ISL Chapter 4 Excercises

Jonathan Bryan

April 20, 2018

1. Using a little bit of algebra, prove that (4.2) is equivalent to (4.3). In other words, the logistic function representation and logit representation for the logistic regression model are equivalent.

$$
\begin{aligned}
p(x) &= \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \text{ (Logistic function)} \\
p(x) + p(x)e^{\beta_0+\beta_1 X} &= e^{\beta_0+\beta_1 X} \\
p(x) &= e^{\beta_0+\beta_1 X} p(x) e^{\beta_0+\beta_1 X} \\
p(x) &= e^{\beta_0+\beta_1 X}(1+p(x)) \\
\frac{p(x)}{(1+p(x))} &= e^{\beta_0+\beta_1 X} \\
log(\frac{p(x)}{(1+p(x))}) &= \beta_0 + \beta_1 X \text{ (logit transformation)}
\end{aligned}
$$

2. It was stated in the text that classifying an observation to the class for which (4.12) is largest is equivalent to classifying an observation to the class for which (4.13) is largest. Prove that this is the case. In other words, under the assumption that the observations in the kth class are drawn from a $N(\mu_k, \sigma^2)$ distribution, the Bayes' classifier assigns an observation to the class for which the discriminant function is maximized.

Given $X \sim N(\mu_k, \sigma^2)$ and that $log()$ is a monotonically increase function such that,

$$\forall x, y,\ x \leq y,\ log(x) < log(y)$$

We can conclude that, $arg_X\ max\ p_k(X) = arg_X\ max\ log(p_k(X))$

3. This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class specific mean vector and a class specific covariance matrix. We consider the simple case where p = 1; i.e. there is only one feature. Suppose that we have K classes, and that if an observation belongs to the kth class then X comes from a one-dimensional normal distribution, $X \sim N(\mu_k, \sigma_k^2)$. Recall that the density function for the one-dimensional normal distribution is given in (4.11). Prove that in this case, the Bayes' classifier is not linear. Argue that it is in fact quadratic.

$$
\begin{aligned}
p_k(X) &= \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2} \pi_k \\
log(p_k(X)) = \delta_k(x) &= -\frac{log(2\pi)}{2} - log(\sigma_k) - \frac{1}{2\sigma_k^2}(x-\mu_k)^2 + log(\pi_k) \\
\delta_k(x) &\propto -\frac{1}{2\sigma_k^2}(x^2 - 2x\mu_k + \mu_k^2) - log(\sigma_k) + log(\pi_k) \\
\delta_k(x) &\propto -\frac{x^2}{2\sigma_k^2} + \frac{x\mu_k}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} - log(\sigma_k) + log(\pi_k)
\end{aligned}
$$

As shown above, we find the value of k that maximizes the observed x. We see that the x term is quadratic in the final equation above and thus our classifier is quadratic.

4. When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a) Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with X = 0.6, we will use observations in the range [0.55, 0.65]. On average, what fraction of the available observations will we use to make the prediction?

We know that when x is in between 0.05 and 0.95 we will have 10 percent of the observations available because there is at least $[x - 0.5, x + 0.5]$ available on both sides of the point. When x is below 0.05 there is only $x + 0.05$ observations available and it varies with x, similarly, when x is above 0.95 there is only $1.05 - x$ observations available. We can show:

For $X \in (0.05, 0.95)$: $\int_{0.05}^{0.95} 0.10 \; dx = 0.10x|_{0.05}^{0.95} = .09 \backslash$ For $X \in (0, 0.05)$: $\int_{0}^{0.05} x + 0.05 \; dx = x^2/2 + 0.05x|_{0}^{0.05} = .00375 \backslash$ For $X \in (0.95, 1)$: $\int_{0.95}^{1} 1.05 - x \; dx = 1.05x - x^2/2|_{0.95}^{1} = .00375 \backslash$

Therefore, on average there are 9.75 percent of observations available.

(b) Now suppose that we have a set of observations, each with measurements on p = 2 features, X_1 and X_2. We assume that (X_1,X_2) are uniformly distributed on [0, 1] × [0, 1]. We wish to predict a test observation's response using only observations that are within 10% of the range of X_1 and within 10% of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with X_1 = 0.6 and X_2 = 0.35, we will use observations in the range [0.55, 0.65] for X_1 and in the range [0.3, 0.4] for X_2. On average, what fraction of the available observations will we use to make the prediction?

Given X_1 and X_2 are independent, we know that the probability space of the joint distribution is much smaller for each point compared to the one dimensional space. For instance, if we evaluated the average number of available observations for $X_1, X_2 \in (0.05, 0.95)$ we would have $0.1 * 0.1 = 0.01$ observations available on average. Since we know the true average over the full domain of the marginal distributions we can compute \$0.0975 * 0.0975 = \$.950625 percent.

(c) Now suppose that we have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

Given all variables are independent, we can compute $.0975^{100}$ percent of observations are available. A very small amount.

(d) Using your answers to parts (a)-(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

As p increases, the domain of the predictors becomes much more sparse between points and the probability that training observation is near the test observation becomes lower. This means KNN loses prediction accuracy because training observations become much more unique as the number of predictors increases.

(e) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

In general, the hypercube must increase in size as the number of predictors increases because of the sparsity described in part (d). We can show that for p = 1: $l = .10$, such that when $X \in (0, 0.05)$ or $(.95, 1)$ we always extend the length to equal 10 and not just within ± 0.05 of the training observation. Extending this out we get for p = 2 and p = 100: $l_2 = \sqrt{.10}$, $l_1 00 = .10^{1/100}$

5. We now examine the differences between LDA and QDA.

First we give the models for LDA and QDA, where each assumes the data is normally distributed and conditioned on class and LDA further assumed shared covariance.

LDA classifier: $\delta_k(x) = x^T \hat{\Sigma^{-1}} \hat{\mu_k} - \frac{1}{2} \hat{\mu_k^T} \hat{\Sigma^{-1}} \hat{\mu_k} + log(\pi_k)$ QDA classifier: $\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \hat{\Sigma_k^{-1}}(x - \mu_k) - \frac{1}{2} log(|\hat{\Sigma_k}|) + log(\pi_k)$

Where for both models in the binary case: $f(Y|x) = \begin{cases} 0, & \text{if } \delta_0 > \delta_1 \\ 1, & \text{if } \delta_1 > \delta_0 \end{cases}$

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

Because QDA is more flexible than LDA we would expect it to perform better on the training set but worse on the test set due to overfitting on average.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

We would expect QDA to perform better on both the training and test set on average.

(c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

As the training sample size increases, the variation of the more flexible QDA model lowers and therefore it improves relative to LDA.

(d) (d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

False. Unless the number of training samples is substantial, QDA has a higher model variance and will overfit to the training data leading to inferior test error rates compared to LDA.

6. Suppose we collect data for a group of students in a statistics class with variables X_1 =hours studied, X_2 =undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0$ = ???6, $\hat{\beta}_1$ = 0.05, $\hat{\beta}_2$ = 1.

