Week Three Exercises

Adeline Casali

2023-11-07

# Question 13

Load packages

library(ISLR2)  
library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':  
##   
## Boston

library(e1071)  
library(class)

1. Below are some numerical and graphical summaries of the Weekly dataset. There do not seem to be any correlated variables, but there does seem to be an overall increase in the volume of trades over time.

names(Weekly)

## [1] "Year" "Lag1" "Lag2" "Lag3" "Lag4" "Lag5"   
## [7] "Volume" "Today" "Direction"

dim(Weekly)

## [1] 1089 9

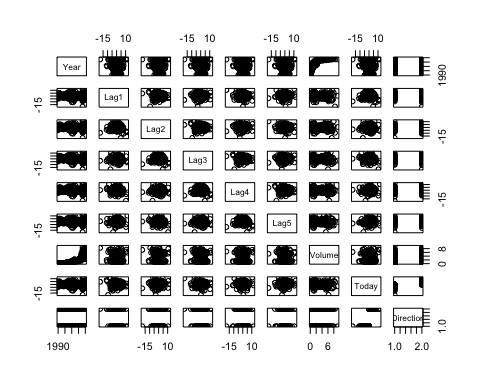
summary(Weekly)

## Year Lag1 Lag2 Lag3   
## Min. :1990 Min. :-18.1950 Min. :-18.1950 Min. :-18.1950   
## 1st Qu.:1995 1st Qu.: -1.1540 1st Qu.: -1.1540 1st Qu.: -1.1580   
## Median :2000 Median : 0.2410 Median : 0.2410 Median : 0.2410   
## Mean :2000 Mean : 0.1506 Mean : 0.1511 Mean : 0.1472   
## 3rd Qu.:2005 3rd Qu.: 1.4050 3rd Qu.: 1.4090 3rd Qu.: 1.4090   
## Max. :2010 Max. : 12.0260 Max. : 12.0260 Max. : 12.0260   
## Lag4 Lag5 Volume Today   
## Min. :-18.1950 Min. :-18.1950 Min. :0.08747 Min. :-18.1950   
## 1st Qu.: -1.1580 1st Qu.: -1.1660 1st Qu.:0.33202 1st Qu.: -1.1540   
## Median : 0.2380 Median : 0.2340 Median :1.00268 Median : 0.2410   
## Mean : 0.1458 Mean : 0.1399 Mean :1.57462 Mean : 0.1499   
## 3rd Qu.: 1.4090 3rd Qu.: 1.4050 3rd Qu.:2.05373 3rd Qu.: 1.4050   
## Max. : 12.0260 Max. : 12.0260 Max. :9.32821 Max. : 12.0260   
## Direction   
## Down:484   
## Up :605   
##   
##   
##   
##

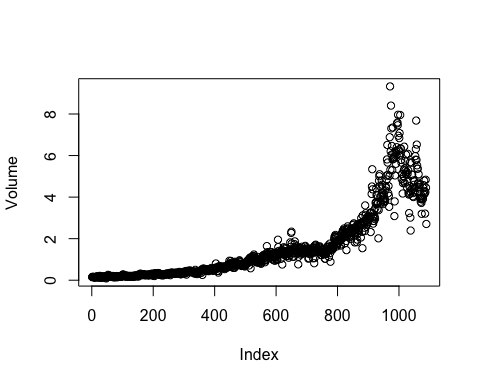
cor(Weekly[, -9])

## Year Lag1 Lag2 Lag3 Lag4  
## Year 1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923  
## Lag1 -0.03228927 1.000000000 -0.07485305 0.05863568 -0.071273876  
## Lag2 -0.03339001 -0.074853051 1.00000000 -0.07572091 0.058381535  
## Lag3 -0.03000649 0.058635682 -0.07572091 1.00000000 -0.075395865  
## Lag4 -0.03112792 -0.071273876 0.05838153 -0.07539587 1.000000000  
## Lag5 -0.03051910 -0.008183096 -0.07249948 0.06065717 -0.075675027  
## Volume 0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617  
## Today -0.03245989 -0.075031842 0.05916672 -0.07124364 -0.007825873  
## Lag5 Volume Today  
## Year -0.030519101 0.84194162 -0.032459894  
## Lag1 -0.008183096 -0.06495131 -0.075031842  
## Lag2 -0.072499482 -0.08551314 0.059166717  
## Lag3 0.060657175 -0.06928771 -0.071243639  
## Lag4 -0.075675027 -0.06107462 -0.007825873  
## Lag5 1.000000000 -0.05851741 0.011012698  
## Volume -0.058517414 1.00000000 -0.033077783  
## Today 0.011012698 -0.03307778 1.000000000

pairs(Weekly)



attach(Weekly)  
plot(Volume)

 b) According to the logistic regression model created below, only Lag2 is considered to have a statistically significant relationship with Direction. Lag1 is the second closest variable to being significant, with a p-value of 0.1181.

weekly\_log <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomial)  
summary(weekly\_log)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +   
## Volume, family = binomial, data = Weekly)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.26686 0.08593 3.106 0.0019 \*\*  
## Lag1 -0.04127 0.02641 -1.563 0.1181   
## Lag2 0.05844 0.02686 2.175 0.0296 \*   
## Lag3 -0.01606 0.02666 -0.602 0.5469   
## Lag4 -0.02779 0.02646 -1.050 0.2937   
## Lag5 -0.01447 0.02638 -0.549 0.5833   
## Volume -0.02274 0.03690 -0.616 0.5377   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1496.2 on 1088 degrees of freedom  
## Residual deviance: 1486.4 on 1082 degrees of freedom  
## AIC: 1500.4  
##   
## Number of Fisher Scoring iterations: 4

