HW5 R Markdown

Santanu Mukherjee

11/02/2021

R Markdown

Chapter 4 page 168:

Q1

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.

Answer 1

So, in (4.2), We have

$p(X)$ = $e^{\beta_0+\beta_1 X}$ / 1 + $e^{\beta_0+\beta_1 X}$

This implies

$e^{\beta_0+\beta_1 X}$ $(1-p(X))$ = $p(X)$

which is equivalent to

$p(X)$ / $(1-p(X))$ = $e^{\beta_0+\beta_1 X}$

Q3

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 ~ $N(\mu_k, \sigma_k^2)$. 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.

Answer 3

So, from (4.11), we get

$f_k(x)$ = $\frac{1}{(\sqrt{(2\pi)})\sigma_k}$ $exp(\frac{-(x-\mu_k)^2}{2\sigma_k^2})$

and, from (4.10), we get according to Bayes Theorem,

$p_k(X)$ = $\frac{\pi_k f_k(x)}{\sum_{l=1}^{k} \pi_l f_l(x)}$ (1)

So, to use the Bayes classifier, we have to find the class (k) such that the equation (1) is largest. As the log function is monotone increasing , it means that we can find k for which the below equation is largest

$log\pi_k - \frac{1}{2\sigma^2}(x-\mu_k)^2$

$= log\pi_k - \frac{1}{2\sigma^2}x^2 + \frac{\mu_k}{\sigma^2}x - \frac{\mu_k^2}{2\sigma^2} - log\sigma_k$

Now, the above equation is not *linear in* x. Furthermore I can argue that the highest order of x is 2 , which means it is **quadratic**.

Q4

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.

Q4a

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?

Answer 4 (a)

So, based on the question, we can say that if $X\epsilon[0.05, 0.95]$, then we would use the observations that are in the interval $[X - 0.05, X + 0.05]$ which represents a length of 0.1 or in other words means within 10% of the observations.

Now, there are couple iof scenarios which is outside of the ranges mentioned above. They are:
If $X\epsilon[0, 0.05)$, training observations in the range $[0, 0.1]$ will be used
If $X\epsilon(0.95, 1]$, training observations in the range $[0.9, 1]$ will be used

Again, as X is evenly distributed $(0, 1)$, these cases will also use 10% of the observations.

So, across all cases we see that the fraction of the available observations will we use to make the prediction is **10%**.

Q4b

Now suppose that we have a set of observations, each with measurements on $p = 2$ features, $X1$ and $X2$. We assume that $(X1, X2)$ are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to predict a test observation's response using only observations that are within 10% of the range of $X1$ and within 10% of the range of $X2$ closest to that test observation. For instance, in order to predict the response for a test observation with $X1 = 0.6$ and $X2 = 0.35$, we will use observations in the range $[0.55, 0.65]$ for $X1$ and in the range $[0.3, 0.4]$ for $X2$. On average, what fraction of the available observations will we use to make the prediction?

Answer 4 (b)

This question is the same as part(a), but the difference is that part(a) talked about one dimension , now it is about 2 dimensions.

So, here I will apply similar logic like part (a) above and say that 10% of observations will satisfy the *first criteria*, and 10% will satisfy the *second criteria*, but the question here is what fraction of the available observations will satisfy **both criteria**.

Now, we need to assume here that $X1$ and $X2$ are independent, and then we can multiply the probabilities of these two events, and so the fraction of observations that will be available to make the prediction is 0.1^2 = 0.01 = 1%.

Q4c

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?

Answer 4 (c)

So, based on the logic used in part(b) above, it can be said that the fraction of observations within 10% of all $p = 100$ that would be used to make a prediction would be 0.1^{100}.

Q4d

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.

Answer 4 (d)

So, for parts (a)-(c) when we are saying *'near'*, it means **'within 10% of the range'**. So, as dimensionality increases, the probability that there will be training observations *'near'* the test observation X across all p dimensions approaches zero:

$\lim_{p \to \infty} (0.1)^p$ = 0

The meaning of the above equation is that in datasets where p is large, the K nearest neighbors will not be very close in reality, because there would not be any training observations that would be *'near'* across all p dimensions.

