Homework3

Nam Jun Lee

11/05/2021

Q1. Question 1 of Chapter 4 of the ISLR book. (Page 168).

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.

Equation (4.2)

$$p(X) = \frac{e^{\beta 0+\beta 1X}}{1+e^{\beta 0+\beta 1X}}$$

Equation (4.3)

$$\frac{p(X)}{1-p(X)} = e^{\beta 0+\beta 1X}$$

Express (4.2) in a different way, $1-p(X) = 1-\frac{e^{\beta 0+\beta 1X}}{1+e^{\beta 0+\beta 1X}} = \frac{1}{1+e^{\beta 0+\beta 2X}}$ **->** $\frac{1}{1-p(X)} = 1+e^{\beta 0+\beta 2X}$ **->** $p(X)\frac{1}{1-p(X)} = \frac{e^{\beta 0+\beta 1X}}{1+e^{\beta 0+\beta 1X}}(1+e^{\beta 0+\beta 1X})$ **->** $\frac{p(X)}{1-p(X)} = e^{\beta 0+\beta 1X}$

Thus, **(4.2) is equivalent to (4.3)**.

Q2. Question 2 of Chapter 4 of the ISLR book. (Page 168).

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(μk, σ^2) distribution, the Bayes' classifier assigns an observation to the class for which the discriminant function is maximized.

(4.12)

$$pk(x) = \frac{\pi k\frac{1}{\sqrt{2\pi}\sigma}exp(-\frac{1}{2\sigma^2}(x-\mu k)^2)}{\sum_{l=1}^{K}\pi l\frac{1}{\sqrt{2\pi}\sigma}exp(-\frac{1}{2\sigma^2}(x-\mu l)^2)}$$

(4.13)

$$\delta k(x) = x * \frac{\mu k}{\sigma^2} - \frac{\mu^2 k}{2\sigma^2} + log(\pi k)$$

The Bayes classifier finds the largest class k because observations must be assigned to the grade where the discriminant function is maximized in (4.12).

$pk(x) = \frac{\pi k e^{-(\frac{1}{2\sigma^2})(x-\mu k^2)}}{\sum_{l=1}^{K} \pi l e^{-(\frac{1}{2\sigma^2})(x-\mu l^2)}}$

Use the logarithmic function to remove denominators and terms unrelated to the largest class k.
$logpk(x) = \delta k(X) = log\pi k - (\frac{1}{2\sigma^2})(x-\mu k)^2 - log\sum_{l=1}^{K} \pi l e^{-(\frac{1}{2\sigma^2})(x-\mu l)^2} = log\pi k - (\frac{x^2}{2\sigma^2} - \frac{\pi\mu k}{\sigma^2} + \frac{\mu^2 k}{2\sigma^2})$

Independent terms can be removed to find the discriminant function:
$\delta k(X) = log(\pi k) - \frac{x\mu k}{\sigma^2} + \frac{\mu^2 k}{2\sigma^2}$

Thus, **it may be seen that the class maximizing (4.12) is equivalent to a class maximizing (4.13)**.

Q3. Question 5 of Chapter 4 of the ISLR book. (Page 169).

We now examine the differences between LDA and QDA.

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?

If the Bayes decision boundary is linear, the LDA will perform better as a test set because QDA may be overfitting linearity in the test set. On the other hand, in the training set, QDA will perform better because QDA is more flexible than LDA.

Training set: QDA
Test set: LDA

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?

If the Bayes decision boundary is nonlinear, **QDA** will perform better on both the training set and the test set because of its high flexibility.

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?

Accuracy may vary depending on whether the boundary is linear or nonlinear, but more data may not fit QDA well into test data. Therefore, increasing the sample size n does **unchange** the test prediction accuracy of QDA for LDA.

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.

If the sample point is small, an error rate may occur due to overfitting if a flexible method such as QDA is used.
Thus, **False**.

Q4. Question 6 of Chapter 4 of the ISLR book. (Page 170).

Suppose we collect data for a group of students in a statistics class with variables X1 = hours studied, X2 = 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.

Probability equation:

$$\hat{p}(X) = \frac{e^{-\beta 0+\beta 1X1+X2}}{\beta 2 + e^{-\beta 0+\beta 1X1+X2}}$$

Therefore,

$$\hat{p}(X) = \frac{e^{-6+0.05X1+X2}}{1 + e^{-6+0.05X1+X2}}$$

Put X1 = 40, X2= 3.5:

$$\hat{p}(X) = \frac{e^{-6+0.05(40)+(3.5)}}{1 + e^{-6+0.05(40)+(3.5)}} = 0.3775$$

Using R:

```
# fit a logistic regression
prob = function(X1, X2) {
    Y = exp(-6 + 0.05 * X1 + 1 * X2)
    return(Y/(1 + Y))
}
# put that a student who studies for 40h and has an undergrad GPA of 3.5
prob(40, 3.5)
```

```
## [1] 0.3775407
```

Therefore, **0.3775**.

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?

Probability equation using part(a) of 50 % chance:

$$\hat{p}(X) = \frac{e^{-6+0.05X1+3.5}}{1 + e^{-6+0.05X1+3.5}} = 0.5$$

This same as:

$$\hat{p}(X) = e^{-6+0.05X1+3.5} = 1$$

Logarithm of both sides:

$$X1 = \frac{2.5}{0.05} = 50$$

Using R:

```
# hours selection from 30 to 60.
hours = seq(30, 60, 1)
# function to apply part(a) and getting an A (3.5) in the class each hours
p = mapply(hours, 3.5, FUN = prob)
# paste 'h' to probs name
names(p) <- paste0(hours, " h")
# show what percent chance of 30 to 60 hours getting an A in the class.
p
```

```
##      30 h      31 h      32 h      33 h      34 h      35 h      36 h      37 h
## 0.2689414 0.2788848 0.2890505 0.2994329 0.3100255 0.3208213 0.3318122 0.3429895
##      38 h      39 h      40 h      41 h      42 h      43 h      44 h      45 h
## 0.3543437 0.3658644 0.3775407 0.3893608 0.4013123 0.4133824 0.4255575 0.4378235
##      46 h      47 h      48 h      49 h      50 h      51 h      52 h      53 h
## 0.4501660 0.4625702 0.4750208 0.4875026 0.5000000 0.5124974 0.5249792 0.5374298
##      54 h      55 h      56 h      57 h      58 h      59 h      60 h
## 0.5498340 0.5621765 0.5744425 0.5866176 0.5986877 0.6106392 0.6224593
```

Therefore, **50** hours.

Q5. Question 7 of Chapter 4 of the ISLR book. (Page 170).