1. This matrix shows that the logistic regression model correctly predicts when the market goes down 54 times, and up 557 times. Up was incorrectly predicted 430 times (False positive, type I error), and down was incorrectly predicted 48 times (False negative, type II error). This equates to correctly predicting direction 611 times out of 1089, or a rate of 56.1%. This also means a training error rate of 43.9%.

weekly\_probs\_log <- predict(weekly\_log, type = "response")  
weekly\_pred\_log <- rep("Down", 1089)  
weekly\_pred\_log[weekly\_probs\_log > 0.5] = "Up"  
table(weekly\_pred\_log, Direction)

## Direction  
## weekly\_pred\_log Down Up  
## Down 54 48  
## Up 430 557

mean(weekly\_pred\_log == Direction)

## [1] 0.5610652

1. The logistic regression model produced with only Lag2 correctly predicts the direction of the market 62.5% of the time with the test data, which equates to a test error rate of 37.5%.

weekly\_train <- (Year < 2009)  
weekly\_test <- Weekly[!weekly\_train, ]  
dim(weekly\_test)

## [1] 104 9

direction\_test <- Direction[!weekly\_train]  
weekly\_log2 <- glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = weekly\_train)  
weekly\_probs\_log2 <- predict(weekly\_log2, weekly\_test, type = "response")  
weekly\_pred\_log2 <- rep("Down", 104)  
weekly\_pred\_log2[weekly\_probs\_log2 > 0.5] = "Up"  
table(weekly\_pred\_log2, direction\_test)

## direction\_test  
## weekly\_pred\_log2 Down Up  
## Down 9 5  
## Up 34 56

mean(weekly\_pred\_log2 == direction\_test)

## [1] 0.625

mean(weekly\_pred\_log2 != direction\_test)

## [1] 0.375

1. The LDA model produced with only Lag2 correctly predicts the direction of the market 62.5% of the time with the test data, which equates to a test error rate of 37.5%.

weekly\_lda <- lda(Direction ~ Lag2, data = Weekly, subset = weekly\_train)  
weekly\_lda

## Call:  
## lda(Direction ~ Lag2, data = Weekly, subset = weekly\_train)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag2  
## Down -0.03568254  
## Up 0.26036581  
##   
## Coefficients of linear discriminants:  
## LD1  
## Lag2 0.4414162

weekly\_pred\_lda <- predict(weekly\_lda, weekly\_test)  
names(weekly\_pred\_lda)

## [1] "class" "posterior" "x"

weekly\_lda\_class <- weekly\_pred\_lda$class  
table(weekly\_lda\_class, direction\_test)

## direction\_test  
## weekly\_lda\_class Down Up  
## Down 9 5  
## Up 34 56

mean(weekly\_lda\_class == direction\_test)

## [1] 0.625

mean(weekly\_lda\_class != direction\_test)

## [1] 0.375

1. The QDA model with only Lag2 correctly predicts the direction of the market 58.7% of the time with the test data, which equates to a test error rate of 41.3%.

weekly\_qda <- qda(Direction ~ Lag2, data = Weekly, subset = weekly\_train)  
weekly\_qda

## Call:  
## qda(Direction ~ Lag2, data = Weekly, subset = weekly\_train)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag2  
## Down -0.03568254  
## Up 0.26036581

weekly\_qda\_class <- predict(weekly\_qda, weekly\_test)$class  
table(weekly\_qda\_class, direction\_test)

## direction\_test  
## weekly\_qda\_class Down Up  
## Down 0 0  
## Up 43 61

mean(weekly\_qda\_class == direction\_test)

## [1] 0.5865385

mean(weekly\_qda\_class != direction\_test)

## [1] 0.4134615

1. The KNN model with K = 1 produced with only Lag2 correctly predicts the direction of the market 50% of the time with the test data, which equates to a test error rate of 50%.

train <- (Weekly$Year < 2009)  
train.X <- as.matrix(cbind(Lag2 = Weekly$Lag2[train]))  
test.X <- as.matrix(cbind(Lag2 = Weekly$Lag2[!train]))  
train.Direction <- Weekly$Direction[train]  
test.Direction <- Weekly$Direction[!train]  
  
set.seed(1)  
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)  
table(knn.pred, test.Direction)

## test.Direction  
## knn.pred Down Up  
## Down 21 30  
## Up 22 31

mean(knn.pred == test.Direction)

## [1] 0.5

mean(knn.pred != test.Direction)

## [1] 0.5

1. The naive Bayes model produced with only Lag2 correctly predicts the direction of the market 58.7% of the time with the test data, which equates to a test error rate of 41.3%.

weekly\_nb <- naiveBayes(Direction ~ Lag2, data = Weekly, subset = weekly\_train)  
weekly\_nb

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## Down Up   
## 0.4477157 0.5522843   
##   
## Conditional probabilities:  
## Lag2  
## Y [,1] [,2]  
## Down -0.03568254 2.199504  
## Up 0.26036581 2.317485

weekly\_nb\_class <- predict(weekly\_nb, direction\_test)

## Warning in predict.naiveBayes(weekly\_nb, direction\_test): Type mismatch between  
## training and new data for variable 'Lag2'. Did you use factors with numeric  
## labels for training, and numeric values for new data?

table(weekly\_nb\_class, direction\_test)

## direction\_test  
## weekly\_nb\_class Down Up  
## Down 0 0  
## Up 43 61

mean(weekly\_nb\_class == direction\_test)

## [1] 0.5865385

mean(weekly\_nb\_class != direction\_test)

## [1] 0.4134615

1. Both the logistic regression and LDA methods appeared to provide the best results on this data, with accuracy rates of 62.5% each.
2. After experimenting with many different combinations of predictors and transformations, the methods that appear to provide the best results are the logistic regression and LDA methods with only the Lag2 predictor, with success results of 62.5% each. Adding in additional predictors and transformations only made for worse accuracy in all methods. Additionally, using K = 20 had the best results with the KNN method at 58.7%, which is still less than the logistic and LDA methods.