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

$$\begin{aligned}
log(\frac{\pi}{1-\pi}) &= -6 + 0.05(40) + 1(3.5) \\
\frac{\pi}{1-\pi} &= exp\{-0.5\} \\
\pi &= exp\{-0.5\}(1-\pi) \\
\pi &= exp\{-0.5\} - exp\{-0.5\}\pi \\
\pi + exp\{-0.5\}\pi &= exp\{-0.5\} \\
\pi(1 + exp\{-0.5\}) &= exp\{-0.5\} \\
\pi &= \frac{exp\{-0.5\}}{(1 + exp\{-0.5\})} \\
\pi &\approx .38
\end{aligned}$$

(b) How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

$$\begin{aligned}
log(\frac{.5}{.5}) &= -6 + 0.05(x_1) + 1(3.5) \\
log(1) &= -6 + 0.05(x_1) + 1(3.5) \\
log(1) + 6 - 1(3.5) &= 0.05(x_1) \\
x_1 &= \frac{log(1) + 6 - 1(3.5)}{0.05} \\
x_1 &= 50
\end{aligned}$$

7. Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit.We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\bar{X} = 10$, while the mean for those that didn't was $\bar{X} = 0$. In addition, the variance of X for these two sets of companies was $\hat{\sigma}^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was X = 4 last year.

$$
\begin{aligned}
Pr(Y|X) &= \frac{Pr(X|Y)Pr(Y)}{\Sigma_{y\in Y} Pr(X|Y)Pr(Y)} \\
Pr(Y = Yes|X = 4) &= \frac{1}{\sqrt{72\pi}} exp\{-\frac{1}{72}(4-10)^2\} \\
Pr(Y = Yes|X = 4) &= .040 \\
Pr(Y = No|X = 4) &= \frac{1}{\sqrt{72\pi}} exp\{-\frac{1}{72}(4-0)^2\} \\
Pr(Y = No|X = 4) &= .053 \\
Pr(Y) &= 0.8 \\
\Sigma_{y\in Y} Pr(X|Y)Pr(Y) &= (.040) * (.80) + (.053) * (.20) = .0426 \\
Pr(Y|X) &= \frac{(.040) * (.80)}{(.0426)} = .751
\end{aligned}
$$

8. Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

If K = 1 for a KNN model, the training error will be 0% because the nearest neighbor will always be the training point being evaluated. This means the test error rate of the KNN model is 36%, which is higher than the logistic regression training error rate of 30%. The logistic regression model is preferred.

9. This problem has to do with odds.

(a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

$$\frac{\pi}{1-\pi} = 0.37$$

\

$$\pi = 0.37(1-\pi)$$

\

$$\pi = 0.37 - 0.37\pi$$

\

$$1.37\pi = 0.37$$

\

$$\pi = \frac{0.37}{1.37} = 27\%$$

(b) Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will default?

$$\frac{.16}{1-.16} = 0.19$$

\

10. This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

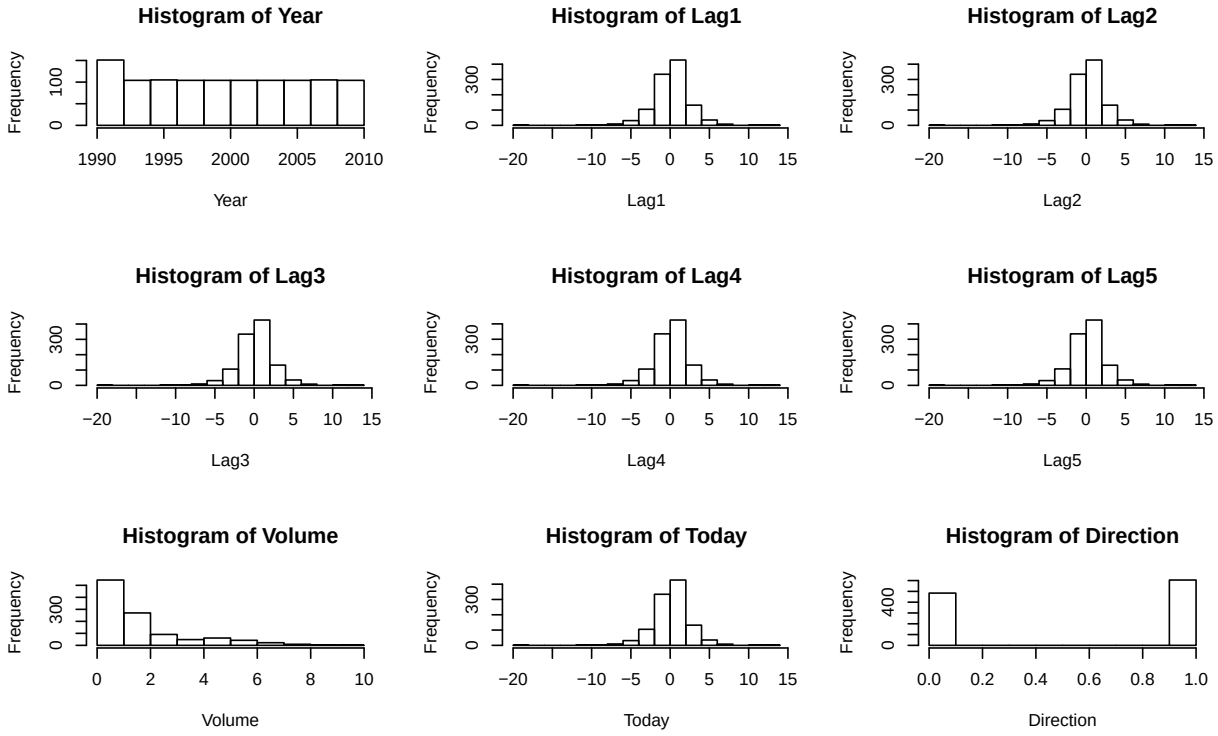
(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
library(ISLR)
library(corrplot)
```

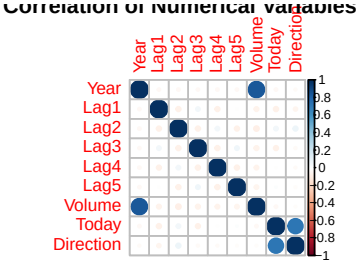
```
## Warning: package 'corrplot' was built under R version 3.4.3
```

```
## corrplot 0.84 loaded
```

```
#Change direction to numerical
Weekly$Direction = as.numeric(Weekly$Direction) - 1

#Histograms
par(mfrow=c(3,3))
for (i in 1:(dim(Weekly)[2])){
  hist(Weekly[,i], main = paste0("Histogram of ", colnames(Weekly)[i]),
                   xlab = colnames(Weekly)[i])
}
```




```
#Correlation
M = cor(Weekly)
corrplot(M, main = "Correlation of Numerical Variables")
```



We observe that returns look approximately normally distributed around zero. There are more positive gain days than negative. Volume has a positive weak correlation with year and market direction has a positive weak correlation with today's returns.

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```
library(stargazer)

#Logistic regression model
log_reg = glm(Direction ~ . -Year - Today, family = "binomial", data = Weekly)
stargazer(log_reg)
```

```
##
## % Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvar
## % Date and time: Wed, Jun 27, 2018 - 1:15:59 PM
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{1}{c}{\textit{Dependent variable:}} \\
## \cline{2-2}
## \\[-1.8ex] & Direction \\
## \hline \\[-1.8ex]
##  Lag1 & $-$0.041 \\
##   & (0.026) \\
##   & \\
##  Lag2 & 0.058$^{**}$ \\
##   & (0.027) \\
##   & \\
##  Lag3 & $-$0.016 \\
##   & (0.027) \\
##   & \\
##  Lag4 & $-$0.028 \\
##   & (0.026) \\
##   & \\
##  Lag5 & $-$0.014 \\
##   & (0.026) \\
##   & \\
##  Volume & $-$0.023 \\
##   & (0.037) \\
##   & \\
##  Constant & 0.267$^{***}$ \\
##   & (0.086) \\
##   & \\
## \hline \\[-1.8ex]
## Observations & 1,089 \\
## Log Likelihood & $-$743.179 \\
## Akaike Inf. Crit. & 1,500.357 \\
## \hline
## \hline \\[-1.8ex]
```

```
## \textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

Percentage return for 2 weeks previous is statistically significance at the $\alpha = 0.05$ level.

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
log_reg_pred = predict(log_reg, type="response")
log_reg_pred[log_reg_pred >= 0.5] = 1
log_reg_pred[log_reg_pred < 0.5] = 0
confusionMatrix(as.factor(log_reg_pred), as.factor(Weekly$Direction), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  54  48
##          1 430 557
##
##                Accuracy : 0.5611
##                  95% CI : (0.531, 0.5908)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 0.369
##
##                   Kappa : 0.035
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9207
##             Specificity : 0.1116
##          Pos Pred Value : 0.5643
##          Neg Pred Value : 0.5294
##              Prevalence : 0.5556
##          Detection Rate : 0.5115
##    Detection Prevalence : 0.9063
##       Balanced Accuracy : 0.5161
##
##        'Positive' Class : 1
##
```

The overall accuracy of the model is 56 percent. This is only marginally better than always predicting the market will go up, as the prevalence of the positive market movement is 55 percent. There logistic regression model has high sensitivity (correct 92 percent of the time) to positive market movements but very low specificity when predicting the market will go down (11 percent). A 95 percent confidence interval of the overall model accuracy is between 53 and 59 percent meaning the model could be either slightly better or worse than always predicting positive market movements.