Q4e

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.

Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$, a hypercube is simply a line segment, when $p = 2$ it is a square, and when $p = 100$ it is a 100-dimensional cube.

Answer 4 (e)

So, here we need to find the length of each side of the hypercube:

For p = 1, the length of the line is 0.1

For p = 2, the length of the side of the square is $(0.1)^{\frac{1}{2}}$.

For p = 100, the length of the side of the hypercube is $(0.1)^{\frac{1}{100}}$.

Q6

Suppose we collect data for a group of students in a statistics class with variables:

$X1$ = hours studied
$X2$ = undergrad GPA
Y = receive an A.

We fit a logistic regression and produce estimated coefficients:

$\hat{\beta}_0$ = -6
$\hat{\beta}_1$ = 0.05
$\hat{\beta}_2$ = 1

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

Answer 6 (a)

Similar to the equation I have used earlier , we can write

$p(X)$ = $e^{\hat{\beta}_0+\hat{\beta}_1 X_1+\hat{\beta}_2 X_2}$ / 1 + $e^{\hat{\beta}_0+\hat{\beta}_1 X_1+\hat{\beta}_2 X_2}$

$p(X)$ =

$$\frac{e^{\hat{\beta}_0+\hat{\beta}_1 X_1+\hat{\beta}_2 X_2}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X_1+\hat{\beta}_2 X_2}}$$

Now inputting the given values , we get

$p(X)$ =

$$\frac{e^{-6+(0.05)(40)+(1)(3.5)}}{1+e^{-6+(0.05)(40)+(1)(3.5)}}$$

$p(X)$ =

$$\frac{e^{-0.5}}{1+e^{-0.5}}$$

```
p1 <- exp(-0.5)
p2 <- 1 + p1

p = p1/p2
print(paste("The probability that a student who studies for 40 hours and has an undergad GPA of
 3.5 gets A in class is:",round(p,3)))
```

```
## [1] "The probability that a student who studies for 40 hours and has an undergad GPA of 3.5 g
ets A in class is: 0.378"
```

(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?

Answer 6 (b)

So, using the equation in part (a) above and putting X_1 as the variable for "number of hours required" and also putting 0.5 for $p(X)$, we get,

0.5 =

$$\frac{e^{-6+(0.05)(X_1)+(1)(3.5)}}{1+e^{-6+(0.05)(X_1)+(1)(3.5)}}$$

which means

0.5 =

$$\frac{e^{0.05X_1-2.5}}{1+e^{0.05X_1-2.5}}$$

which implies

$e^{0.05X_1-2.5}$ = 1

X_1 = $\frac{log(1)+2.5}{0.05}$

```
p1 <- log(1)
p2 <- (p1 +2.5)/0.05

print(paste("The number of hours required for a student to have a 50% chance of getting A in cla
ss is:",p2))
```

```
## [1] "The number of hours required for a student to have a 50% chance of getting A in class i
s: 50"
```

Q8

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?

Answer 8

First of all, lets discuss the two different classification methods.

The training error for KNN as the error that occurs when the training dataset is used.

So, when we run K nearest neighbor with K = 1, this means that when KNN makes a prediction on an observation, it will look for the single closest observation available in the training data (which will be itself). It will then assign that training observations response value as the prediction for the test observation.

This will always have zero error, irrespective of the dataset or whether classification/regression is being used.

This means then, if KNN (where K = 1) averages an 18% error across train & test, its training error will be 0, so its test error must be $2 * 18\%$ = 36%, which is worse than the 30% test error of logistic regression.

For this reason I would prefer to use logistic regression compared to the K nearest neighbor classifier.

So, I would prefer Logistic Regression.