# KNN with K = 3  
set.seed(1)  
knn.pred <- knn(train.X, test.X, train.Direction, k = 3)  
table(knn.pred, test.Direction)

## test.Direction  
## knn.pred Down Up  
## Down 16 20  
## Up 27 41

mean(knn.pred == test.Direction)

## [1] 0.5480769

mean(knn.pred != test.Direction)

## [1] 0.4519231

# KNN with K = 10  
set.seed(1)  
knn.pred <- knn(train.X, test.X, train.Direction, k = 10)  
table(knn.pred, test.Direction)

## test.Direction  
## knn.pred Down Up  
## Down 17 21  
## Up 26 40

mean(knn.pred == test.Direction)

## [1] 0.5480769

mean(knn.pred != test.Direction)

## [1] 0.4519231

# KNN with K = 20  
set.seed(1)  
knn.pred <- knn(train.X, test.X, train.Direction, k = 20)  
table(knn.pred, test.Direction)

## test.Direction  
## knn.pred Down Up  
## Down 21 21  
## Up 22 40

mean(knn.pred == test.Direction)

## [1] 0.5865385

mean(knn.pred != test.Direction)

## [1] 0.4134615

# KNN with K = 30  
set.seed(1)  
knn.pred <- knn(train.X, test.X, train.Direction, k = 30)  
table(knn.pred, test.Direction)

## test.Direction  
## knn.pred Down Up  
## Down 20 24  
## Up 23 37

mean(knn.pred == test.Direction)

## [1] 0.5480769

mean(knn.pred != test.Direction)

## [1] 0.4519231

# Logistic regression with Lag1 and Lag2  
weekly\_log3 <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial, subset = weekly\_train)  
weekly\_probs\_log3 <- predict(weekly\_log3, weekly\_test, type = "response")  
weekly\_pred\_log3 <- rep("Down", 104)  
weekly\_pred\_log3[weekly\_probs\_log3 > 0.5] = "Up"  
table(weekly\_pred\_log3, direction\_test)

## direction\_test  
## weekly\_pred\_log3 Down Up  
## Down 7 8  
## Up 36 53

mean(weekly\_pred\_log3 == direction\_test)

## [1] 0.5769231

mean(weekly\_pred\_log3 != direction\_test)

## [1] 0.4230769

# Logistic regression with Lag1, Lag2, and Volume squared  
weekly\_log3 <- glm(Direction ~ Lag1 + Lag2 + Volume^2, data = Weekly, family = binomial, subset = weekly\_train)  
weekly\_probs\_log3 <- predict(weekly\_log3, weekly\_test, type = "response")  
weekly\_pred\_log3 <- rep("Down", 104)  
weekly\_pred\_log3[weekly\_probs\_log3 > 0.5] = "Up"  
table(weekly\_pred\_log3, direction\_test)

## direction\_test  
## weekly\_pred\_log3 Down Up  
## Down 27 33  
## Up 16 28

mean(weekly\_pred\_log3 == direction\_test)

## [1] 0.5288462

mean(weekly\_pred\_log3 != direction\_test)

## [1] 0.4711538

# LDA with Lag1 and Lag2  
weekly\_lda2 <- lda(Direction ~ Lag1 + Lag2, data = Weekly, subset = weekly\_train)  
weekly\_lda2

## Call:  
## lda(Direction ~ Lag1 + Lag2, data = Weekly, subset = weekly\_train)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag1 Lag2  
## Down 0.289444444 -0.03568254  
## Up -0.009213235 0.26036581  
##   
## Coefficients of linear discriminants:  
## LD1  
## Lag1 -0.3013148  
## Lag2 0.2982579

weekly\_pred\_lda2 <- predict(weekly\_lda2, weekly\_test)  
names(weekly\_pred\_lda2)

## [1] "class" "posterior" "x"

weekly\_lda\_class2 <- weekly\_pred\_lda2$class  
table(weekly\_lda\_class2, direction\_test)

## direction\_test  
## weekly\_lda\_class2 Down Up  
## Down 7 8  
## Up 36 53

mean(weekly\_lda\_class2 == direction\_test)

## [1] 0.5769231

mean(weekly\_lda\_class2 != direction\_test)

## [1] 0.4230769

# LDA with Lag1, Lag2, and Volume squared  
weekly\_lda2 <- lda(Direction ~ Lag1 + Lag2 + Volume^2, data = Weekly, subset = weekly\_train)  
weekly\_lda2

## Call:  
## lda(Direction ~ Lag1 + Lag2 + Volume^2, data = Weekly, subset = weekly\_train)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag1 Lag2 Volume  
## Down 0.289444444 -0.03568254 1.266966  
## Up -0.009213235 0.26036581 1.156529  
##   
## Coefficients of linear discriminants:  
## LD1  
## Lag1 -0.2979204  
## Lag2 0.2366224  
## Volume -0.3545069

weekly\_pred\_lda2 <- predict(weekly\_lda2, weekly\_test)  
names(weekly\_pred\_lda2)

## [1] "class" "posterior" "x"

weekly\_lda\_class2 <- weekly\_pred\_lda2$class  
table(weekly\_lda\_class2, direction\_test)

## direction\_test  
## weekly\_lda\_class2 Down Up  
## Down 27 33  
## Up 16 28

mean(weekly\_lda\_class2 == direction\_test)

## [1] 0.5288462

mean(weekly\_lda\_class2 != direction\_test)