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
#Logistic regression model with Lag2 as only predictor
log_reg2 = glm(Direction ~ Lag2, family = "binomial", data = Weekly, subset = (1990 <= Weekly$Year & We

#Prediction
log_reg2_pred = predict(log_reg2, newdata = Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,],type = "
log_reg2_pred[log_reg2_pred >= 0.5] = 1
log_reg2_pred[log_reg2_pred < 0.5] = 0

#Confusion matrix
confusionMatrix(as.factor(log_reg2_pred), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  9  5
##          1 34 56
##
##                Accuracy : 0.625
##                  95% CI : (0.5247, 0.718)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.2439
##
##                   Kappa : 0.1414
##  Mcnemar's Test P-Value : 7.34e-06
##
##             Sensitivity : 0.9180
##             Specificity : 0.2093
##          Pos Pred Value : 0.6222
##          Neg Pred Value : 0.6429
##              Prevalence : 0.5865
##          Detection Rate : 0.5385
##    Detection Prevalence : 0.8654
##       Balanced Accuracy : 0.5637
##
##        'Positive' Class : 1
##
```

(e) Repeat (d) using LDA.

```
library(MASS)

#Linear discriminant analysis
lda = lda(Direction ~ Lag2, data = Weekly, subset = (1990 <= Weekly$Year & Weekly$Year <= 2008))

#Prediction
lda_pred = predict(lda, newdata = Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,],type = "response")

#Confusion matrix
confusionMatrix(as.factor(lda_pred), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$Direc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  9  5
##          1 34 56
##
##                Accuracy : 0.625
##                  95% CI : (0.5247, 0.718)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.2439
##
##                   Kappa : 0.1414
##  Mcnemar's Test P-Value : 7.34e-06
##
##             Sensitivity : 0.9180
##             Specificity : 0.2093
##          Pos Pred Value : 0.6222
##          Neg Pred Value : 0.6429
##              Prevalence : 0.5865
##          Detection Rate : 0.5385
##    Detection Prevalence : 0.8654
##       Balanced Accuracy : 0.5637
##
##        'Positive' Class : 1
##
```

(f) Repeat (d) using QDA.

```
#Quadratic discriminant analysis
qda = qda(Direction ~ Lag2, data = Weekly, subset = (1990 <= Weekly$Year & Weekly$Year <= 2008))

#Prediction
qda_pred = predict(qda, newdata = Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,],type = "response")$

#Confusion matrix
confusionMatrix(as.factor(qda_pred), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$Direc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  0  0
##          1 43 61
##
##                Accuracy : 0.5865
##                  95% CI : (0.4858, 0.6823)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.5419
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 1.504e-10
##
##             Sensitivity : 1.0000
```

```
##             Specificity : 0.0000
##          Pos Pred Value : 0.5865
##          Neg Pred Value :    NaN
##              Prevalence : 0.5865
##          Detection Rate : 0.5865
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 1
##
```

(g) Repeat (d) using KNN with K = 1.

```
library(class)

#KNN K = 1
knn = knn(train = as.matrix(Weekly[1990 <= Weekly$Year & Weekly$Year <= 2008,]$Lag2),
          test = as.matrix(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$Lag2),
          cl = as.factor(Weekly[1990 <= Weekly$Year & Weekly$Year <= 2008,]$Direction))

#Confusion matrix
confusionMatrix(as.factor(knn), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$Direction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 21 30
##          1 22 31
##
##                Accuracy : 0.5
##                  95% CI : (0.4003, 0.5997)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.9700
##
##                   Kappa : -0.0033
##  Mcnemar's Test P-Value : 0.3317
##
##             Sensitivity : 0.5082
##             Specificity : 0.4884
##          Pos Pred Value : 0.5849
##          Neg Pred Value : 0.4118
##              Prevalence : 0.5865
##          Detection Rate : 0.2981
##    Detection Prevalence : 0.5096
##       Balanced Accuracy : 0.4983
##
##        'Positive' Class : 1
##
```

(h) Which of these methods appears to provide the best results on this data?

Both the logistic regression and and linear discriminant analysis models with the Lag2 predictor have nearly identical predictive power, with 62.5 percent overall accuracy. The KNN model has better specificity compared

to the logistic regression and LDA models ($.48 > .21$) but it has an overall accuracy lower than the the no information rate ($0.5 < 0.58$). The logistic regression model would be preferred given it has equal performance to the LDA model and less model assumptions.

(i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

Logistic Regression

```
#Logistic regression model
log_reg3 = glm(Direction ~ Lag1 + Lag2 + Year, family = "binomial", data = Weekly, subset = (1990 <= We

#Prediction
log_reg3_pred = predict(log_reg3, newdata = Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,],type = "
log_reg3_pred[log_reg3_pred >= 0.5] = 1
log_reg3_pred[log_reg3_pred < 0.5] = 0

#Confusion matrix
confusionMatrix(as.factor(log_reg3_pred), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20 19
##          1 23 42
##
##                Accuracy : 0.5962
##                  95% CI : (0.4954, 0.6913)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.4626
##
##                   Kappa : 0.1558
##  Mcnemar's Test P-Value : 0.6434
##
##             Sensitivity : 0.6885
##             Specificity : 0.4651
##          Pos Pred Value : 0.6462
##          Neg Pred Value : 0.5128
##              Prevalence : 0.5865
##          Detection Rate : 0.4038
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.5768
##
##        'Positive' Class : 1
##
```

LDA

```
#Linear discriminant analysis
lda2 = lda(Direction ~ Lag1 + Lag2 + Year, data = Weekly, subset = (1990 <= Weekly$Year & Weekly$Year <

#Prediction
lda_pred2 = predict(lda2, newdata = Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,],type = "response
```

```r
#Confusion matrix
confusionMatrix(as.factor(lda_pred2), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$Direc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20 19
##          1 23 42
##
##                Accuracy : 0.5962
##                  95% CI : (0.4954, 0.6913)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.4626
##
##                   Kappa : 0.1558
##  Mcnemar's Test P-Value : 0.6434
##
##             Sensitivity : 0.6885
##             Specificity : 0.4651
##          Pos Pred Value : 0.6462
##          Neg Pred Value : 0.5128
##              Prevalence : 0.5865
##          Detection Rate : 0.4038
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.5768
##
##        'Positive' Class : 1
##
```

QDA

```r
#Linear discriminant analysis
qda2 = qda(Direction ~ Lag1 + Lag2 + Year, data = Weekly, subset = (1990 <= Weekly$Year & Weekly$Year <

#Prediction
qda_pred2 = predict(qda2, newdata = Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,],type = "response

#Confusion matrix
confusionMatrix(as.factor(qda_pred2), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$Direc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 17 27
##          1 26 34
##
##                Accuracy : 0.4904
##                  95% CI : (0.391, 0.5903)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.9811
##
##                   Kappa : -0.0471
```

```
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.5574
##             Specificity : 0.3953
##          Pos Pred Value : 0.5667
##          Neg Pred Value : 0.3864
##              Prevalence : 0.5865
##          Detection Rate : 0.3269
##    Detection Prevalence : 0.5769
##       Balanced Accuracy : 0.4764
##
##        'Positive' Class : 1
##
```

KNN, K = 5

```
#KNN K = 5
knn = knn(train = as.matrix(Weekly[1990 <= Weekly$Year & Weekly$Year <= 2008,c("Lag1", "Lag2", "Year")]),
          test = as.matrix(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,c("Lag1", "Lag2", "Year")]),
          cl = as.factor(Weekly[1990 <= Weekly$Year & Weekly$Year <= 2008,]$Direction), k=5)