Q10

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 1089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

Q10a

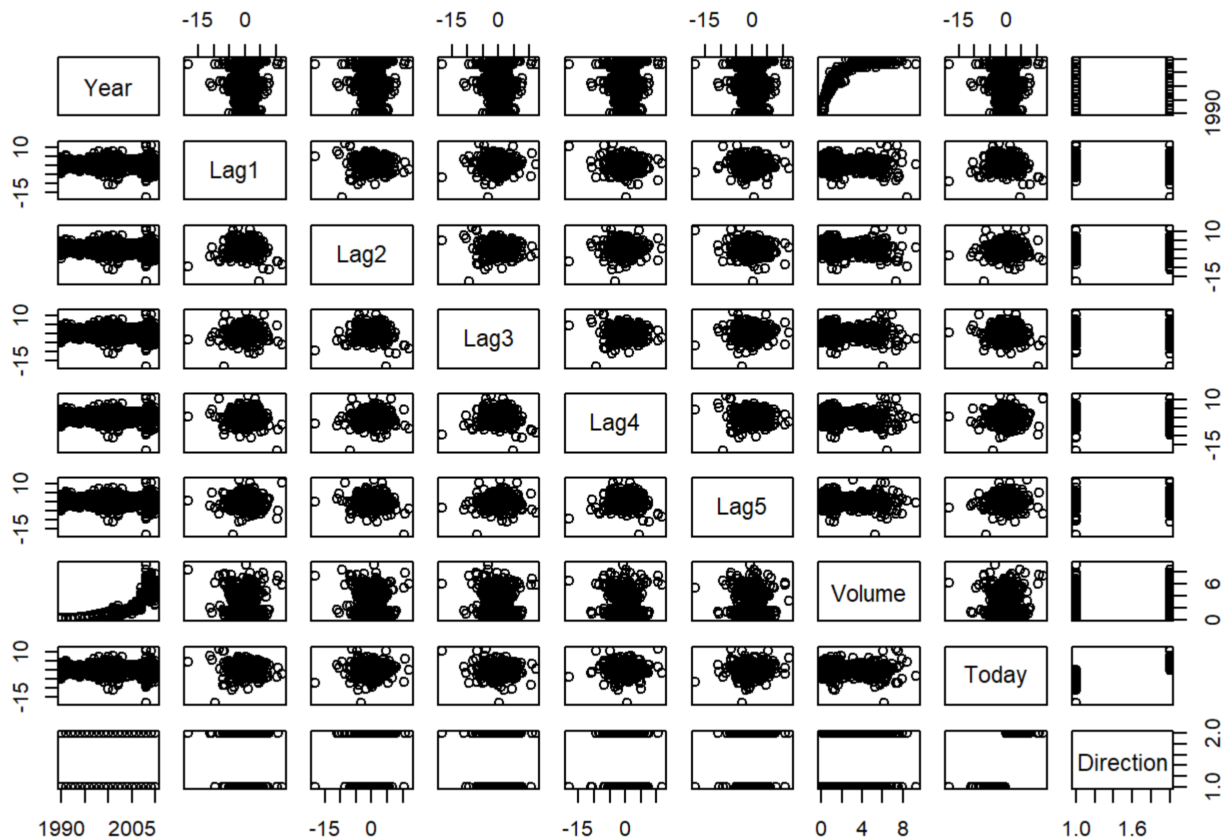
Produce some numerical and graphical summaries of the *“Weekly”* data. Do there appear to be any patterns ?

