.test.4.3 <- boston.test[,pred4]
boston.knn.4.3 <- knn(boston.train.4.3, boston.test.4.3, boston.train[,15],k = 3)
table(boston.knn.4.3, boston.test[,15])
```

```
##
## boston.knn.4.3  0  1
##              0 51  2
##              1  2 46
```

```r
mean(boston.knn.4.3 != boston.test[,15])
```

```
## [1] 0.03960396
```

k = 5

```r
pred4 <- c(5,8,9,10)
boston.train.4.5 <- boston.train[, pred4]
boston.test.4.5 <- boston.test[,pred4]
boston.knn.4.5 <- knn(boston.train.4.5, boston.test.4.5, boston.train[,15],k = 5)
table(boston.knn.4.5, boston.test[,15])
```

```
##
## boston.knn.4.5  0  1
##              0 52  3
##              1  1 45
```

```r
mean(boston.knn.4.5 != boston.test[,15])
```

```
## [1] 0.03960396
```

k = 7

```r
pred4 <- c(5,8,9,10)
boston.train.4.7 <- boston.train[, pred4]
boston.test.4.7 <- boston.test[,pred4]
boston.knn.4.7 <- knn(boston.train.4.7, boston.test.4.7, boston.train[,15],k = 7)
table(boston.knn.4.7, boston.test[,15])
```

```
##
## boston.knn.4.7  0  1
##              0 51  3
##              1  2 45
```

```r
mean(boston.knn.4.7 != boston.test[,15])
```

```
## [1] 0.04950495
```

k = 9

```r
pred4 <- c(5,8,9,10)
boston.train.4.9 <- boston.train[, pred4]
boston.test.4.9 <- boston.test[,pred4]
boston.knn.4.9 <- knn(boston.train.4.9, boston.test.4.9, boston.train[,15],k = 9)
table(boston.knn.4.9, boston.test[,15])
```

```
##
## boston.knn.4.9  0  1
##              0 51  7
##              1  2 41
```

```r
mean(boston.knn.4.9 != boston.test[,15])
```

```
## [1] 0.08910891
```

k = 11

```r
pred4 <- c(5,8,9,10)
boston.train.4.11 <- boston.train[, pred4]
boston.test.4.11 <- boston.test[,pred4]
boston.knn.4.11 <- knn(boston.train.4.11, boston.test.4.11, boston.train[,15],k = 11)
table(boston.knn.4.11, boston.test[,15])
```

```
##
## boston.knn.4.11  0  1
##               0 51  7
##               1  2 41
```

```r
mean(boston.knn.4.11 != boston.test[,15])
```

```
## [1] 0.08910891
```

The lowest misclassification rate occurs when k = 1 for when there are 4 predictors.

5 predictors

k = 1

```r
pred5 <- c(5,7,8,9,10)
boston.train.5.1 <- boston.train[, pred5]
boston.test.5.1 <- boston.test[,pred5]
boston.knn.5.1 <- knn(boston.train.5.1, boston.test.5.1, boston.train[,15],k = 1)
table(boston.knn.5.1, boston.test[,15])
```

```
##
## boston.knn.5.1  0  1
##              0 50  5
##              1  3 43
```

```r
mean(boston.knn.5.1 != boston.test[,15])
```

```
## [1] 0.07920792
```

k = 3

```r
pred5 <- c(5,7,8,9,10)
boston.train.5.3 <- boston.train[, pred5]
boston.test.5.3 <- boston.test[,pred5]
boston.knn.5.3 <- knn(boston.train.5.3, boston.test.5.3, boston.train[,15],k = 3)
table(boston.knn.5.3, boston.test[,15])
```

```
##
## boston.knn.5.3  0  1
##              0 52  4
##              1  1 44
```

```r
mean(boston.knn.5.3 != boston.test[,15])
```

```
## [1] 0.04950495
```

k = 5

```r
pred5 <- c(5,7,8,9,10)
boston.train.5.5 <- boston.train[, pred5]
boston.test.5.5 <- boston.test[,pred5]
boston.knn.5.5 <- knn(boston.train.5.5, boston.test.5.5, boston.train[,15],k = 5)
table(boston.knn.5.5, boston.test[,15])
```

```
##
## boston.knn.5.5  0  1
##              0 52  4
##              1  1 44
```

```r
mean(boston.knn.5.5 != boston.test[,15])
```

```
## [1] 0.04950495
```

k = 7

```r
pred5 <- c(5,7,8,9,10)
boston.train.5.7 <- boston.train[, pred5]
boston.test.5.7 <- boston.test[,pred5]
boston.knn.5.7 <- knn(boston.train.5.7, boston.test.5.7, boston.train[,15],k = 7)
table(boston.knn.5.7, boston.test[,15])
```

```
##
## boston.knn.5.7  0  1
##              0 52  4
##              1  1 44
```

```
mean(boston.knn.5.7 != boston.test[,15])
```

```
## [1] 0.04950495
```

k = 9

```
pred5 <- c(5,7,8,9,10)
boston.train.5.9 <- boston.train[, pred5]
boston.test.5.9 <- boston.test[,pred5]
boston.knn.5.9 <- knn(boston.train.5.9, boston.test.5.9, boston.train[,15],k = 9)
table(boston.knn.5.9, boston.test[,15])
```

```
##
## boston.knn.5.9  0  1
##              0 52  4
##              1  1 44
```

```
mean(boston.knn.5.9 != boston.test[,15])
```

```
## [1] 0.04950495
```

k = 11

```
pred5 <- c(5,7,8,9,10)
boston.train.5.11 <- boston.train[, pred5]
boston.test.5.11 <- boston.test[,pred5]
boston.knn.5.11 <- knn(boston.train.5.11, boston.test.5.11, boston.train[,15],k = 11)
table(boston.knn.5.11, boston.test[,15])
```

```
##
## boston.knn.5.11  0  1
##               0 52  4
##               1  1 44
```

```
mean(boston.knn.5.11 != boston.test[,15])
```

```
## [1] 0.04950495
```

The lowest misclassification rate occurs when k = 3,5,7,9,11 for when there are 5 predictors.