## [1] 0.4711538

# QDA with Lag1 and Lag2  
weekly\_qda2 <- qda(Direction ~ Lag1 + Lag2, data = Weekly, subset = weekly\_train)  
weekly\_qda2

## Call:  
## qda(Direction ~ Lag1 + Lag2, data = Weekly, subset = weekly\_train)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag1 Lag2  
## Down 0.289444444 -0.03568254  
## Up -0.009213235 0.26036581

weekly\_qda\_class2 <- predict(weekly\_qda2, weekly\_test)$class  
table(weekly\_qda\_class2, direction\_test)

## direction\_test  
## weekly\_qda\_class2 Down Up  
## Down 7 10  
## Up 36 51

mean(weekly\_qda\_class2 == direction\_test)

## [1] 0.5576923

mean(weekly\_qda\_class2 != direction\_test)

## [1] 0.4423077

# QDA with Lag1, Lag2, and Volume squared  
weekly\_qda2 <- qda(Direction ~ Lag1 + Lag2 + Volume^2, data = Weekly, subset = weekly\_train)  
weekly\_qda2

## Call:  
## qda(Direction ~ Lag1 + Lag2 + Volume^2, data = Weekly, subset = weekly\_train)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag1 Lag2 Volume  
## Down 0.289444444 -0.03568254 1.266966  
## Up -0.009213235 0.26036581 1.156529

weekly\_qda\_class2 <- predict(weekly\_qda2, weekly\_test)$class  
table(weekly\_qda\_class2, direction\_test)

## direction\_test  
## weekly\_qda\_class2 Down Up  
## Down 31 44  
## Up 12 17

mean(weekly\_qda\_class2 == direction\_test)

## [1] 0.4615385

mean(weekly\_qda\_class2 != direction\_test)

## [1] 0.5384615

# Naive Bayes with Lag1 and Lag2  
weekly\_nb2 <- naiveBayes(Direction ~ Lag1 + Lag2, data = Weekly, subset = weekly\_train)  
weekly\_nb2

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## Down Up   
## 0.4477157 0.5522843   
##   
## Conditional probabilities:  
## Lag1  
## Y [,1] [,2]  
## Down 0.289444444 2.211721  
## Up -0.009213235 2.308387  
##   
## Lag2  
## Y [,1] [,2]  
## Down -0.03568254 2.199504  
## Up 0.26036581 2.317485

weekly\_nb\_class2 <- predict(weekly\_nb2, direction\_test)

## Warning in predict.naiveBayes(weekly\_nb2, direction\_test): Type mismatch  
## between training and new data for variable 'Lag1'. Did you use factors with  
## numeric labels for training, and numeric values for new data?

## Warning in predict.naiveBayes(weekly\_nb2, direction\_test): Type mismatch  
## between training and new data for variable 'Lag2'. Did you use factors with  
## numeric labels for training, and numeric values for new data?

table(weekly\_nb\_class2, direction\_test)

## direction\_test  
## weekly\_nb\_class2 Down Up  
## Down 0 0  
## Up 43 61

mean(weekly\_nb\_class2 == direction\_test)

## [1] 0.5865385

mean(weekly\_nb\_class2 != direction\_test)

## [1] 0.4134615

# Naive Bayes with Lag1, Lag2, and Volume squared  
weekly\_nb2 <- naiveBayes(Direction ~ Lag1 + Lag2 + Volume^2, data = Weekly, subset = weekly\_train)  
weekly\_nb2

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## Down Up   
## 0.4477157 0.5522843   
##   
## Conditional probabilities:  
## Lag1  
## Y [,1] [,2]  
## Down 0.289444444 2.211721  
## Up -0.009213235 2.308387  
##   
## Lag2  
## Y [,1] [,2]  
## Down -0.03568254 2.199504  
## Up 0.26036581 2.317485  
##   
## Volume  
## Y [,1] [,2]  
## Down 1.266966 1.320152  
## Up 1.156529 1.204614

weekly\_nb\_class2 <- predict(weekly\_nb2, direction\_test)

## Warning in predict.naiveBayes(weekly\_nb2, direction\_test): Type mismatch  
## between training and new data for variable 'Lag1'. Did you use factors with  
## numeric labels for training, and numeric values for new data?  
  
## Warning in predict.naiveBayes(weekly\_nb2, direction\_test): Type mismatch  
## between training and new data for variable 'Lag2'. Did you use factors with  
## numeric labels for training, and numeric values for new data?

## Warning in predict.naiveBayes(weekly\_nb2, direction\_test): Type mismatch  
## between training and new data for variable 'Volume'. Did you use factors with  
## numeric labels for training, and numeric values for new data?

table(weekly\_nb\_class2, direction\_test)

## direction\_test  
## weekly\_nb\_class2 Down Up  
## Down 0 0  
## Up 43 61

mean(weekly\_nb\_class2 == direction\_test)

## [1] 0.5865385

mean(weekly\_nb\_class2 != direction\_test)

## [1] 0.4134615

# Question 16

Exploring the dataset

names(Boston)

## [1] "crim" "zn" "indus" "chas" "nox" "rm" "age"   
## [8] "dis" "rad" "tax" "ptratio" "black" "lstat" "medv"

dim(Boston)

## [1] 506 14

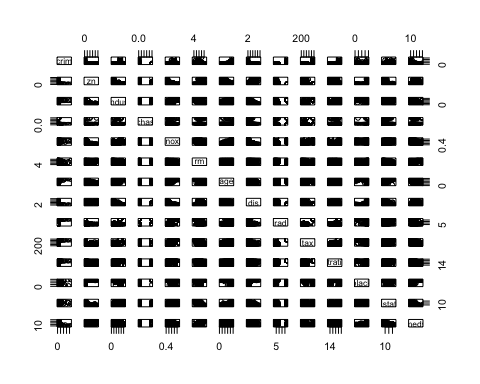
summary(Boston)