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 $\overline{X} = 10$, while the mean for those that didn't was $\overline{X} = 0$. 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.

Hint: Recall that the density function for a normal random variable is f(x) = $\frac{1}{2\pi\sigma^2}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$. You will need to use Bayes' theorem.

Bayes' theorem: $pk(X) = \frac{\pi k \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu k)^2)}{\sum_{l=1}^{k} \pi l \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu l)^2)}$

Set given values: $\pi Yes = 0.8$, $\pi No = 0.2$, $\mu Yes = 10$, $\mu No = 0$, $\hat{\sigma}^2 = 36$

values in equation: $pYes(4) = \frac{0.8e^{-\frac{1}{2*36}(4-10)^2}}{0.8e^{-\frac{1}{2*36}(4-10)^2}+0.2e^{-\frac{1}{2*36}(4-0)^2}} = 0.75185$

Therefore, **0.75185**.

Q6. Question 10 of Chapter 4 of the ISLR book (Page 171) (all parts except part (g)). You may also consider using regularized logistic regression to select predictors.

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.

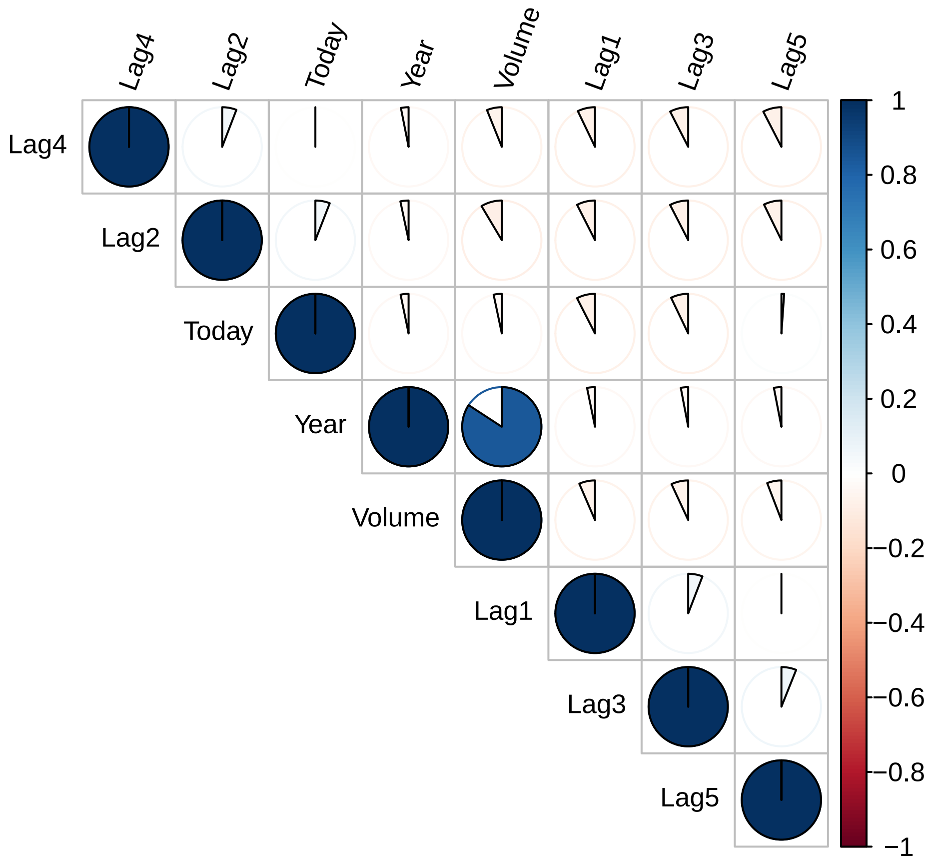
```r
# Weekly dataset into df
df <- ISLR::Weekly
```

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

```r
# summary of Weekly data
summary(df)
```

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

```r
# find patterns
corr <- cor(df[, -9])
corrplot(corr, method = "pie", type = "upper", tl.col = "black", tl.srt = 70, tl.cex = 0.8,
    order = "hclust")
```



When checking the correlation plot, there is a strong linear relationship between the **Year** variable and the **Volume** variable. Other variables show linearly low linear relationships.

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?

```
# logistic regression with Direction as five Lags and Volume
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = df,
    family = binomial)
# summary result
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## 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
```

When checking the results, **Lag2** is the only statistically significant variable. Other variables fail to reject the null hypothesis because the p-value is greater than 0.05.

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.

```r
# predictions for all values in the training set
prob_df <- predict(glm.fit, type = "response")
# create all values 'Down' in pred_df
pred_df <- rep("Down", length(prob_df))
# all of the elements for which the predicted probability of a market increase
# exceeds 0.5 in pred_df
pred_df[prob_df > 0.5] = "Up"
# confusion matrix
table(pred_df, df$Direction)
```

```
##
## pred_df Down  Up
##    Down   54  48
##    Up    430 557
```

```r
# fraction of days for which the prediction was correct
mean(pred_df == df$Direction)
```

```
## [1] 0.5610652
```

```r
# training error rate
mean(pred_df != df$Direction)
```

```
## [1] 0.4389348
```

Therefore, percentage of current predictions
Training correct prediction:
$\frac{(54+557)}{(54+48+430+557)} = 0.5610652$ **56.11 %**
Training error rate:
$1 - 0.5610651974 = 0.4389348$ **43.89 %**
Also, specificity is $\frac{557}{48+557} = 0.92066115702$; **92.06 %**
On the contrary, sensitivity is $\frac{54}{430+54} = 0.11157024793$; **11.16 %**

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).

```
# create a vector corresponding to the observations from 1990 through 2008
train_data = (df$Year < 2009)
# set correct predictions for the held out data (from 2009 and 2010)
week.20092010 = df[!train_data, ]
direction.20092010 = df$Direction[!train_data]
```

```
# fit model with Lag2 only
glm.fit1 <- glm(Direction ~ Lag2, data = df, family = binomial, subset = train_data)
# predictions data from 2009 and 2010 in the training set
prob_df1 <- predict(glm.fit1, week.20092010, type = "response")
# create all values 'Down' in pred_df1
pred_df1 <- rep("Down", length(prob_df1))
# all of the elements for which the predicted probability of a market increase
# exceeds 0.5 in pred_df1
pred_df1[prob_df1 > 0.5] = "Up"
# confusion matrix
table(pred_df1, direction.20092010)
```

```
##          direction.20092010
## pred_df1 Down Up
##     Down    9  5
##     Up     34 56
```

```
# fraction of days for which the prediction was correct
mean(pred_df1 == direction.20092010)
```

```
## [1] 0.625
```

```
# training error rate
mean(pred_df1 != direction.20092010)
```

```
## [1] 0.375
```