Answer 10a

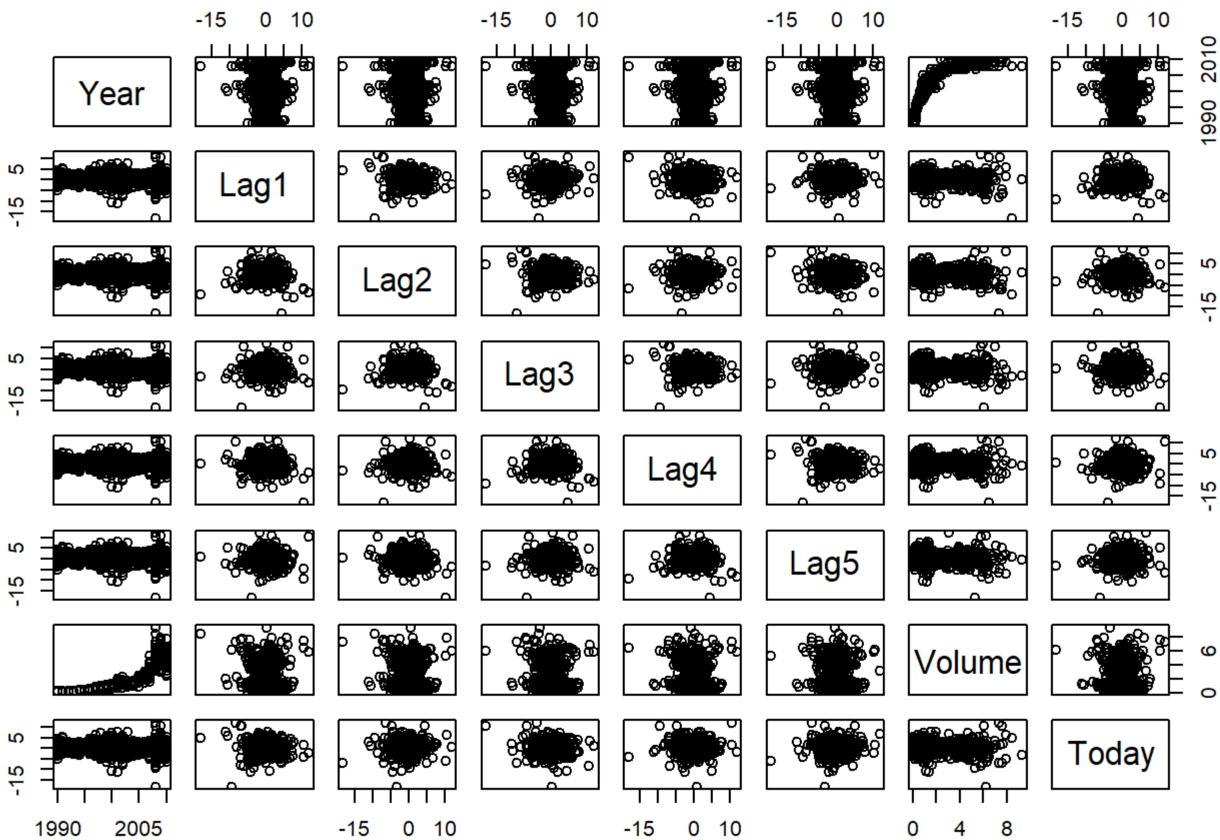
```
##Part (a) Weekly Data Summary
library(ISLR)
summary(Weekly)
```