## crim zn indus chas   
## Min. : 0.00632 Min. : 0.00 Min. : 0.46 Min. :0.00000   
## 1st Qu.: 0.08205 1st Qu.: 0.00 1st Qu.: 5.19 1st Qu.:0.00000   
## Median : 0.25651 Median : 0.00 Median : 9.69 Median :0.00000   
## Mean : 3.61352 Mean : 11.36 Mean :11.14 Mean :0.06917   
## 3rd Qu.: 3.67708 3rd Qu.: 12.50 3rd Qu.:18.10 3rd Qu.:0.00000   
## Max. :88.97620 Max. :100.00 Max. :27.74 Max. :1.00000   
## nox rm age dis   
## Min. :0.3850 Min. :3.561 Min. : 2.90 Min. : 1.130   
## 1st Qu.:0.4490 1st Qu.:5.886 1st Qu.: 45.02 1st Qu.: 2.100   
## Median :0.5380 Median :6.208 Median : 77.50 Median : 3.207   
## Mean :0.5547 Mean :6.285 Mean : 68.57 Mean : 3.795   
## 3rd Qu.:0.6240 3rd Qu.:6.623 3rd Qu.: 94.08 3rd Qu.: 5.188   
## Max. :0.8710 Max. :8.780 Max. :100.00 Max. :12.127   
## rad tax ptratio black   
## Min. : 1.000 Min. :187.0 Min. :12.60 Min. : 0.32   
## 1st Qu.: 4.000 1st Qu.:279.0 1st Qu.:17.40 1st Qu.:375.38   
## Median : 5.000 Median :330.0 Median :19.05 Median :391.44   
## Mean : 9.549 Mean :408.2 Mean :18.46 Mean :356.67   
## 3rd Qu.:24.000 3rd Qu.:666.0 3rd Qu.:20.20 3rd Qu.:396.23   
## Max. :24.000 Max. :711.0 Max. :22.00 Max. :396.90   
## lstat medv   
## Min. : 1.73 Min. : 5.00   
## 1st Qu.: 6.95 1st Qu.:17.02   
## Median :11.36 Median :21.20   
## Mean :12.65 Mean :22.53   
## 3rd Qu.:16.95 3rd Qu.:25.00   
## Max. :37.97 Max. :50.00

cor(Boston)

## crim zn indus chas nox  
## crim 1.00000000 -0.20046922 0.40658341 -0.055891582 0.42097171  
## zn -0.20046922 1.00000000 -0.53382819 -0.042696719 -0.51660371  
## indus 0.40658341 -0.53382819 1.00000000 0.062938027 0.76365145  
## chas -0.05589158 -0.04269672 0.06293803 1.000000000 0.09120281  
## nox 0.42097171 -0.51660371 0.76365145 0.091202807 1.00000000  
## rm -0.21924670 0.31199059 -0.39167585 0.091251225 -0.30218819  
## age 0.35273425 -0.56953734 0.64477851 0.086517774 0.73147010  
## dis -0.37967009 0.66440822 -0.70802699 -0.099175780 -0.76923011  
## rad 0.62550515 -0.31194783 0.59512927 -0.007368241 0.61144056  
## tax 0.58276431 -0.31456332 0.72076018 -0.035586518 0.66802320  
## ptratio 0.28994558 -0.39167855 0.38324756 -0.121515174 0.18893268  
## black -0.38506394 0.17552032 -0.35697654 0.048788485 -0.38005064  
## lstat 0.45562148 -0.41299457 0.60379972 -0.053929298 0.59087892  
## medv -0.38830461 0.36044534 -0.48372516 0.175260177 -0.42732077  
## rm age dis rad tax ptratio  
## crim -0.21924670 0.35273425 -0.37967009 0.625505145 0.58276431 0.2899456  
## zn 0.31199059 -0.56953734 0.66440822 -0.311947826 -0.31456332 -0.3916785  
## indus -0.39167585 0.64477851 -0.70802699 0.595129275 0.72076018 0.3832476  
## chas 0.09125123 0.08651777 -0.09917578 -0.007368241 -0.03558652 -0.1215152  
## nox -0.30218819 0.73147010 -0.76923011 0.611440563 0.66802320 0.1889327  
## rm 1.00000000 -0.24026493 0.20524621 -0.209846668 -0.29204783 -0.3555015  
## age -0.24026493 1.00000000 -0.74788054 0.456022452 0.50645559 0.2615150  
## dis 0.20524621 -0.74788054 1.00000000 -0.494587930 -0.53443158 -0.2324705  
## rad -0.20984667 0.45602245 -0.49458793 1.000000000 0.91022819 0.4647412  
## tax -0.29204783 0.50645559 -0.53443158 0.910228189 1.00000000 0.4608530  
## ptratio -0.35550149 0.26151501 -0.23247054 0.464741179 0.46085304 1.0000000  
## black 0.12806864 -0.27353398 0.29151167 -0.444412816 -0.44180801 -0.1773833  
## lstat -0.61380827 0.60233853 -0.49699583 0.488676335 0.54399341 0.3740443  
## medv 0.69535995 -0.37695457 0.24992873 -0.381626231 -0.46853593 -0.5077867  
## black lstat medv  
## crim -0.38506394 0.4556215 -0.3883046  
## zn 0.17552032 -0.4129946 0.3604453  
## indus -0.35697654 0.6037997 -0.4837252  
## chas 0.04878848 -0.0539293 0.1752602  
## nox -0.38005064 0.5908789 -0.4273208  
## rm 0.12806864 -0.6138083 0.6953599  
## age -0.27353398 0.6023385 -0.3769546  
## dis 0.29151167 -0.4969958 0.2499287  
## rad -0.44441282 0.4886763 -0.3816262  
## tax -0.44180801 0.5439934 -0.4685359  
## ptratio -0.17738330 0.3740443 -0.5077867  
## black 1.00000000 -0.3660869 0.3334608  
## lstat -0.36608690 1.0000000 -0.7376627  
## medv 0.33346082 -0.7376627 1.0000000

pairs(Boston)