Therefore, percentage of current logistic predictions
Training correct prediction:
$\frac{(9+56)}{(9+5+34+56)} = 0.625$ **62.5 %**

Training error rate:
$1 - 0.625 = 0.375$ **37.5 %**
Also, specificity is $\frac{56}{5+56} = 0.91803278688$; **91.80 %**
On the contrary, sensitivity is $\frac{9}{9+34} = 0.20930232558$; **20.93 %**

e. Repeat (d) using LDA.

```
# fit classifier
lda.fit <- lda(Direction ~ Lag2, data = df, subset = train_data)
# predictions data from 2009 and 2010 in the training set
lda.pred <- predict(lda.fit, week.20092010)
# contains LDA's predictions about the movement of the market
lda.class = lda.pred$class
# confusing matrix
table(lda.class, direction.20092010)
```

```
##          direction.20092010
## lda.class Down Up
##      Down    9  5
##      Up     34 56
```

```
# fraction of days for which the prediction was correct
mean(lda.class == direction.20092010)
```

```
## [1] 0.625
```

```
# training error rate
mean(lda.class != direction.20092010)
```

```
## [1] 0.375
```

Therefore, percentage of current LDA predictions
Training correct prediction:
$\frac{(9+56)}{(9+5+34+56)} = 0.625$ **62.5 %**
Training error rate:
$1 - 0.625 = 0.375$ **37.5 %**
Also, specificity is $\frac{56}{5+56} = 0.91803278688$; **91.80 %**
On the contrary, sensitivity is $\frac{9}{9+34} = 0.20930232558$; **20.93 %**

f. Repeat (d) using QDA.

```
# fit classifier
qda.fit <- qda(Direction ~ Lag2, data = df, subset = train_data)
# predictions data from 2009 and 2010 in the training set
qda.pred = predict(qda.fit, week.20092010)
# contains RDA's predictions about the movement of the market
qda.class = qda.pred$class
# confusing matrix
table(qda.class, direction.20092010)
```

```
##          direction.20092010
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```r
# fraction of days for which the prediction was correct
mean(qda.class == direction.20092010)
```

```
## [1] 0.5865385
```

```r
# training error rate
mean(qda.class != direction.20092010)
```

```
## [1] 0.4134615
```

Therefore, percentage of current QDA predictions
Training correct prediction:
$\frac{(0+61)}{(0+0+43+61)} = 0.5865385$ **58.65 %**
Training error rate:
$1 - 0.5865385 = 0.4134615$ **41.35 %**
Also, specificity is $\frac{61}{0+61} = 1$; **100 %**
On the contrary, sensitivity is $\frac{0}{0+43} = 0$; **0 %**

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

Logistic regression QDA and LDA analysis results.
Logistic Regression Accuracy: *63.5 %*
LDA Accuracy: *63.5 %*
QDA Accuracy: *58.65 %*

As such, **Logistic regression** and **LDA** provide better results for this Weekly dataset.

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.

```r
# Logistic regression with interaction (Lag2:Lag1)
glm.fit2 <- glm(Direction ~ Lag2:Lag1, data = df, family = binomial, subset = train_data)
# predictions data from 2009 and 2010 in the training set
prob_df2 <- predict(glm.fit2, week.20092010, type = "response")
# create all values 'Down' in pred_df2
pred_df2 <- rep("Down", length(prob_df2))
# all of the elements for which the predicted probability of a market increase
# exceeds 0.5 in pred_df2
pred_df2[prob_df2 > 0.5] = "Up"
# confusion matrix
table(pred_df2, direction.20092010)
```

```
##         direction.20092010
## pred_df2 Down Up
##     Down    1  1
##     Up     42 60
```

```
# fraction of days for which the prediction was correct
mean(pred_df2 == direction.20092010)
```

```
## [1] 0.5865385
```

```
# training error rate
mean(pred_df2 != direction.20092010)
```

```
## [1] 0.4134615
```

Percentage of current interaction logistic predictions
Training correct prediction:
$\frac{(1+60)}{(1+1+42+60)} = 0.5865385$ **58.65 %**
Training error rate:
$1 - 0.5865385 = 0.4134615$ **41.35 %**
Also, specificity is $\frac{60}{1+60} = 0.98361$; **98.36 %**
On the contrary, sensitivity is $\frac{1}{1+42} = 0.02326$; **0.02 %**

```
# LDA with interaction (Lag2:Lag1)
lda.fit1 <- lda(Direction ~ Lag2:Lag1, data = df, subset = train_data)
# predictions data from 2009 and 2010 in the training set
lda.pred1 <- predict(lda.fit1, week.20092010)
# contains LDA's predictions about the movement of the market
lda.class1 = lda.pred1$class
# confusing matrix
table(lda.class1, direction.20092010)
```

```
##           direction.20092010
## lda.class1 Down Up
##       Down    0  1
##       Up     43 60
```

```
# fraction of days for which the prediction was correct
mean(lda.class1 == direction.20092010)
```

```
## [1] 0.5769231
```

```
# training error rate
mean(lda.class1 != direction.20092010)
```

```
## [1] 0.4230769
```

Percentage of current interaction LDA predictions
Training correct prediction:
$\frac{(0+60)}{(0+1+43+60)} = 0.5769231$ **57.69 %**

Training error rate:
$1 - 0.5769231 = 0.4230769$ **42.31 %**
Also, specificity is $\frac{60}{1+61} = 0.98361$; **98.36 %**
On the contrary, sensitivity is $\frac{0}{0+43} = 0$; **0 %**

```
# qda with interaction (Lag2:Lag1)
qda.fit1 <- qda(Direction ~ Lag2:Lag1, data = df, subset = train_data)
# predictions data from 2009 and 2010 in the training set
qda.pred1 = predict(qda.fit1, week.20092010)
# contains RDA's predictions about the movement of the market
qda.class1 = qda.pred1$class
# confusing matrix
table(qda.class1, direction.20092010)
```

```
##           direction.20092010
## qda.class1 Down Up
##       Down   16 32
##       Up     27 29
```

```
# fraction of days for which the prediction was correct
mean(qda.class1 == direction.20092010)
```

```
## [1] 0.4326923
```

```
# training error rate
mean(qda.class1 != direction.20092010)
```

```
## [1] 0.5673077
```