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

```
pairs(Weekly)
```



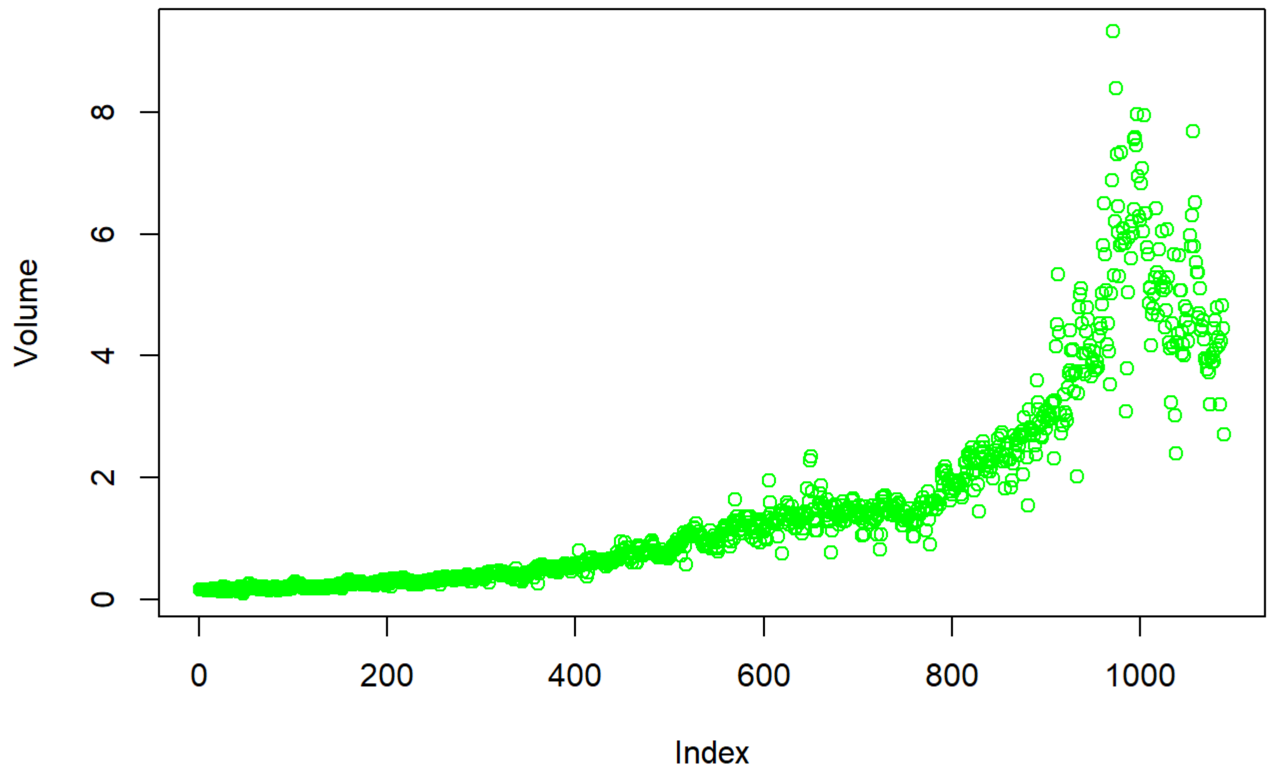

```
pairs(Weekly[,-9])
```




```
cor(Weekly[, -9])
```

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

```
attach(Weekly)
plot(Volume, col="green")
```



Step by Step Observations:
1. The *Summary* and subsequently the *pairs* showed that the variable "Direction" was insignificant.
2. So, then I got the correlation matrix with all variables except **Direction**.
3. The correlations between the **"lag"** variables and **Today* variable are close to zero.**
4.** The correlation between variables **"Year"** and **"Volume"** is the only significant one.
5. So, I have done plot "Volume", and I see that is increasing over time.

Q10b

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 ?

Answer 10b

```
##Part (b) Logistic Regression

log.reg <-glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Weekly,family=binomial)
summary(log.reg)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## 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
```

It seems that “Lag2” is the only predictor which is statistically significant at $\alpha = 0.05$ as its p-value is less than 0.05.

Q10c

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.

Answer 10c

```
##Part (c) Confusion Matrix

prob.log.reg <- predict(log.reg, type = "response")
pred.log.reg <- rep("Down", length(prob.log.reg))
pred.log.reg[prob.log.reg > 0.5] <- "Up"
table(pred.log.reg, Direction)
```

```
##             Direction
## pred.log.reg Down  Up
##         Down   54  48
##         Up    430 557
```

Based on the results of the table above, We may conclude that the percentage of correct predictions (Down * Down & Up *Up) on the training data is $(54 + 557)/1089$ which is equal to 56.11%. So, we can say that 43.89% is the training error rate.

If we look at the data from another angle , meaning we could also conclude that for the *weeks* when the market goes **Up**, the model is right 92.07% of the time $(557/(48 + 557))$.
Similarly, for the *weeks* when the market goes **Down**, the model is right only 11.16% of the time $(54/(54 + 430))$.