#Confusion matrix
confusionMatrix(as.factor(knn), as.factor(Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010,]$Direction))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 32 46
##          1 11 15
##
##                Accuracy : 0.4519
##                  95% CI : (0.3541, 0.5526)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.9979
##
##                   Kappa : -0.0088
##  Mcnemar's Test P-Value : 6.687e-06
##
##             Sensitivity : 0.2459
##             Specificity : 0.7442
##          Pos Pred Value : 0.5769
##          Neg Pred Value : 0.4103
##              Prevalence : 0.5865
##          Detection Rate : 0.1442
##    Detection Prevalence : 0.2500
##       Balanced Accuracy : 0.4950
##
##        'Positive' Class : 1
##
```

Lag1 was added to each model because it was close to being significant in the full logistic regression model. Year was also added as different years may have had unique effects on market movements, such as recessions and booms. Both the logistic regression and LDA models had nearly identical performance and were superior to the QDA and KNN (K = 5) models. However, accuracy with the added variables was lower for each of the models compared to just using Lag2.

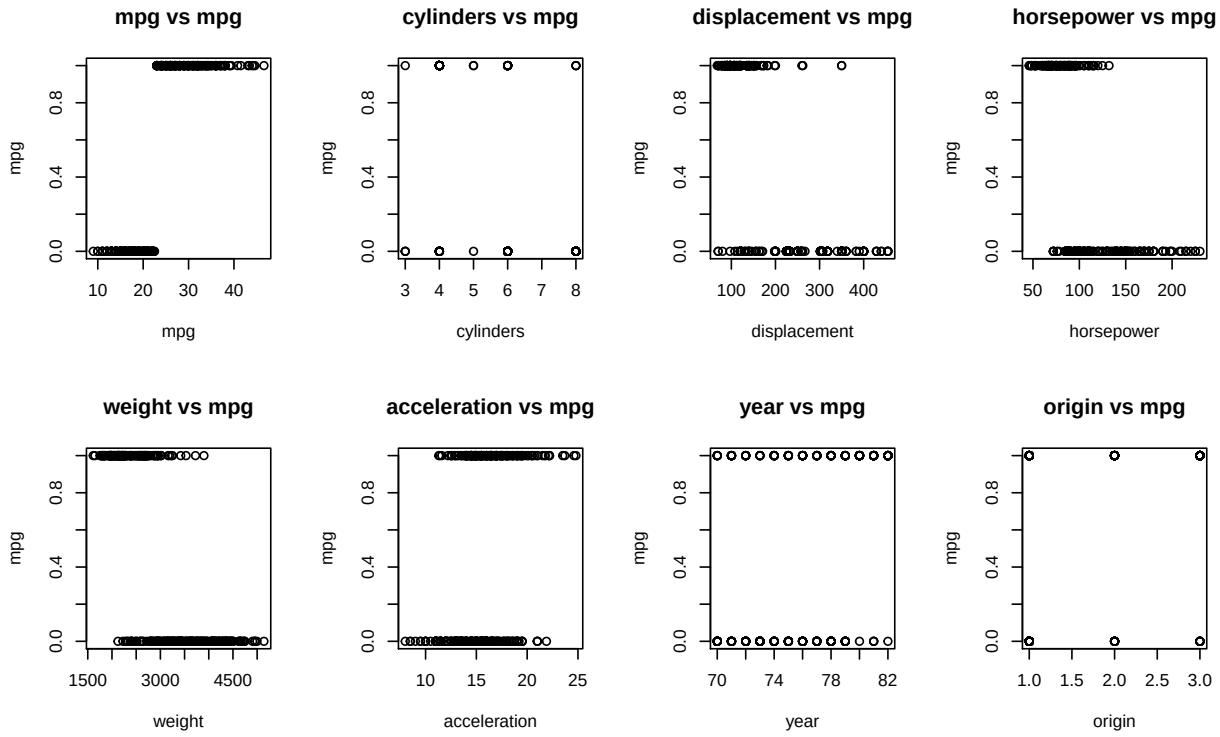
10. In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

(a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

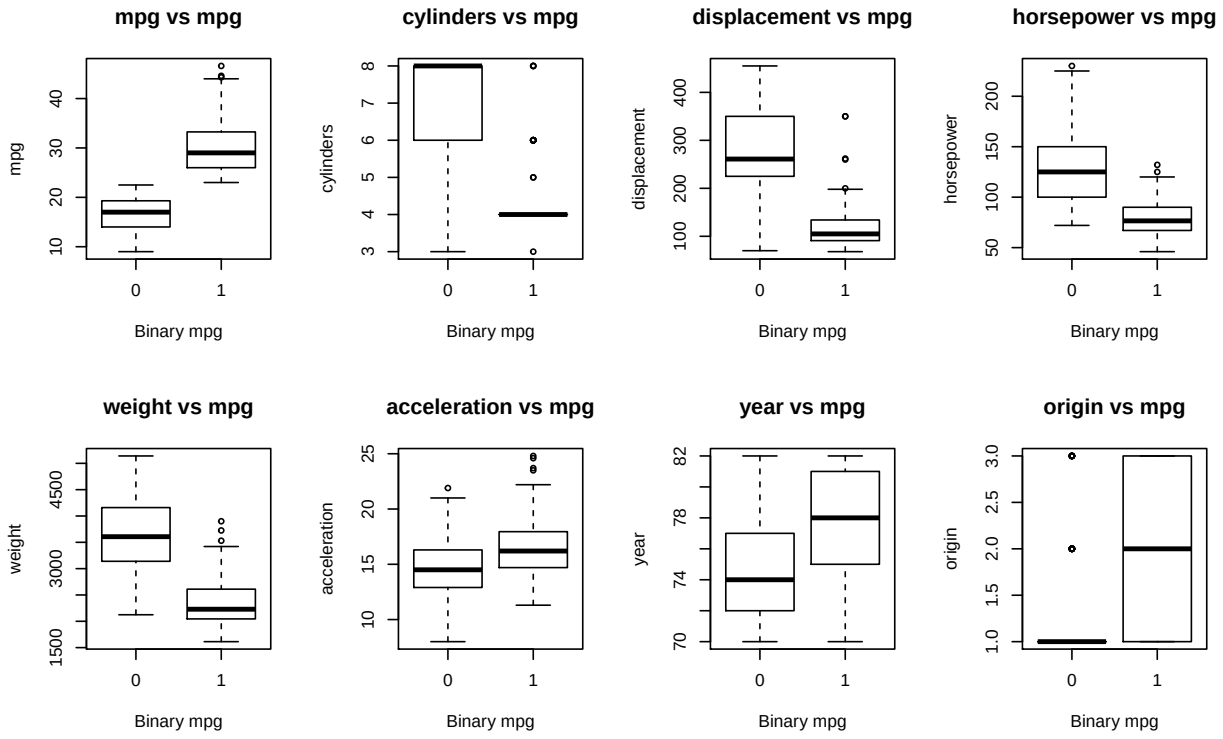
```
library(ISLR)

#Create binary mpd variable
mpg01_med= median(Auto$mpg)
Auto$mpg01 = NA
Auto[Auto$mpg >= mpg01_med,]$mpg01 = 1
Auto[Auto$mpg < mpg01_med,]$mpg01 = 0
```

(b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
par(mfrow=c(2,4))
for (i in 1:(ncol(Auto) - 2)){
  plot(Auto[,i], Auto$mpg01, main = paste0(colnames(Auto)[i], " vs mpg"),
                             xlab = colnames(Auto)[i],
                             ylab = "mpg")
}
```



```
par(mfrow=c(2,4))
for (i in 1:(ncol(Auto) - 2)){
  boxplot(Auto[,i] ~ Auto$mpg01,
          main = paste0(colnames(Auto)[i], " vs mpg"),
          xlab = "Binary mpg",
          ylab = colnames(Auto)[i])
}
```



(c) Split the data into a training set and a test set.

```
n = nrow(Auto)
train = seq(1,0.75*n,1)
Auto_train = Auto[train,]
Auto_test = Auto[-train,]
```