Percentage of current interaction QDA predictions
Training correct prediction:
$\frac{(16+29)}{(16+32+27+29)} = 0.4326923$ **43.27 %**
Training error rate:
$1 - 0.4326923 = 0.5673077$ **56.73 %**
Also, specificity is $\frac{29}{32+29} = 0.47541$; **47.54 %**
On the contrary, sensitivity is $\frac{16}{16+27} = 0.37209$; **0.3721 %**

```
# create train data
train_x = cbind(df$Lag1, df$Lag2)[train_data, ]
# create test data
test_x = cbind(df$Lag1, df$Lag2)[!train_data, ]
# vector containing the class labels for the training observations
train_direction = df$Direction[train_data]
```

```
# K = 3 predictions data from 2009 and 2010 in the knn
knn_pred = knn(train_x, test_x, train_direction, k = 3)
# confusing matrix
table(knn_pred, direction.20092010)
```

```
##         direction.20092010
## knn_pred Down Up
##     Down   22 29
##     Up     21 32
```

```r
# fraction of days for which the prediction was correct
mean(knn_pred == direction.20092010)
```

```
## [1] 0.5192308
```

```r
# training error rate
mean(knn_pred != direction.20092010)
```

```
## [1] 0.4807692
```

Percentage of current KNN [k = 3] predictions
Training correct prediction:
$\frac{(22+32)}{(22+29+21+32)} = 0.5192308$ **51.92 %**
Training error rate:
$1 - 0.5192308 = 0.4807692$ **48.08 %**
Also, specificity is $\frac{32}{29+32} = 0.52459$; **52.46 %**
On the contrary, sensitivity is $\frac{22}{22+21} = 0.51162$; **51.16 %**

```r
# K = 5 predictions data from 2009 and 2010 in the knn
knn_pred = knn(train_x, test_x, train_direction, k = 5)
# confusing matrix
table(knn_pred, direction.20092010)
```

```
##          direction.20092010
## knn_pred Down Up
##     Down   22 32
##     Up     21 29
```

```r
# fraction of days for which the prediction was correct
mean(knn_pred == direction.20092010)
```

```
## [1] 0.4903846
```

```r
# training error rate
mean(knn_pred != direction.20092010)
```

```
## [1] 0.5096154
```

Percentage of current KNN [k = 5] predictions
Training correct prediction:
$\frac{(22+29)}{(22+32+21+29)} = 0.4903846$ **49.04 %**
Training error rate:
$1 - 0.4903846 = 0.5096154$ **50.96 %**
Also, specificity is $\frac{29}{32+29} = 0.47540$; **47.54 %**
On the contrary, sensitivity is $\frac{22}{22+21} = 0.51162$; **51.16 %**

```r
# K = 7 predictions data from 2009 and 2010 in the knn
knn_pred = knn(train_x, test_x, train_direction, k = 7)
# confusing matrix
table(knn_pred, direction.20092010)
```

```
##         direction.20092010
## knn_pred Down Up
##     Down   22 28
##     Up     21 33
```

```
# fraction of days for which the prediction was correct
mean(knn_pred == direction.20092010)
```

```
## [1] 0.5288462
```

```
# training error rate
mean(knn_pred != direction.20092010)
```

```
## [1] 0.4711538
```

Percentage of current KNN [k = 7] predictions
Training correct prediction:
$\frac{(22+33)}{(22+28+21+33)} = 0.5288462$ **52.88 %**
Training error rate:
$1 - 0.5288462 = 0.4711538$ **47.12 %**
Also, specificity is $\frac{33}{28+33} = 0.54098$; **54.10 %**
On the contrary, sensitivity is $\frac{22}{22+21} = 0.51162$; **51.16 %**

Thus, Experiments using combinations of different predictors, including possible transformations and interactions, resulted in each accuracy:
Interaction Logistic: **58.65 %**
Interaction LDA: **57.69 %**
Interaction QDA: **43.27 %**
Interaction KNN (k=3): **51.92 %**
Interaction KNN (k=5): **49.04 %**
Interaction KNN (k=7): **52.88 %**

As such, **Logistic regression with interaction** provide better results for this dataset.

Q7. Question 13 of Chapter 4 of the ISLR book. (Page 173). (Use LDA, QDA, logistic regression, regularized logistic regression, you may also consider linear regression).

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.

Import Boston data set:

```
# Boston dataset into bt
bt <- MASS::Boston
# summary bt dataset
summary(bt)
```

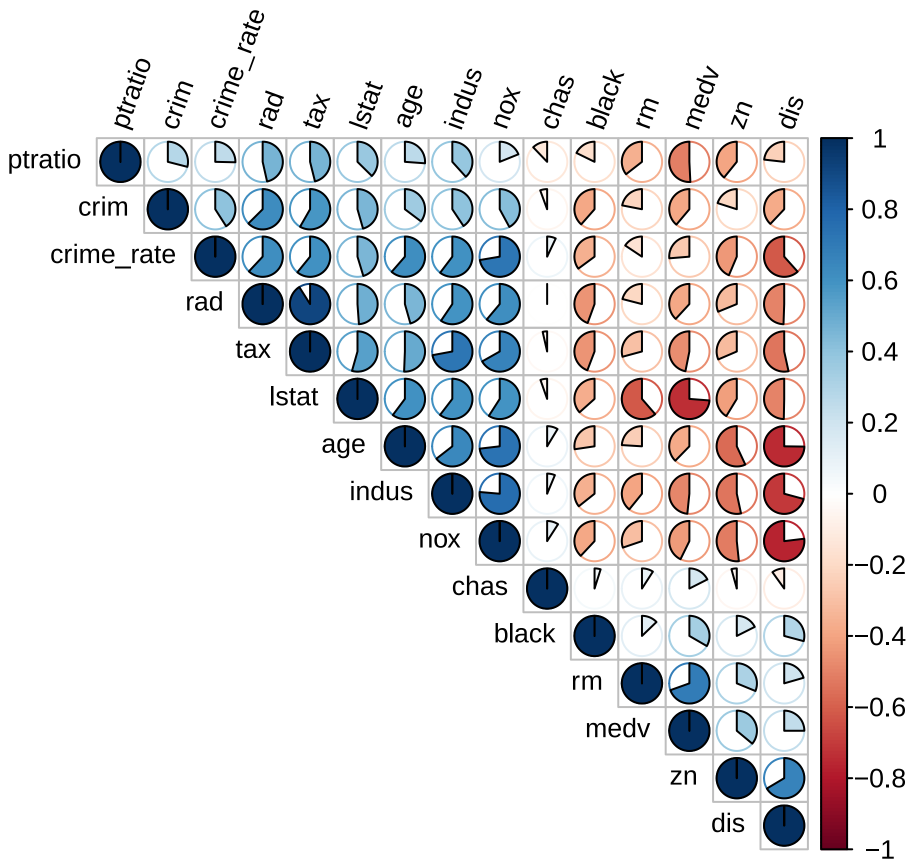
```
##       crim                zn             indus             chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   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
```