Q10d

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 $2009 to 2010$).

Answer 10d

```
##Part (d) Logistic regression with data from 2009-2010 and the only predictor being "Lag2"

train.data <- (Year < 2009)
Weekly.2009.2010 <- Weekly[!train.data, ]
Direction.2009.2010 <- Direction[!train.data]
log.reg.lag2 <- glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train.data)
summary(log.reg.lag2)
```

```
## 
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = Weekly, 
##     subset = train.data)
## 
## Deviance Residuals: 
##    Min      1Q  Median      3Q     Max  
## -1.536  -1.264   1.021   1.091   1.368  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)   
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 * 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
## 
## Number of Fisher Scoring iterations: 4
```

```
##Part (d) Confusion Matrix

prob2.log.reg <- predict(log.reg.lag2, Weekly.2009.2010, type = "response")
pred2.log.reg <- rep("Down", length(prob2.log.reg))
pred2.log.reg[prob2.log.reg > 0.5] <- "Up"
table(pred2.log.reg, Direction.2009.2010)
```

```
##              Direction.2009.2010
## pred2.log.reg Down Up
##          Down    9  5
##          Up     34 56
```

Based on the results of the table above, we can conclude that the percentage of correct predictions on the test data is $(9 + 56)/104$ (Down * Down & Up * Up) which is equal to 62.5%. So, we can say that 37.5% is the test error rate.

If we look at the data from another angle , meaning we could also conclude that for the *weeks* when the market goes **Up**, the model is right 91.80% of the time $(56/(56 + 5))$.
Similarly, for the *weeks* when the market goes **Down**, the model is right only 20.93% of the time $(9/(9 + 34))$.

Q10e

Repeat (d) using LDA.

Answer 10e

```
##Part (e) 1st part - Repeating part (d) using LDA

library(MASS)
fit.lda <- lda(Direction ~ Lag2, data = Weekly, subset = train.data)
fit.lda
```

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

```
##Part (e) 2nd part - Repeating part (d) using LDA

pred.e.lda <- predict(fit.lda, Weekly.2009.2010)
table(pred.e.lda$class, Direction.2009.2010)
```

```
##       Direction.2009.2010
##        Down Up
##   Down    9  5
##   Up     34 56
```

Based on the results, we conclude that the output is exactly the same as part (d), which means in this case, the **Logistic Regression** and **LDA** has yielded the same result.

Q10f

Repeat (d) using QDA.

Answer 10f

```
##Part (f) 1st part - Repeating part (d) using QDA

library(MASS)
fit.qda <- qda(Direction ~ Lag2, data = Weekly, subset = train.data)
fit.qda
```

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

```
##Part (f) 2nd part - Repeating part (d) using QDA

pred.f.qda <- predict(fit.qda, Weekly.2009.2010)
table(pred.f.qda$class, Direction.2009.2010)
```

```
##       Direction.2009.2010
##        Down Up
##   Down    0  0
##   Up     43 61
```

Based on the results of the table above, we can conclude that the percentage of correct predictions on the test data is $(0 + 61)/104$ (Down * Down & Up * Up) which is equal to 58.65%. So, we can say that 41.35% is the test error rate.

If we look at the data from another angle , meaning we could also conclude that for the *weeks* when the market goes **Up**, the model is right 100% of the time $(61/(61 + 0))$.
Similarly, for the *weeks* when the market goes **Down**, the model is right 0% of the time $(0/(0 + 43))$.

So, here in QDA, the model chooses **Up** the entire time.