 Creating a column denoting if a suburb is above or below the median crime rate.

median\_crim <- median(Boston$crim)  
Boston$med\_crime <- ifelse(Boston$crim > median\_crim, 1, 0)

According to this logistic regression containing all variables, the predictors zn, nox, dis, rad, tax, ptratio, black, and medv have a statistically significant relationship with whether or not a suburb is above the median crime rate.

boston\_log <- glm(med\_crime ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + black + lstat + medv, data = Boston, family = binomial)  
summary(boston\_log)

##   
## Call:  
## glm(formula = med\_crime ~ zn + indus + chas + nox + rm + age +   
## dis + rad + tax + ptratio + black + lstat + medv, family = binomial,   
## data = Boston)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -34.103704 6.530014 -5.223 1.76e-07 \*\*\*  
## zn -0.079918 0.033731 -2.369 0.01782 \*   
## indus -0.059389 0.043722 -1.358 0.17436   
## chas 0.785327 0.728930 1.077 0.28132   
## nox 48.523782 7.396497 6.560 5.37e-11 \*\*\*  
## rm -0.425596 0.701104 -0.607 0.54383   
## age 0.022172 0.012221 1.814 0.06963 .   
## dis 0.691400 0.218308 3.167 0.00154 \*\*   
## rad 0.656465 0.152452 4.306 1.66e-05 \*\*\*  
## tax -0.006412 0.002689 -2.385 0.01709 \*   
## ptratio 0.368716 0.122136 3.019 0.00254 \*\*   
## black -0.013524 0.006536 -2.069 0.03853 \*   
## lstat 0.043862 0.048981 0.895 0.37052   
## medv 0.167130 0.066940 2.497 0.01254 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 701.46 on 505 degrees of freedom  
## Residual deviance: 211.93 on 492 degrees of freedom  
## AIC: 239.93  
##   
## Number of Fisher Scoring iterations: 9

A logistic regression model with the predictors deemed statistically significant shows an accuracy of 93.6%, with an error of only 6.4%.

boston\_train <- Boston[1:350, ]  
boston\_test <- Boston[-(1:350), ]  
med\_crime\_test <- Boston$med\_crime[-(1:350)]  
boston\_log2 <- glm(med\_crime ~ zn + nox + dis + rad + tax + ptratio + black + medv, data = boston\_train, family = binomial)  
boston\_probs\_log2 <- predict(boston\_log2, boston\_test, type = "response")  
boston\_pred\_log2 <- rep(0, nrow(boston\_test))  
boston\_pred\_log2[boston\_probs\_log2 > 0.5] = 1  
table(boston\_pred\_log2, med\_crime\_test)

## med\_crime\_test  
## boston\_pred\_log2 0 1  
## 0 11 0  
## 1 10 135

mean(boston\_pred\_log2 == med\_crime\_test)

## [1] 0.9358974

mean(boston\_pred\_log2 != med\_crime\_test)

## [1] 0.06410256

An LDA model with the same predictors has an accuracy of 92.9% with an error of 7.1%.

boston\_lda <- lda(med\_crime ~ zn + nox + dis + rad + tax + ptratio + black + medv, data = boston\_train)  
boston\_lda

## Call:  
## lda(med\_crime ~ zn + nox + dis + rad + tax + ptratio + black +   
## medv, data = boston\_train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.6628571 0.3371429   
##   
## Group means:  
## zn nox dis rad tax ptratio black medv  
## 0 21.706897 0.4648004 5.133497 4.202586 295.1121 17.72284 388.9606 25.48491  
## 1 2.576271 0.6017458 2.982705 5.033898 340.0847 17.66102 363.6945 24.25085  
##   
## Coefficients of linear discriminants:  
## LD1  
## zn -0.001451496  
## nox 11.972332681  
## dis -0.040371825  
## rad 0.155227812  
## tax 0.002272409  
## ptratio 0.176543614  
## black -0.002754823  
## medv 0.047683009

boston\_lda\_class <- predict(boston\_lda, boston\_test)$class  
table(boston\_lda\_class, med\_crime\_test)

## med\_crime\_test  
## boston\_lda\_class 0 1  
## 0 10 0  
## 1 11 135

mean(boston\_lda\_class == med\_crime\_test)

## [1] 0.9294872

mean(boston\_lda\_class != med\_crime\_test)

## [1] 0.07051282

A QDA model with the same predictors has an accuracy of 13.5% with an error of 86.5%.

boston\_qda <- qda(med\_crime ~ zn + nox + dis + rad + tax + ptratio + black + medv, data = boston\_train)  
boston\_qda

## Call:  
## qda(med\_crime ~ zn + nox + dis + rad + tax + ptratio + black +   
## medv, data = boston\_train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.6628571 0.3371429   
##   
## Group means:  
## zn nox dis rad tax ptratio black medv  
## 0 21.706897 0.4648004 5.133497 4.202586 295.1121 17.72284 388.9606 25.48491  
## 1 2.576271 0.6017458 2.982705 5.033898 340.0847 17.66102 363.6945 24.25085

boston\_qda\_class <- predict(boston\_qda, boston\_test)$class  
table(boston\_qda\_class, med\_crime\_test)

## med\_crime\_test  
## boston\_qda\_class 0 1  
## 0 15 129  
## 1 6 6

mean(boston\_qda\_class == med\_crime\_test)