There are 14 variables in this dataset.

```r
# given suburb has a crime rate above or below the median
crime_rate <- rep(0, nrow(bt))
crime_rate[bt$crim > median(bt$crim)] <- 1
bt_new <- data.frame(bt, crime_rate)
# check str new boston data
str(bt_new)
```

```
## 'data.frame':    506 obs. of  15 variables:
##  $ crim      : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn        : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus     : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox       : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm        : num  6.58 6.42 7.18 7 7.15 ...
##  $ age       : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis       : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad       : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax       : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio   : num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black     : num  397 397 393 395 397 ...
##  $ lstat     : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv      : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
##  $ crime_rate: num  0 0 0 0 0 0 0 0 0 0 ...
```

```r
# find patterns
corrB <- cor(bt_new)
corrplot(corrB, method = "pie", type = "upper", tl.col = "black", tl.srt = 70, tl.cex = 0.8,
    order = "hclust")
```



The correlation of this new Boston dataset shows that **rad**, **tax**, **lstat**, **age**, **indus**, and **nox** have a strong positive relationship with the crime_rate variable, so just use these six variables to predictions.

```r
# create test and train use to predictions
train <- 1:(dim(bt_new)[1]/2)
test <- (dim(bt_new)[1]/2 + 1):dim(bt_new)[1]
bt_train <- bt_new[train, ]
bt_test <- bt_new[test, ]
crim_test <- crime_rate[test]
```

```r
# fit logistic model with 6 variables only
glm.fit.bt <- glm(crime_rate ~ rad + tax + lstat + age + indus + nox, data = bt_new,
    family = binomial, subset = train)
# summary logistic model
summary(glm.fit.bt)
```

```
##
## Call:
## glm(formula = crime_rate ~ rad + tax + lstat + age + indus +
##     nox, family = binomial, data = bt_new, subset = train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1427  -0.2250  -0.0271   0.5040   3.3412
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -41.078177   7.806493  -5.262 1.42e-07 ***
## rad           0.944016   0.206414   4.573 4.80e-06 ***
## tax          -0.015031   0.005155  -2.916  0.00355 **
## lstat         0.066691   0.043159   1.545  0.12229
## age          -0.004786   0.013063  -0.366  0.71410
## indus        -0.243206   0.080335  -3.027  0.00247 **
## nox          80.820887  17.091445   4.729 2.26e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.37  on 252  degrees of freedom
## Residual deviance: 144.62  on 246  degrees of freedom
## AIC: 158.62
##
## Number of Fisher Scoring iterations: 8
```

This summary show that, the null hypothesis for **age** and **lstat** cannot be rejected.

```
# predictions for all values in the training set
prob_bt <- predict(glm.fit.bt, bt_test, type = "response")
# create all values '0' in pred_dt
pred_bt <- rep(0, length(prob_bt))
# all of the elements for which the predicted probability of a market increase
# exceeds 0.5 in pred_dt
pred_bt[prob_bt > 0.5] <- 1
# confusion matrix
table(pred_bt, crim_test)
```

```
##        crim_test
## pred_bt   0   1
##       0  73  11
##       1  17 152
```

```
# fraction of days for which the prediction was correct
mean(pred_bt == crim_test)
```

```
## [1] 0.8893281
```

```
# training error rate
mean(pred_bt != crim_test)
```

```
## [1] 0.1106719
```

Percentage of current logistic predictions
Training correct prediction:
$\frac{(73+152)}{(73+11+17+152)} = 0.8893281$ **88.93 %**
Training error rate:
$1 - 0.8893281 = 0.1106719$ **11.07 %**
Also, specificity is $\frac{152}{11+152} = 0.9325$; **93.25 %**
On the contrary, sensitivity is $\frac{73}{73+17} = 0.81111$; **81.11 %**

```
# fit regularized logistic model with 4 variables only
glm.fit.bt1 <- glm(crime_rate ~ rad + tax + indus + nox, data = bt_new, family = binomial,
    subset = train)
# summary regularized logistic model
summary(glm.fit.bt1)
```

```
##
## Call:
## glm(formula = crime_rate ~ rad + tax + indus + nox, family = binomial,
##     data = bt_new, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9891  -0.2149  -0.0412   0.4604   3.2740
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -38.081510   7.150139  -5.326 1.00e-07 ***
## rad           0.833751   0.183216   4.551 5.35e-06 ***
## tax          -0.013737   0.004904  -2.801  0.00509 **
## indus        -0.217507   0.075646  -2.875  0.00404 **
## nox          75.865697  15.705290   4.831 1.36e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.37  on 252  degrees of freedom
## Residual deviance: 147.14  on 248  degrees of freedom
## AIC: 157.14
##
## Number of Fisher Scoring iterations: 8
```

age and lstat cannot reject the null hypothesis, so it can be seen that all variables reject the null hypothesis as a result of removing age and lstat and using the remaining four variables.

```
# predictions for all values in the training set
prob_bt1 <- predict(glm.fit.bt1, bt_test, type = "response")
# create all values '0' in pred_bt1
pred_bt1 <- rep(0, length(prob_bt1))
# all of the elements for which the predicted probability of a market increase
# exceeds 0.5 in pred_bt1
pred_bt1[prob_bt1 > 0.5] <- 1
# confusion matrix
table(pred_bt1, crim_test)
```

```
##         crim_test
## pred_bt1   0   1
##        0  71   6
##        1  19 157
```

```
# fraction of days for which the prediction was correct
mean(pred_bt1 == crim_test)
```

```
## [1] 0.9011858
```

```
# training error rate
mean(pred_bt1 != crim_test)
```

```
## [1] 0.09881423
```

Percentage of current regularized logistic predictions
Training correct prediction:
$\frac{(71+157)}{(71+6+19+157)} = 0.9011858$ **90.12 %**
Training error rate:
$1 - 0.9011858 = 0.09881423$ **9.88 %**
Also, specificity is $\frac{157}{6+157} = 0.96319$; **96.32 %**
On the contrary, sensitivity is $\frac{71}{71+19} = 0.78888$; **78.89 %**

```
# fit LDA
lda.fit.bt <- lda(crime_rate ~ rad + tax + lstat + age + indus + nox, data = bt_new,
    subset = train)
# predictions data in the training set
lda.pred.bt <- predict(lda.fit.bt, bt_test)
# contains LDA's predictions about the movement of the boston
lda.class.bt <- lda.pred.bt$class
# confusing matrix
table(lda.class.bt, crim_test)
```

```
##             crim_test
## lda.class.bt   0   1
##            0  81  19
##            1   9 144
```

```
# fraction of days for which the prediction was correct
mean(lda.class.bt == crim_test)
```

```
## [1] 0.8893281
```

```
# training error rate
mean(lda.class.bt != crim_test)
```

```
## [1] 0.1106719
```