Q10g

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

Answer 10g

```
##Part (g) 1st part - Repeating part (d) using KNN with K = 1

library(class)
train.lag2 <- as.matrix(Lag2[train.data])
test.lag2 <- as.matrix(Lag2[!train.data])
train.Direction <- Direction[train.data]
```

```
##Part (g) 2nd part - Repeating part (d) using KNN with K = 1

set.seed(1)
pred.g.knn <- knn(train.lag2, test.lag2, train.Direction, k = 1)
table(pred.g.knn, Direction.2009.2010)
```

```
##            Direction.2009.2010
## pred.g.knn Down Up
##       Down   21 30
##       Up     22 31
```

Based on the results of the table above, we can conclude that the percentage of correct predictions on the test data is $(21 + 31)/104$ (Down * Down & Up * Up) which is equal to 50%. So, we can say that 50% is the test error rate.

If we look at the data from another angle , meaning we could also conclude that for the *weeks* when the market goes **Up**, the model is right 50.82% of the time $(30/(30 + 31))$.
Similarly, for the *weeks* when the market goes **Down**, the model is right only 48.84% of the time $(21/(21 + 22))$.

Q10h

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

Answer 10h

If we just want to compare the test error rates, we see that $Logistic\ Regression$ and LDA have the minimum error rates and so are the best, followed by QDA and KNN.

Q10i

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.

Answer 10i

$Logistic\ Regression$

```
##Logistic Regression with predictors Lag2:Lag1

fit.10.11 <- glm(Direction ~ Lag2:Lag1, data = Weekly, family=binomial, subset = train.data)
prob.10.11 <- predict(fit.10.11, Weekly.2009.2010, type = "response")
pred.10.11 <- rep("Down", length(prob.10.11))
pred.10.11[prob.10.11 > 0.5] <- "Up"
table(pred.10.11, Direction.2009.2010)
```

```
##           Direction.2009.2010
## pred.10.11 Down Up
##       Down    1  1
##       Up     42 60
```

```
##Part (g) 2nd part - Repeating part (d) using KNN with K = 1

mean(pred.10.11 == Direction.2009.2010)
```

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

LDA

```
# LDA with Lag2 interaction with Lag1

fit.lda.10.21 <- lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train.data)
pred.lda.10.21 <- predict(fit.lda.10.21, Weekly.2009.2010)
mean(pred.lda.10.21$class == Direction.2009.2010)
```

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

QDA

```
# QDA with sqrt(abs(Lag2))

fit.qda.10.21 <- qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = train.data)
pred.qda.10.21 <- predict(fit.qda.10.21, Weekly.2009.2010)
mean(pred.qda.10.21$class == Direction.2009.2010)
```

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

$KNN\ k = 10$

```
# KNN k =10
pred.knn.10.41 <- knn(train.lag2, test.lag2, train.Direction, k = 10)
table(pred.knn.10.41, Direction.2009.2010)
```

```
##                Direction.2009.2010
## pred.knn.10.41 Down Up
##           Down   17 18
##           Up     26 43
```

```
mean(pred.knn.10.41 == Direction.2009.2010)
```

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

$KNN\, k = 100$

```
# KNN k =100
pred.knn.10.51 <- knn(train.lag2, test.lag2, train.Direction, k = 100)
table(pred.knn.10.51, Direction.2009.2010)
```

```
##                Direction.2009.2010
## pred.knn.10.51 Down Up
##           Down    9 12
##           Up     34 49
```

```
mean(pred.knn.10.51 == Direction.2009.2010)
```

```
## [1] 0.5576923
```

After running multiple different combinations, the results show that the original $Logistic\ Regression$ and LDA have the best performance in terms of test error rates.