(d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
lda_auto = lda(mpg01 ~ cylinders + displacement + horsepower + weight + year + origin, data = Auto_trai
lda_auto_pred = predict(lda_auto, newdata = Auto_test)$class
confusionMatrix(lda_auto_pred, as.factor(Auto_test$mpg01), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 14
##          1  0 79
```

```
##
##                Accuracy : 0.8571
##                  95% CI : (0.7719, 0.9196)
##     No Information Rate : 0.949
##     P-Value [Acc > NIR] : 0.999866
##
##                   Kappa : 0.3654
##  Mcnemar's Test P-Value : 0.000512
##
##             Sensitivity : 0.8495
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.2632
##              Prevalence : 0.9490
##          Detection Rate : 0.8061
##    Detection Prevalence : 0.8061
##       Balanced Accuracy : 0.9247
##
##        'Positive' Class : 1
##
```

The LDA model uses cylinders, displacement, horsepower, weight, year, and origin as predictors. The overall test error is 14%. The sensitivity and specificity of the model is 0.84 and 1.0 respectively.

(e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
qda_auto = qda(mpg01 ~ cylinders + displacement + horsepower + weight + year + origin, data = Auto_trai
qda_auto_pred = predict(qda_auto, newdata = Auto_test)$class
confusionMatrix(qda_auto_pred, as.factor(Auto_test$mpg01), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 19
##          1  0 74
##
##                Accuracy : 0.8061
##                  95% CI : (0.7139, 0.879)
##     No Information Rate : 0.949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2844
##  Mcnemar's Test P-Value : 3.636e-05
##
##             Sensitivity : 0.7957
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.2083
##              Prevalence : 0.9490
##          Detection Rate : 0.7551
##    Detection Prevalence : 0.7551
##       Balanced Accuracy : 0.8978
##
```

```
##        'Positive' Class : 1
##
```

The QDA model's overall test error is 19%. The sensitivity and specificity of the model is 0.79 and 1.0 respectively.

(f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
log_reg_auto = glm(mpg01 ~ cylinders + displacement + horsepower + weight + year + origin, family = "bi
log_reg_auto_pred = predict(log_reg_auto, newdata = Auto_test, type = "response")
log_reg_auto_pred[log_reg_auto_pred >= 0.5] = 1
log_reg_auto_pred[log_reg_auto_pred < 0.5] = 0
confusionMatrix(as.factor(log_reg_auto_pred), as.factor(Auto_test$mpg01), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 17
##          1  0 76
##
##                Accuracy : 0.8265
##                  95% CI : (0.7369, 0.8956)
##     No Information Rate : 0.949
##     P-Value [Acc > NIR] : 0.9999978
##
##                   Kappa : 0.3133
##  Mcnemar's Test P-Value : 0.0001042
##
##             Sensitivity : 0.8172
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.2273
##              Prevalence : 0.9490
##          Detection Rate : 0.7755
##    Detection Prevalence : 0.7755
##       Balanced Accuracy : 0.9086
##
##        'Positive' Class : 1
##
```

The logistic regression model's overall test error is 17%. The sensitivity and specificity of the model is 0.81 and 1.0 respectively.

(g) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

```
set.seed(1)
trControl = trainControl(method  = "cv",
                         number  = 5)
Auto$mpg01 = as.factor(Auto$mpg01)
```

```r
train(mpg01 ~ cylinders + displacement + horsepower + weight + year + origin,
      method     = "knn",
      tuneGrid   = expand.grid(k = 1:20),
      trControl  = trControl,
      metric     = "Kappa",
      data       = Auto)
```

```
## k-Nearest Neighbors
##
## 392 samples
##   6 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 314, 312, 314, 314, 314
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8801923  0.7603846
##    2  0.8572436  0.7144872
##    3  0.8750641  0.7501282
##    4  0.8698718  0.7397436
##    5  0.8826923  0.7653846
##    6  0.8699359  0.7398718
##    7  0.8775000  0.7550000
##    8  0.8825641  0.7651282
##    9  0.8775641  0.7551282
##   10  0.8876923  0.7753846
##   11  0.8929487  0.7858974
##   12  0.8826923  0.7653846
##   13  0.8853205  0.7706410
##   14  0.8725641  0.7451282
##   15  0.8851923  0.7703846
##   16  0.8801282  0.7602564
##   17  0.8800641  0.7601282
##   18  0.8750641  0.7501282
##   19  0.8801923  0.7603846
##   20  0.8775641  0.7551282
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was k = 11.
```

```r
knn_auto = knn(Auto_train[,c("cylinders", "displacement", "horsepower", "weight", "year", "origin")],
               Auto_test[,c("cylinders", "displacement", "horsepower", "weight", "year", "origin")],
               Auto_train$mpg01, k = 11)
confusionMatrix(knn_auto, as.factor(Auto_test$mpg01), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 22
##          1  0 71
```

```
##
##                Accuracy : 0.7755
##                  95% CI : (0.6801, 0.8536)
##     No Information Rate : 0.949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2477
##  Mcnemar's Test P-Value : 7.562e-06
##
##             Sensitivity : 0.7634
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.1852
##              Prevalence : 0.9490
##          Detection Rate : 0.7245
##    Detection Prevalence : 0.7245
##       Balanced Accuracy : 0.8817
##
##        'Positive' Class : 1
##
```

The KNN (K = 11) model's overall test error is 22%. The sensitivity and specificity of the model is 0.76 and 1.0 respectively.

12. This problem involves writing functions.

(a) Write a function, Power(), that prints out the result of raising 2 to the 3rd power. In other words, your function should compute 2^3 and print out the results.

```
Power = function(){
  return(2^3)
}
```

(b) Create a new function, Power2(), that allows you to pass any two numbers, x and a, and prints out the value of x^a.

```
Power2 = function(x,a){
  return(x^a)
}
```

(c) Using the Power2() function that you just wrote, compute 10^3, 8^17, and 131^3

```
Power2(10,3)
```

```
## [1] 1000
```

```
Power2(8,17)
```

```
## [1] 2.2518e+15
```

```
Power2(131,3)
```

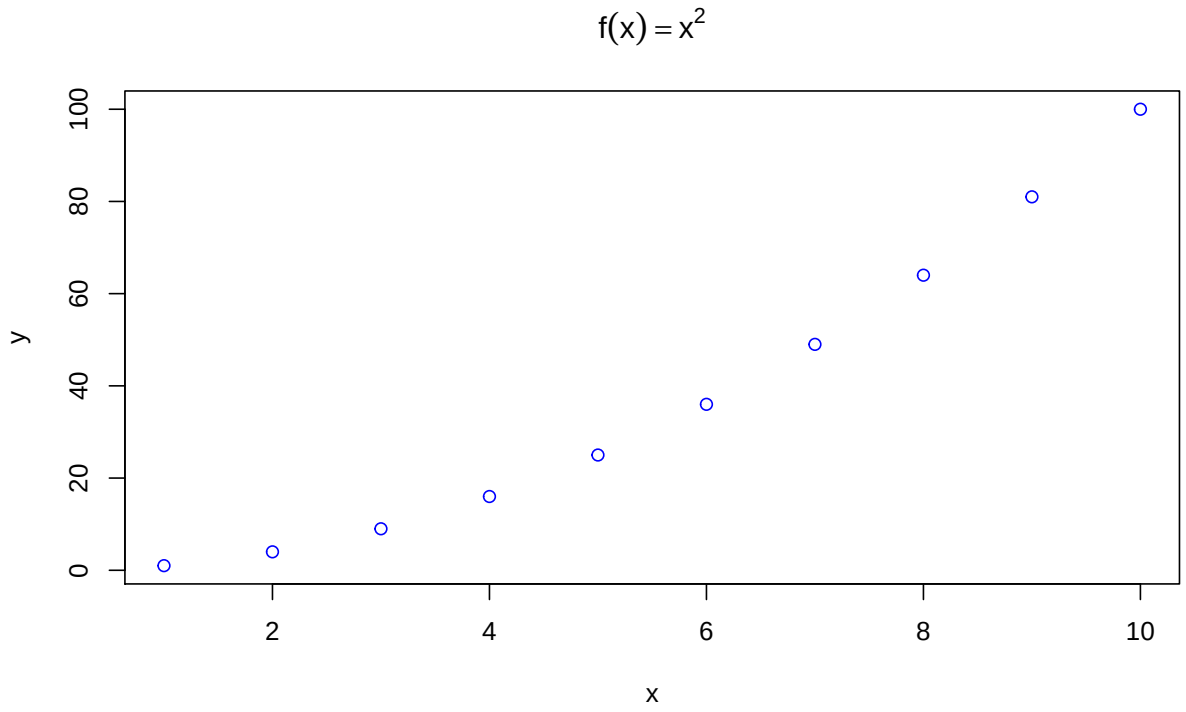
```
## [1] 2248091
```

(d) Now create a new function, Power3(), that actually returns the result x^a as an R object, rather than simply printing it to the screen.