Percentage of current LDA predictions
Training correct prediction:
$\frac{(81+144)}{(81+19+9+144)} = 0.889381$ **88.94 %**
Training error rate:
$1 - 0.8893281 = 0.1106719$ **11.07 %**
Also, specificity is $\frac{144}{19+144} = 0.883435$; **88.34 %**
On the contrary, sensitivity is $\frac{81}{81+9} = 0.9$; **90.00 %**

```
# fit QDA
qda.fit.bt <- qda(crime_rate ~ rad + tax + lstat + age + indus + nox, data = bt_new,
    subset = train)
# predictions data in the training set
qda.pred.bt = predict(qda.fit.bt, bt_test)
# contains QDA's predictions about the movement of the boston
qda.class.bt = qda.pred.bt$class
# confusing matrix
table(qda.class.bt, crim_test)
```

```
##              crim_test
## qda.class.bt   0   1
##            0  83 147
##            1   7  16
```

```
# fraction of days for which the prediction was correct
mean(qda.class.bt == crim_test)
```

```
## [1] 0.3913043
```

```
# training error rate
mean(qda.class.bt != crim_test)
```

```
## [1] 0.6086957
```

Percentage of current QDA predictions
Training correct prediction:
$\frac{(83+16)}{(83+147+7+16)} = 0.3913043$ **39.13 %**
Training error rate:
$1 - 0.3913043 = 0.6086957$ **60.87 %**
Also, specificity is $\frac{16}{147+16} = 0.09815$; **9.82 %**
On the contrary, sensitivity is $\frac{83}{83+7} = 0.922222$; **92.22 %**

```
# create train data
train_x_bt <- cbind(bt_new$rad, bt_new$tax, bt_new$lstat, bt_new$age, bt_new$indus,
    bt_new$nox)[train, ]
# create test data
test_x_bt <- cbind(bt_new$rad, bt_new$tax, bt_new$lstat, bt_new$age, bt_new$indus,
    bt_new$nox)[test, ]
# vector containing the class labels for the training observations
train_crim_test <- crim_test[train]
```

```r
# K = 1 predictions data in the knn
knn_pred_bt = knn(train_x_bt, test_x_bt, train_crim_test, k = 1)
# confusing matrix
table(knn_pred_bt, train_crim_test)
```

```
##            train_crim_test
## knn_pred_bt   0   1
##           0  33 156
##           1  57   7
```

```r
# fraction of days for which the prediction was correct
mean(knn_pred_bt == train_crim_test)
```

```
## [1] 0.1581028
```

```r
# training error rate
mean(knn_pred_bt != train_crim_test)
```

```
## [1] 0.8418972
```

Percentage of current KNN [k = 1] predictions
Training correct prediction:
$\frac{(33+7)}{(33+156+57+7)} = 0.1581028$ **15.81 %**
Training error rate:
$1 - 0.1581028 = 0.8418972$ **84.19 %**
Also, specificity is $\frac{7}{156+7} = 0.04294$; **4.29 %**
On the contrary, sensitivity is $\frac{33}{33+57} = 0.3666$; **36.67 %**

```r
# K = 3 predictions data in the knn
knn_pred_bt = knn(train_x_bt, test_x_bt, train_crim_test, k = 3)
# confusing matrix
table(knn_pred_bt, train_crim_test)
```

```
##            train_crim_test
## knn_pred_bt   0   1
##           0  27  18
##           1  63 145
```

```r
# fraction of days for which the prediction was correct
mean(knn_pred_bt == train_crim_test)
```

```
## [1] 0.6798419
```

```r
# training error rate
mean(knn_pred_bt != train_crim_test)
```

```
## [1] 0.3201581
```

Percentage of current KNN [k = 3] predictions
Training correct prediction:
$\frac{(27+145)}{(27+18+63+145)} = 0.6798419$ **67.98 %**
Training error rate:
$1 - 0.6798419 = 0.3201581$ **32.02 %**
Also, specificity is $\frac{145}{18+145} = 0.88957$; **88.96 %**
On the contrary, sensitivity is $\frac{27}{27+63} = 0.3$; **30.00 %**

```r
# K = 7 predictions data in the knn
knn_pred_bt = knn(train_x_bt, test_x_bt, train_crim_test, k = 7)
# confusing matrix
table(knn_pred_bt, train_crim_test)
```

```
##            train_crim_test
## knn_pred_bt   0   1
##           0  44  16
##           1  46 147
```

```r
# fraction of days for which the prediction was correct
mean(knn_pred_bt == train_crim_test)
```

```
## [1] 0.7549407
```

```r
# training error rate
mean(knn_pred_bt != train_crim_test)
```

```
## [1] 0.2450593
```

Percentage of current KNN [k = 7] predictions
Training correct prediction:
$\frac{(44+147)}{(44+16+46+147)} = 0.7549407$ **75.49 %**
Training error rate:
$1 - 0.7549407 = 0.2450593$ **24.51 %**
Also, specificity is $\frac{147}{16+147} = 0.90184$; **90.18 %**
On the contrary, sensitivity is $\frac{44}{44+46} = 0.4888$; **48.89 %**

```r
# K = 9 predictions data in the knn
knn_pred_bt = knn(train_x_bt, test_x_bt, train_crim_test, k = 9)
# confusing matrix
table(knn_pred_bt, train_crim_test)
```

```
##            train_crim_test
## knn_pred_bt   0   1
##           0  41  13
##           1  49 150
```

```r
# fraction of days for which the prediction was correct
mean(knn_pred_bt == train_crim_test)
```

```
## [1] 0.7549407
```

```r
# training error rate
mean(knn_pred_bt != train_crim_test)
```

```
## [1] 0.2450593
```