## [1] 0.1346154

mean(boston\_qda\_class != med\_crime\_test)

## [1] 0.8653846

A naive Bayes model with the same predictors has an accuracy of 44.2% with an error of 55.8%.

boston\_nb <- naiveBayes(med\_crime ~ zn + nox + dis + rad + tax + ptratio + black + medv, data = boston\_train)  
boston\_nb

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.6628571 0.3371429   
##   
## Conditional probabilities:  
## zn  
## Y [,1] [,2]  
## 0 21.706897 29.042816  
## 1 2.576271 6.784103  
##   
## nox  
## Y [,1] [,2]  
## 0 0.4648004 0.04827832  
## 1 0.6017458 0.11867266  
##   
## dis  
## Y [,1] [,2]  
## 0 5.133497 1.839583  
## 1 2.982705 1.286911  
##   
## rad  
## Y [,1] [,2]  
## 0 4.202586 1.632302  
## 1 5.033898 1.407745  
##   
## tax  
## Y [,1] [,2]  
## 0 295.1121 65.82329  
## 1 340.0847 61.23370  
##   
## ptratio  
## Y [,1] [,2]  
## 0 17.72284 1.741683  
## 1 17.66102 2.903433  
##   
## black  
## Y [,1] [,2]  
## 0 388.9606 23.09907  
## 1 363.6945 58.42536  
##   
## medv  
## Y [,1] [,2]  
## 0 25.48491 7.13785  
## 1 24.25085 10.60898

boston\_nb\_class <- predict(boston\_nb, boston\_test)  
table(boston\_nb\_class, med\_crime\_test)

## med\_crime\_test  
## boston\_nb\_class 0 1  
## 0 6 72  
## 1 15 63

mean(boston\_nb\_class == med\_crime\_test)

## [1] 0.4423077

mean(boston\_nb\_class != med\_crime\_test)

## [1] 0.5576923

A KNN with K = 20 and the same predictors has an accuracy of 94.9% with an error of 5.1%.

predictors <- c("zn", "nox", "dis", "rad", "tax", "ptratio", "black", "medv")  
X <- Boston[, predictors]  
y <- Boston$med\_crime  
train\_indices <- 1:350  
X\_train <- X[train\_indices, ]  
y\_train <- y[train\_indices]  
test\_indices <- (351:nrow(Boston))  
X\_test <- X[test\_indices, ]  
y\_test <- y[test\_indices]  
  
set.seed(1)  
boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 1)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 20 95  
## 1 1 40

mean(boston\_knn == med\_crime\_test)

## [1] 0.3846154

mean(boston\_knn != med\_crime\_test)

## [1] 0.6153846

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 5)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 17 20  
## 1 4 115

mean(boston\_knn == med\_crime\_test)

## [1] 0.8461538

mean(boston\_knn != med\_crime\_test)

## [1] 0.1538462

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 10)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 16 4  
## 1 5 131

mean(boston\_knn == med\_crime\_test)

## [1] 0.9423077

mean(boston\_knn != med\_crime\_test)

## [1] 0.05769231

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 20)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 16 3  
## 1 5 132

mean(boston\_knn == med\_crime\_test)

## [1] 0.9487179

mean(boston\_knn != med\_crime\_test)

## [1] 0.05128205

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 30)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 18 35  
## 1 3 100

mean(boston\_knn == med\_crime\_test)

## [1] 0.7564103

mean(boston\_knn != med\_crime\_test)

## [1] 0.2435897

A logistic regression model with only the four most significant predictors (nox, dis, rad, and ptratio) has an accuracy of 93.6% and an error of 6.4%.

boston\_log3 <- glm(med\_crime ~ nox + dis + rad + ptratio, data = boston\_train, family = binomial)  
boston\_probs\_log3 <- predict(boston\_log3, boston\_test, type = "response")  
boston\_pred\_log3 <- rep(0, nrow(boston\_test))  
boston\_pred\_log3[boston\_probs\_log3 > 0.5] = 1  
table(boston\_pred\_log3, med\_crime\_test)

## med\_crime\_test  
## boston\_pred\_log3 0 1  
## 0 11 0  
## 1 10 135

mean(boston\_pred\_log3 == med\_crime\_test)

## [1] 0.9358974

mean(boston\_pred\_log3 != med\_crime\_test)

## [1] 0.06410256

An LDA model with only the four most significant predictors (nox, dis, rad, and ptratio) has an accuracy of 93.6% and an error of 6.4%.

boston\_lda2 <- lda(med\_crime ~ nox + dis + rad + ptratio, data = boston\_train)  
boston\_lda2

## Call:  
## lda(med\_crime ~ nox + dis + rad + ptratio, data = boston\_train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.6628571 0.3371429   
##   
## Group means:  
## nox dis rad ptratio  
## 0 0.4648004 5.133497 4.202586 17.72284  
## 1 0.6017458 2.982705 5.033898 17.66102  
##   
## Coefficients of linear discriminants:  
## LD1  
## nox 10.78304239  
## dis -0.11472070  
## rad 0.20323013  
## ptratio 0.08604641

boston\_lda\_class2 <- predict(boston\_lda2, boston\_test)$class  
table(boston\_lda\_class2, med\_crime\_test)

## med\_crime\_test  
## boston\_lda\_class2 0 1  
## 0 11 0  
## 1 10 135

mean(boston\_lda\_class2 == med\_crime\_test)

## [1] 0.9358974

mean(boston\_lda\_class2 != med\_crime\_test)

## [1] 0.06410256

A QDA model with only the four most significant predictors (nox, dis, rad, and ptratio) has an accuracy of 5.8% and an error of 94.2%.