```
Power3 = function(x,a){
  result = Power2(x,a)
}
```

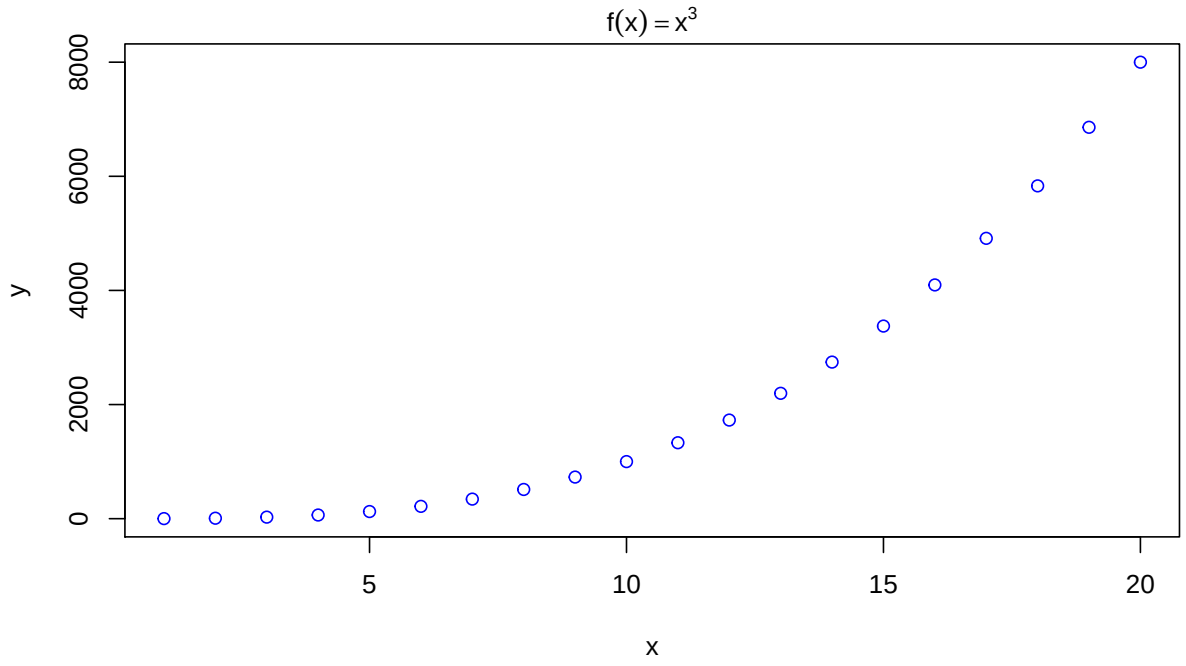
(e) Now using the Power3() function, create a plot of $f(x) == x^2$. The x-axis should display a range of integers from 1 to 10, and the y-axis should display $x2. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the log-scale.

```
x = seq(1,10,by=1)
plot(x,Power3(x,2), main = expression(f(x)==x^2),
                    xlab = "x",
                    ylab = "y",
                    col = "blue")
```



(f) Create a function, PlotPower(), that allows you to create a plot of x against x^a for a fixed a and for a range of values of x.

```
PlotPower = function(x,a){
  plot(x,Power3(x,a), main = mtext(bquote(f(x)==x^.(a))),
                      xlab = "x",
                      ylab = "y",
                      col = "blue")
}
PlotPower(1:20,3)
```

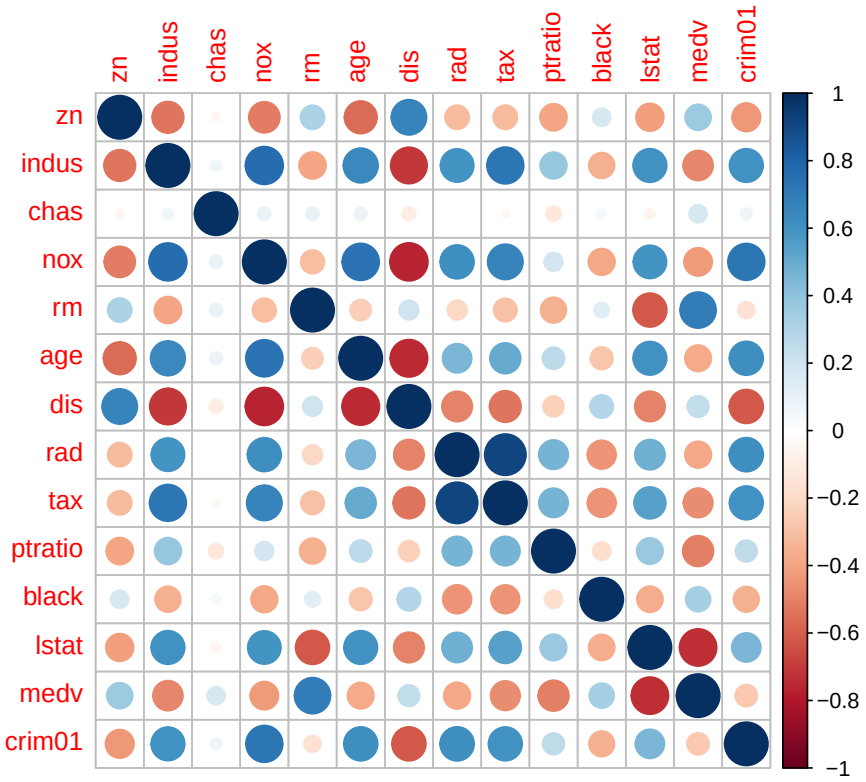


13. Using the Boston data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

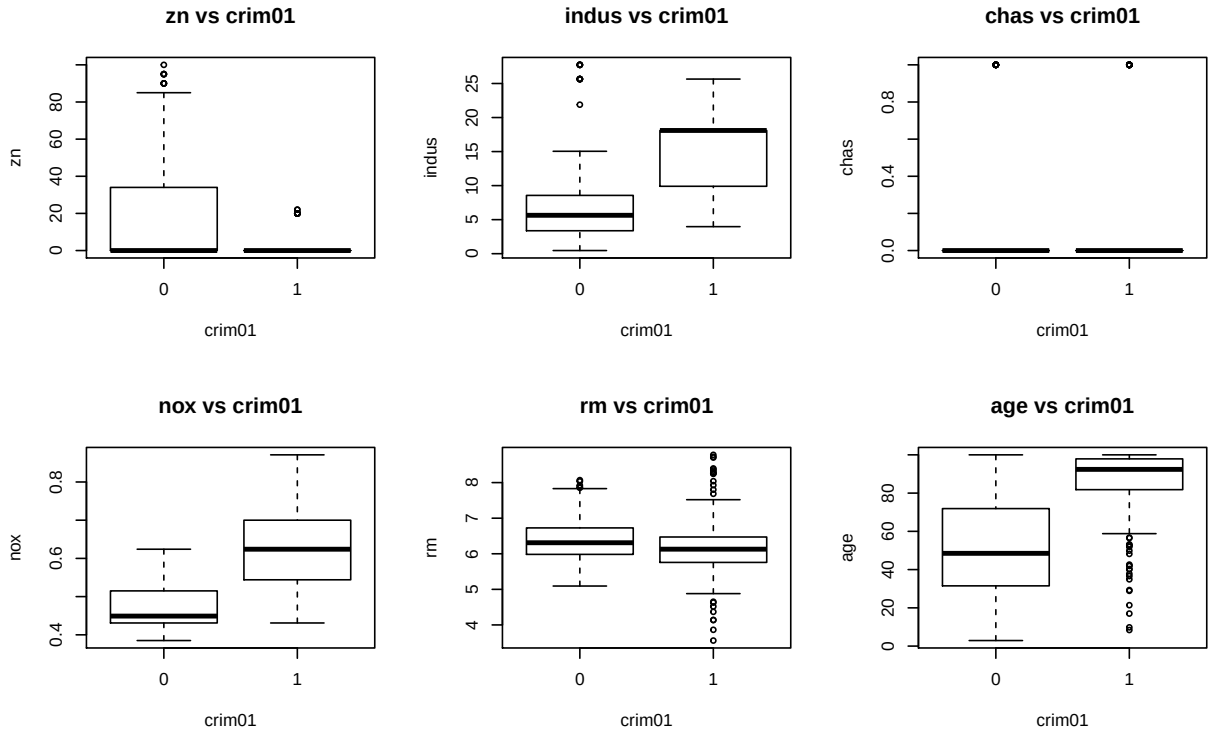
```
library(MASS)