Percentage of current KNN [k = 9] predictions
Training correct prediction:
$\frac{(41+150)}{(41+13+49+150)} = 0.7549407$ **75.49 %**
Training error rate:
$1 - 0.7549407 = 0.2450593$ **24.51 %**
Also, specificity is $\frac{150}{13+150} = 0.92024$; **92.02 %**
On the contrary, sensitivity is $\frac{41}{41+49} = 0.45555$; **45.56 %**

```r
# fit linear regression
lm.fit.bt <- lm(crime_rate ~ rad + tax + lstat + age + indus + nox, data = bt_new,
    subset = train)
# summary linear regression model
summary(lm.fit.bt)
```

```
##
## Call:
## lm(formula = crime_rate ~ rad + tax + lstat + age + indus + nox,
##     data = bt_new, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71124 -0.22954 -0.06565  0.26336  0.98823
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.1573727  0.1491690  -7.759 2.29e-13 ***
## rad          0.0587780  0.0148989   3.945 0.000104 ***
## tax          0.0003235  0.0003925   0.824 0.410525
## lstat       -0.0035085  0.0044250  -0.793 0.428604
## age          0.0033628  0.0010882   3.090 0.002229 **
## indus        0.0082280  0.0052923   1.555 0.121302
## nox          1.6932393  0.3376640   5.015 1.02e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3571 on 246 degrees of freedom
## Multiple R-squared:  0.4589, Adjusted R-squared:  0.4457
## F-statistic: 34.77 on 6 and 246 DF,  p-value: < 2.2e-16
```

This summary show that, the null hypothesis for **tax**,**lstat**, and **indus** cannot be rejected.

```r
# predictions for all values in the training set
prob_bt_line <- predict(lm.fit.bt, bt_test, type = "response")
# create all values '0' in pred_dt_line
pred_bt_line <- rep(0, length(prob_bt_line))
# all of the elements for which the predicted probability of a market increase
# exceeds 0.5 in pred_dt_line
```

```r
pred_bt_line[prob_bt_line > 0.5] <- 1
# confusion matrix
table(pred_bt_line, crim_test)
```

```
##             crim_test
## pred_bt_line   0   1
##            0  81  19
##            1   9 144
```

```r
# fraction of days for which the prediction was correct
mean(pred_bt_line == crim_test)
```

```
## [1] 0.8893281
```

```r
# training error rate
mean(pred_bt_line != crim_test)
```

```
## [1] 0.1106719
```

Percentage of current linear regression predictions
Training correct prediction:
$\frac{(81+144)}{(81+19+9+144)} = 0.8893281$ **88.93 %**
Training error rate:
$1 - 0.8893281 = 0.1106719$ **11.07 %**
Also, specificity is $\frac{144}{19+144} = 0.88343$; **88.34 %**
On the contrary, sensitivity is $\frac{81}{81+9} = 0.90$; **90.00 %**

As a result, each accuracy of predictions:
Logistic regression: **88.93 %**
LDA: **88.94 %**
QDA: **39.13 %**
KNN (k=1): **15.81 %**
KNN (k=3): **67.98 %**
KNN (k=7): **75.49 %**
KNN (k=9): **75.49 %**
Linear regression: **88.93 %**

As such, **LDA**, **logistic regression**, and **linear regression** are judged to be highly accurate. In addition, as a result of examining through the logistic regression model, it can be seen that only the **indus, nox, tax, and rad** variables are statistically significant variables. It can be seen that the **accuracy of the nearest neighbor K with K=1** is 15.81%, which is not effective when classifying the model, and that the **error rate improves as K increases**.

Q8. Consider the data set provided with this homework assignment. Implement LDA and QDA classifiers on this data and compare the two classifiers using a ROC curve.

a. Import the data set in R.

```r
# dataset into dat
hw3 <- read.csv("Hw3data.csv")
# show first 6 data
head(hw3)
```

```
##   response predictor.1 predictor.2 predictor.3 predictor.4 predictor.5
## 1        0   1.2398256   1.5653305   1.5829808    1.681949   0.5853821
## 2        0   0.4658289   0.3528325   0.1112685    0.286668  -0.7642320
## 3        0   2.2148792   2.2409103   1.8192105    1.245698   0.7272527
## 4        0   1.5625651   2.0400575   1.7234388    1.434978   0.6785989
## 5        0   1.4291893   1.2423552   0.9397363    1.217380   0.3375855
## 6        0   1.2749576   1.7065999   1.2813213    1.048333  -0.1789173
```

```r
# fit LDA
lda.fit.hw <- lda(response ~ ., data = hw3)
# fit predict
lda.pred.hw <- predict(lda.fit.hw)
# contains LDA's predictions about the movement of the hw3
pred_hw.class = lda.pred.hw$class
# confusion matrix
table(pred_hw.class, hw3$response)
```

```
##
## pred_hw.class  0  1
##             0 29 25
##             1 21 25
```

```r
# fraction of days for which the prediction was correct
mean(pred_hw.class == hw3$response)
```

```
## [1] 0.54
```

```r
# training error rate
mean(pred_hw.class != hw3$response)
```

```
## [1] 0.46
```

Percentage of current LDA predictions
Training correct prediction:
$\frac{(29+25)}{(29+25+21+25)} = 0.54$ **54.00 %**
Training error rate:
$1 - 0.54 = 0.46$ **46.00 %**
Also, specificity is $\frac{25}{25+25} = 0.5$; **50.00 %**
On the contrary, sensitivity is $\frac{29}{29+21} = 0.58$; **58.00 %**

```r
# fit QDA
qda.fit.hw <- qda(response ~ ., data = hw3)
# fit predict
qda.pred.hw <- predict(qda.fit.hw)
# contains QDA's predictions about the movement of the hw3
```

```r
qda_pred_hw.class = qda.pred.hw$class
# confusion matrix
table(qda_pred_hw.class, hw3$response)
```

```
##
## qda_pred_hw.class  0  1
##                 0 47  4
##                 1  3 46
```

```r
# fraction of days for which the prediction was correct
mean(qda_pred_hw.class == hw3$response)
```