boston\_qda2 <- qda(med\_crime ~ nox + dis + rad + ptratio, data = boston\_train)  
boston\_qda2

## Call:  
## qda(med\_crime ~ nox + dis + rad + ptratio, data = boston\_train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.6628571 0.3371429   
##   
## Group means:  
## nox dis rad ptratio  
## 0 0.4648004 5.133497 4.202586 17.72284  
## 1 0.6017458 2.982705 5.033898 17.66102

boston\_qda\_class2 <- predict(boston\_qda2, boston\_test)$class  
table(boston\_qda\_class2, med\_crime\_test)

## med\_crime\_test  
## boston\_qda\_class2 0 1  
## 0 6 132  
## 1 15 3

mean(boston\_qda\_class2 == med\_crime\_test)

## [1] 0.05769231

mean(boston\_qda\_class2 != med\_crime\_test)

## [1] 0.9423077

A naive Bayes model with only the four most significant predictors (nox, dis, rad, and ptratio) has an accuracy of 10.9% and an error of 89.1%.

boston\_nb <- naiveBayes(med\_crime ~ nox + dis + rad + ptratio, data = boston\_train)  
boston\_nb

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.6628571 0.3371429   
##   
## Conditional probabilities:  
## nox  
## Y [,1] [,2]  
## 0 0.4648004 0.04827832  
## 1 0.6017458 0.11867266  
##   
## dis  
## Y [,1] [,2]  
## 0 5.133497 1.839583  
## 1 2.982705 1.286911  
##   
## rad  
## Y [,1] [,2]  
## 0 4.202586 1.632302  
## 1 5.033898 1.407745  
##   
## ptratio  
## Y [,1] [,2]  
## 0 17.72284 1.741683  
## 1 17.66102 2.903433

boston\_nb\_class <- predict(boston\_nb, boston\_test)  
table(boston\_nb\_class, med\_crime\_test)

## med\_crime\_test  
## boston\_nb\_class 0 1  
## 0 6 124  
## 1 15 11

mean(boston\_nb\_class == med\_crime\_test)

## [1] 0.1089744

mean(boston\_nb\_class != med\_crime\_test)

## [1] 0.8910256

A KNN with K = 5 or K = 10 with only the four most significant predictors (nox, dis, rad, and ptratio) has an accuracy of 94.9% and an error of 5.1%.

predictors <- c("nox", "dis", "rad", "ptratio")  
X <- Boston[, predictors]  
y <- Boston$med\_crime  
train\_indices <- 1:350  
X\_train <- X[train\_indices, ]  
y\_train <- y[train\_indices]  
test\_indices <- (351:nrow(Boston))  
X\_test <- X[test\_indices, ]  
y\_test <- y[test\_indices]  
  
set.seed(1)  
boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 1)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 16 4  
## 1 5 131

mean(boston\_knn == med\_crime\_test)

## [1] 0.9423077

mean(boston\_knn != med\_crime\_test)

## [1] 0.05769231

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 5)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 16 3  
## 1 5 132

mean(boston\_knn == med\_crime\_test)

## [1] 0.9487179

mean(boston\_knn != med\_crime\_test)

## [1] 0.05128205

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 10)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 16 3  
## 1 5 132

mean(boston\_knn == med\_crime\_test)

## [1] 0.9487179

mean(boston\_knn != med\_crime\_test)

## [1] 0.05128205

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 20)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 11 3  
## 1 10 132

mean(boston\_knn == med\_crime\_test)

## [1] 0.9166667

mean(boston\_knn != med\_crime\_test)

## [1] 0.08333333

boston\_knn <- knn(train = X\_train, test = X\_test, cl = y\_train, k = 30)  
table(boston\_knn, y\_test)

## y\_test  
## boston\_knn 0 1  
## 0 11 3  
## 1 10 132

mean(boston\_knn == med\_crime\_test)

## [1] 0.9166667

mean(boston\_knn != med\_crime\_test)

## [1] 0.08333333

In summary, the most accurate classification model was created with the KNN method, with an accuracy of 94.9%. This accuracy was achieved with all significant predictors and K = 20 with just the four most significant predictors and K = 5 or K = 10.

methods\_table\_all\_sig\_predictors <- data.frame(  
 Method = c("Logistic", "LDA", "QDA", "Naive Bayes", "KNN (K = 20)"),  
 Accuracy = c("93.6%", "92.9%", "13.5%", "44.2%", "94.9%"),  
 Error = c("6.4%", "7.1%", "86.5%", "55.8%", "5.1%")  
)  
title1 <- "All Significant Predictors"  
cat(title1, "\n")

## All Significant Predictors

print(methods\_table\_all\_sig\_predictors)

## Method Accuracy Error  
## 1 Logistic 93.6% 6.4%  
## 2 LDA 92.9% 7.1%  
## 3 QDA 13.5% 86.5%  
## 4 Naive Bayes 44.2% 55.8%  
## 5 KNN (K = 20) 94.9% 5.1%

methods\_table\_four\_predictors <- data.frame(  
 Method = c("Logistic", "LDA", "QDA", "Naive Bayes", "KNN (K = 5 or 10)"),  
 Accuracy = c("93.6%", "93.6%", "5.8%", "10.9%", "94.9%"),  
 Error = c("6.4%", "6.4%", "94.1%", "89.1%", "5.1%")  
)  
title2 <- "Four Most Significant Predictors"  
cat(title2, "\n")

## Four Most Significant Predictors

print(methods\_table\_four\_predictors)

## Method Accuracy Error  
## 1 Logistic 93.6% 6.4%  
## 2 LDA 93.6% 6.4%  
## 3 QDA 5.8% 94.1%  
## 4 Naive Bayes 10.9% 89.1%  
## 5 KNN (K = 5 or 10) 94.9% 5.1%