#Turn crime rate into binary variable
crim01_median = median(Boston$crim)
Boston$crim01 = NA
Boston[Boston$crim >= crim01_median,]$crim01 = 1
Boston[Boston$crim < crim01_median,]$crim01 = 0
```

Exploratory Data Analysis

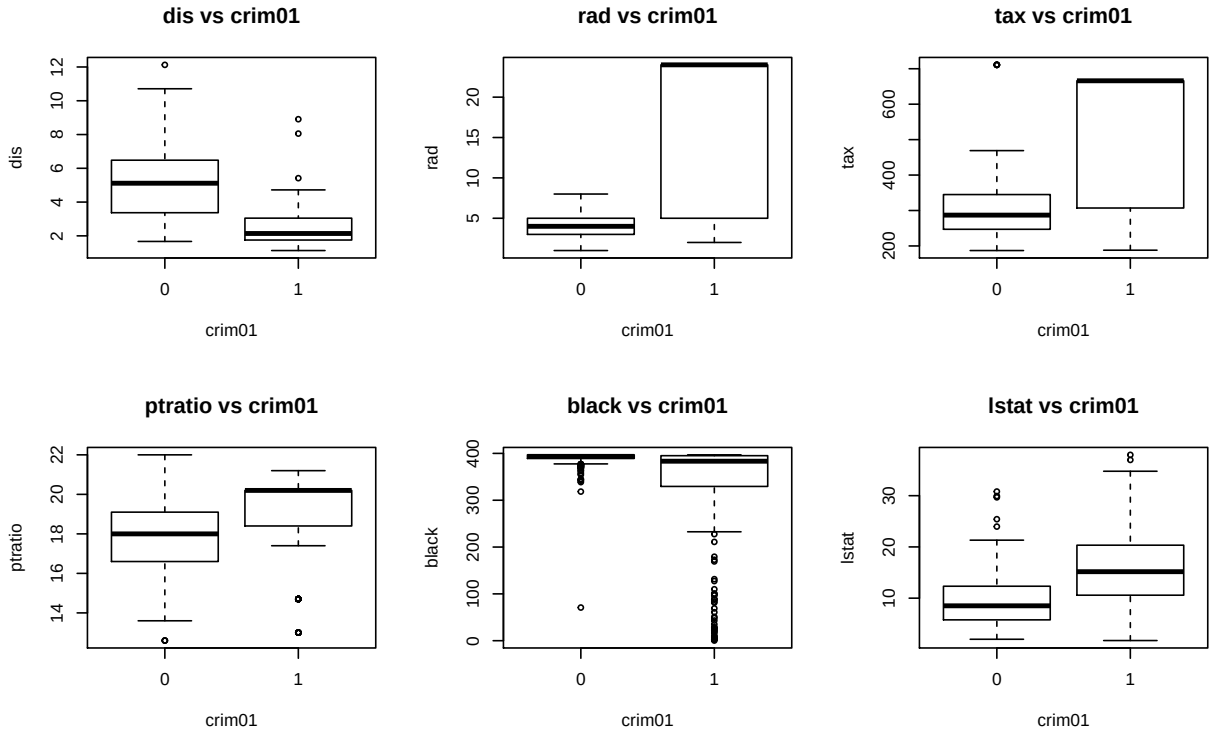
```
#correlation plot
M = cor(Boston[,2:15])
corrplot(M)
```



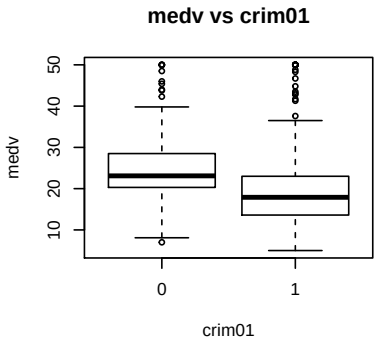

```
#boxplot against crim01
par(mfrow=c(2,3))

for (i in 2:7){
  boxplot(Boston[,i] ~ Boston$crim01,
          main = paste0(colnames(Boston)[i], " vs crim01"),
          xlab = "crim01",
          ylab = colnames(Boston)[i])
}
```




```
par(mfrow=c(2,3))
for (i in 8:14){
  boxplot(Boston[,i] ~ Boston$crim01,
          main = paste0(colnames(Boston)[i], " vs crim01"),
          xlab = "crim01",
          ylab = colnames(Boston)[i])
}
```




```
#Create training and test set
set.seed(1)
train_boston = nrow(Boston)
train_boston = Boston[sample(train_boston,size = train_boston*0.75, replace = FALSE),]
test_boston = Boston[-as.numeric(rownames(train_boston)),]
```



Logistic Regression

```
log_reg_bos = glm(crim01 ~ indus + nox + age + dis + rad + tax + lstat + black, family = "binomial", da
log_reg_bos_full = glm(crim01 ~ ., family = "binomial", data = train_boston)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
stargazer(log_reg_bos_full)
```

```
##
## % Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvar
## % Date and time: Wed, Jun 27, 2018 - 1:16:09 PM
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{1}{c}{\textit{Dependent variable:}} \\
## \cline{2-2}
## \\[-1.8ex] & crim01 \\
## \hline \\[-1.8ex]
##  crim & 793.945 \\
##   & (16,093.250) \\
##   & \\
##  zn & 1.056 \\
##   & (74.411) \\
##   & \\
##  indus & $-$3.027 \\
```

```
##   & (522.474) \\
##   & \\
##  chas & 25.528 \\
##   & (14,510.210) \\
##   & \\
##  nox & 298.105 \\
##   & (43,562.550) \\
##   & \\
##  rm & $-$11.797 \\
##   & (1,870.706) \\
##   & \\
##  age & 0.460 \\
##   & (106.564) \\
##   & \\
##  dis & $-$5.465 \\
##   & (1,873.880) \\
##   & \\
##  rad & $-$12.681 \\
##   & (1,610.794) \\
##   & \\
##  tax & $-$0.072 \\
##   & (35.907) \\
##   & \\
##  ptratio & 12.035 \\
##   & (2,804.559) \\
##   & \\
##  black & 0.015 \\
##   & (19.268) \\
##   & \\
##  lstat & $-$1.319 \\
##   & (145.708) \\
##   & \\
##  medv & 4.706 \\
##   & (202.583) \\
##   & \\
##  Constant & $-$511.630 \\
##   & (52,424.220) \\
##   & \\
## \hline \\[-1.8ex]
## Observations & 379 \\
## Log Likelihood & $-$0.00001 \\
## Akaike Inf. Crit. & 30.000 \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

```
stargazer(log_reg_bos)
```

```
##
## % Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvar
## % Date and time: Wed, Jun 27, 2018 - 1:16:10 PM
## \begin{table}[!htbp] \centering
##   \caption{}
```

```
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{1}{c}{\textit{Dependent variable:}} \\
## \cline{2-2}
## \\[-1.8ex] & crim01 \\
## \hline \\[-1.8ex]
##  indus & $-$0.080 \\
##   & (0.053) \\
##   & \\
##  nox & 44.519$^{***}$ \\
##   & (8.585) \\
##   & \\
##  age & 0.037$^{***}$ \\
##   & (0.012) \\
##   & \\
##  dis & 0.362$^{**}$ \\
##   & (0.178) \\
##   & \\
##  rad & 0.517$^{***}$ \\
##   & (0.135) \\
##   & \\
##  tax & $-$0.008$^{**}$ \\
##   & (0.003) \\
##   & \\
##  lstat & $-$0.052 \\
##   & (0.043) \\
##   & \\
##  black & $-$0.006 \\
##   & (0.005) \\
##   & \\
##  Constant & $-$24.609$^{***}$ \\
##   & (5.190) \\
##   & \\
## \hline \\[-1.8ex]
## Observations & 379 \\
## Log Likelihood & $-$85.875 \\
## Akaike Inf. Crit. & 189.751 \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