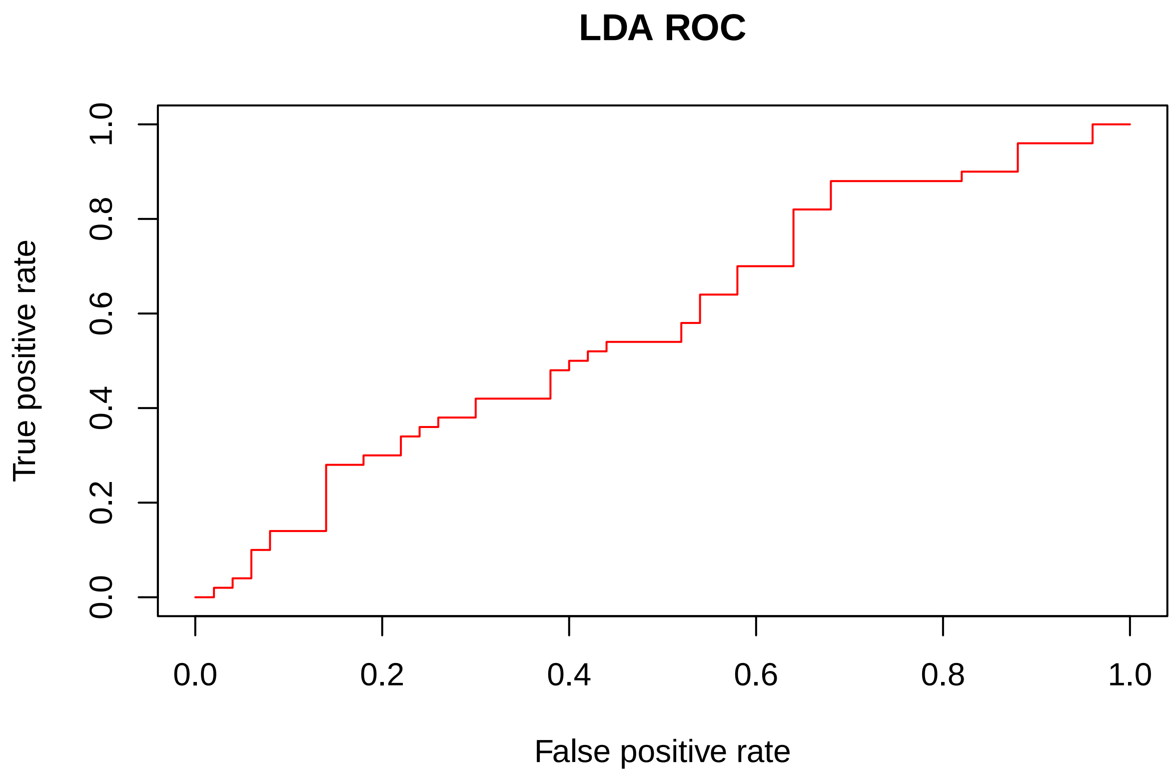
```
## [1] 0.93
```

```r
# training error rate
mean(qda_pred_hw.class != hw3$response)
```

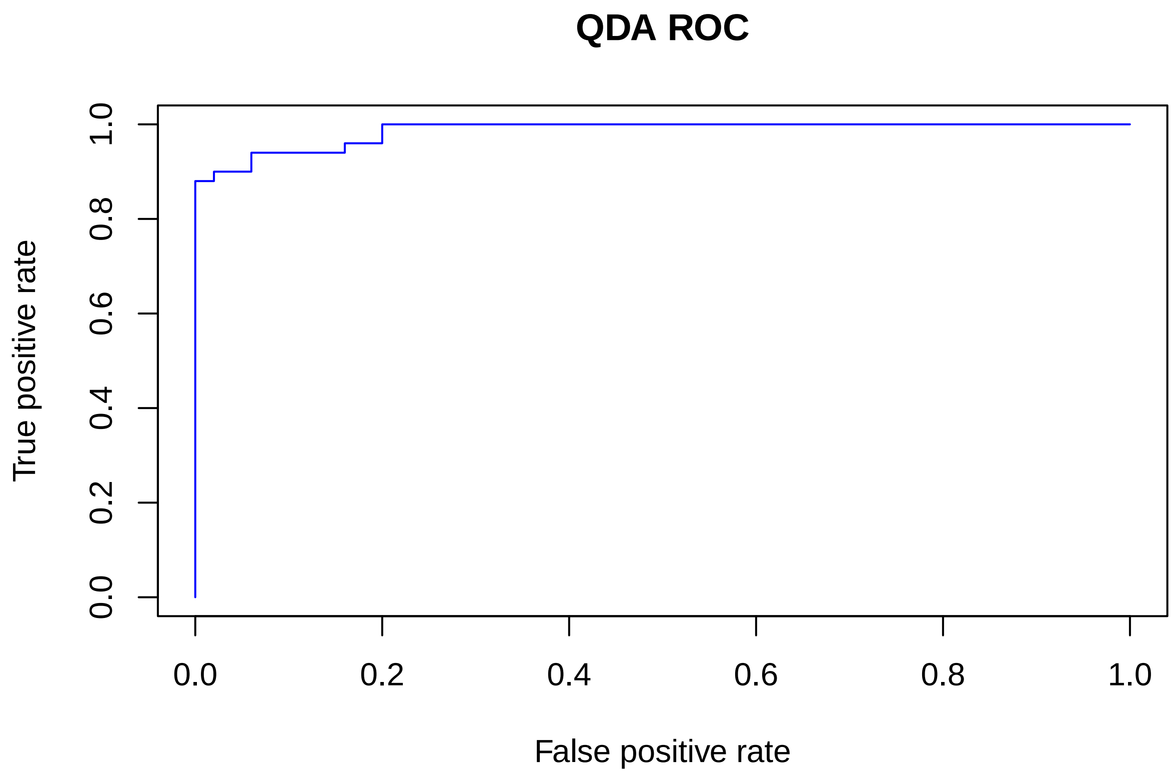
```
## [1] 0.07
```

Percentage of current RDA predictions
Training correct prediction:
$\frac{(47+46)}{(47+4+3+46)} = 0.93$ **93.00 %**
Training error rate:
$1 - 0.93 = 0.07$ **70.00 %**
Also, specificity is $\frac{46}{4+46} = 0.92$; **92.00 %**
On the contrary, sensitivity is $\frac{47}{47+3} = 0.94$; **94.00 %**

```r
# the classification rate is calculated by comparing the calculated probability
# p with the actual test data
preda <- prediction(lda.pred.hw$posterior[, 2], hw3$response)
# calculate the sensitivity and 1-specificity to draw the ROC curve
oerf <- performance(preda, measure = "tpr", x.measure = "fpr")
# show LDA ROC plot
plot(oerf, col = "red", main = "LDA ROC")
```




```r
# the classification rate is calculated by comparing the calculated probability
# p with the actual test data
predab <- prediction(qda.pred.hw$posterior[, 2], hw3$response)
# calculate the sensitivity and 1-specificity to draw the ROC curve
oerff <- performance(predab, measure = "tpr", x.measure = "fpr")
# show QDA ROC plot
plot(oerff, main = "QDA ROC", col = "blue")
```



For a model with a perfect ROC curve graph, TPR is 1 and FPR is 0 at all data points. Therefore, the **LDA classifier is better** in that the *LDA ROC curve TPR is closer to 1 more than the QDA ROC curve TPR and the LDA ROC curve FPR is closer to 0 more than the QDA ROC curve FPR*. Also, when comparing the test error rates of the two classifiers:
LDA: **46.00 %**
QDA: **70.00 %**

Like this, the **LDA classifier is better than the RDA classifier**.