```
#Out-of-sample testing
log_reg_bos_pred = predict(log_reg_bos, newdata=test_boston, type="response")
log_reg_bos_full_pred = predict(log_reg_bos_full, newdata=test_boston, type="response")

log_reg_bos_full_pred[log_reg_bos_full_pred >= 0.5] = 1
log_reg_bos_full_pred[log_reg_bos_full_pred < 0.5] = 0

log_reg_bos_pred[log_reg_bos_pred >= 0.5] = 1
log_reg_bos_pred[log_reg_bos_pred < 0.5] = 0
```

```
cm1 = confusionMatrix(as.factor(log_reg_bos_full_pred), as.factor(test_boston$crim01), positive = "1")
cm2 = confusionMatrix(as.factor(log_reg_bos_pred), as.factor(test_boston$crim01), positive = "1")
```

LDA

```
lda_boston_full = lda(crim01 ~ ., data = train_boston)
lda_boston = lda(crim01 ~ indus + nox + age + dis + rad + tax + lstat + black, data = train_boston)


#Out-of-sample testing
lda_bos_full_pred = predict(lda_boston_full, newdata=test_boston)$class
lda_bos_pred = predict(lda_boston, newdata=test_boston)$class


cm3 = confusionMatrix(as.factor(lda_bos_full_pred), as.factor(test_boston$crim01), positive = "1")
cm4 = confusionMatrix(as.factor(lda_bos_pred), as.factor(test_boston$crim01), positive = "1")
```

QDA

```
qda_boston = qda(crim01 ~ indus + nox + age + dis + rad + tax + lstat + black, data = train_boston)
qda_boston_full = qda(crim01 ~ ., data = train_boston)

#Out-of-sample testing
qda_bos_pred = predict(qda_boston, newdata=test_boston)$class
qda_bos_full_pred = predict(qda_boston_full, newdata=test_boston)$class

cm5 = confusionMatrix(as.factor(qda_bos_full_pred ), as.factor(test_boston$crim01), positive = "1")
cm6 = confusionMatrix(as.factor(qda_bos_pred), as.factor(test_boston$crim01), positive = "1")
```

KNN

```
#KNN cross validation
set.seed(1)
trControl = trainControl(method  = "cv",
                         number  = 5)

train_boston$crim01 = as.factor(train_boston$crim01)

train(crim01 ~ indus + nox + age + dis + rad + tax + lstat + black,
      method     = "knn",
      tuneGrid   = expand.grid(k = 1:20),
      trControl  = trControl,
      metric     = "Kappa",
      data       = train_boston)
```

```
## k-Nearest Neighbors
##
## 379 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 303, 303, 303, 303, 304
## Resampling results across tuning parameters:
##
```

```
##   k   Accuracy   Kappa
##    1  0.9156491  0.8311543
##    2  0.8971579  0.7941107
##    3  0.9209474  0.8418131
##    4  0.9104211  0.8207605
##    5  0.9051579  0.8101975
##    6  0.8998596  0.7996058
##    7  0.9024912  0.8048689
##    8  0.8892632  0.7784336
##    9  0.8813333  0.7625901
##   10  0.8787018  0.7572829
##   11  0.8813684  0.7626442
##   12  0.8787368  0.7573370
##   13  0.8761053  0.7520739
##   14  0.8734737  0.7468107
##   15  0.8655439  0.7310111
##   16  0.8655789  0.7310953
##   17  0.8629123  0.7257554
##   18  0.8708772  0.7416757
##   19  0.8708772  0.7416318
##   20  0.8629123  0.7257114
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
knn_boston_full = knn(train_boston[,c(-1,-15)],
                 test_boston[,c(-1,-15)],
                 train_boston$crim01, k = 3)

knn_boston = knn(train_boston[,c("indus", "nox", "age", "dis", "rad", "tax", "lstat", "black")],
                 test_boston[,c("indus", "nox", "age", "dis", "rad", "tax", "lstat", "black")],
                 train_boston$crim01, k = 3)

cm7 = confusionMatrix(knn_boston_full,  as.factor(test_boston$crim01), positive = "1")
cm8 = confusionMatrix(knn_boston,  as.factor(test_boston$crim01), positive = "1")
```

```
Accuracy = c(cm1$overall[[1]],cm2$overall[[1]],
             cm3$overall[[1]],cm4$overall[[1]],
             cm5$overall[[1]],cm6$overall[[1]],
             cm7$overall[[1]],cm8$overall[[1]])

Sensitivity = c(cm1$byClass[[1]],cm2$byClass[[1]],
                cm3$byClass[[1]],cm4$byClass[[1]],
                cm5$byClass[[1]],cm6$byClass[[1]],
                cm7$byClass[[1]],cm8$byClass[[1]])

Specificity = c(cm1$byClass[[2]],cm2$byClass[[2]],
                cm3$byClass[[2]],cm4$byClass[[2]],
                cm5$byClass[[2]],cm6$byClass[[2]],
                cm7$byClass[[2]],cm8$byClass[[2]])

comparison = data.frame(Accuracy,Sensitivity,Specificity)
rownames(comparison) = c("LogRegFull",
                         "LogRegSelect",
                         "LDAFull",
```

```
                          "LDASelect",
                          "QDAFull",
                          "QDASelect",
                          "KNNFull",
                          "KNNSelect")
knitr::kable(comparison, caption = "Comparison of Classification Models for Boston Housing Data")
```

Table 1: Comparison of Classification Models for Boston Housing Data

	Accuracy	Sensitivity	Specificity
LogRegFull	0.9763780	0.9677419	0.9846154
LogRegSelect	0.8897638	0.8548387	0.9230769
LDAFull	0.8582677	0.7580645	0.9538462
LDASelect	0.8503937	0.7258065	0.9692308
QDAFull	0.9448819	0.8870968	1.0000000
QDASelect	0.8582677	0.7258065	0.9846154
KNNFull	0.9212598	0.8548387	0.9846154
KNNSelect	0.8976378	0.8548387	0.9384615

A model with all predictors and a parsimonious model with selection of predictors were modeled using logistic regression, LDA, QDA, and KNN classifiers. The predictors for the parsimonious model were proportion of non-retail business acres per town, nitrogen oxides concentration, proportion of owner-occupied units built prior to 1940, weighted mean of distances to five Boston employment centers, index of accessibility to radial highways, full-value property-tax rate per $10,000, percent lower status of the population, and proportion of blacks by town. These predictors were selected by visually inspecting which predictors were somewhat correlated with the the binary crime variable and exploring box plots for clear visual separation between the crime classes and each predictor.

The full models for each method were on average 5 percent more accurate than smaller models. The best model for out-of-sample prediction was the full logistic regression model at 97.5 percent. KNN delivered the best smaller model at 89.7 percent. The QDA full model achieved 100 percent specificity and the logistic regression full model also had the highest sensitivity.