Problem NOT from the text book

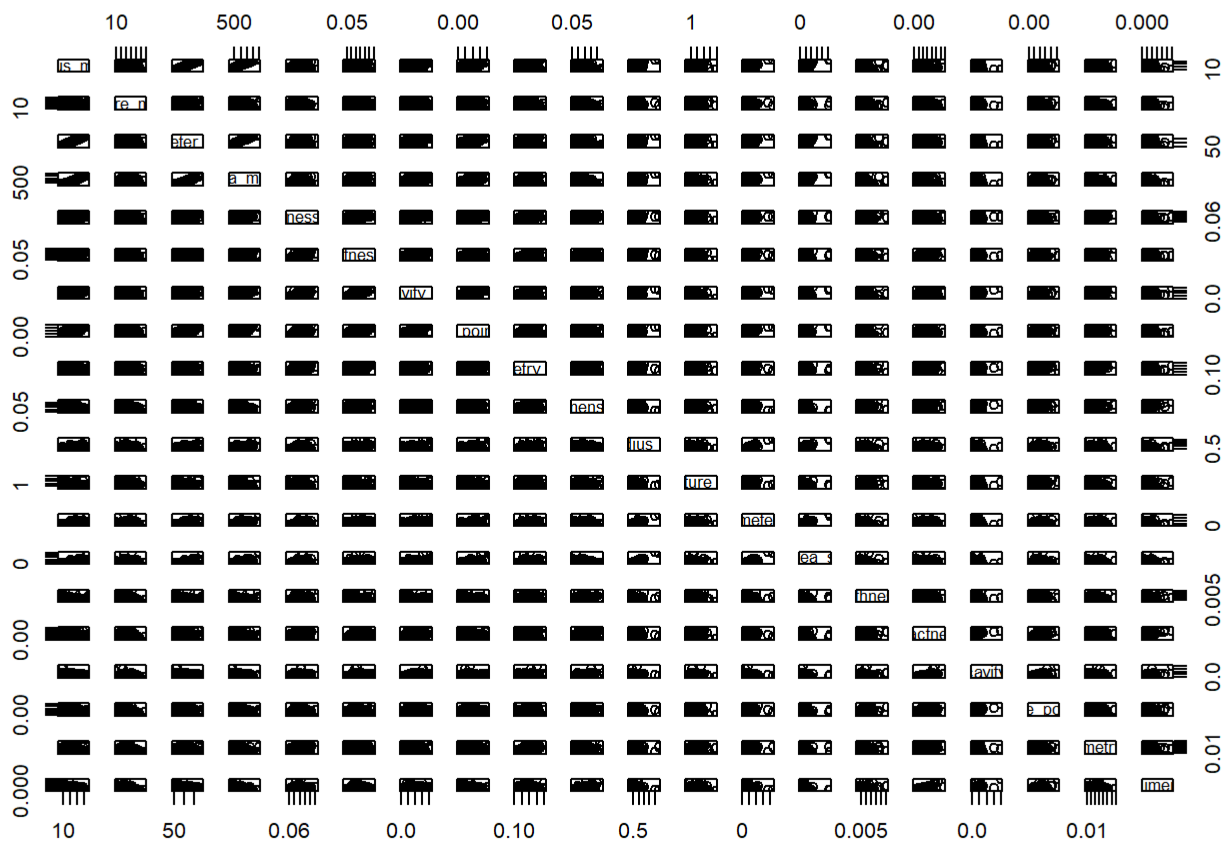
```
library(RCurl)
fileURL = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdb
c.data"
BC_data = read.csv(fileURL, header = FALSE, sep = ",")
names(BC_data)[1] <- 'id_number'
names(BC_data)[2] <- 'diagnosis'
names(BC_data)[3] <- 'radius_mean'
names(BC_data)[4] <- 'texture_mean'
names(BC_data)[5] <- 'perimeter_mean'
names(BC_data)[6] <- 'area_mean'
names(BC_data)[7] <- 'smoothness_mean'
names(BC_data)[8] <- 'compactness_mean'
names(BC_data)[9] <- 'concavity_mean'
names(BC_data)[10] <- 'concave_points_mean'
names(BC_data)[11] <- 'symmetry_mean'
names(BC_data)[12] <- 'fractal_dimension_mean'
names(BC_data)[13] <- 'radius_se'
names(BC_data)[14] <- 'texture_se'
names(BC_data)[15] <- 'perimeter_se'
names(BC_data)[16] <- 'area_se'
names(BC_data)[17] <- 'smoothness_se'
names(BC_data)[18] <- 'compactness_se'
names(BC_data)[19] <- 'concavity_se'
names(BC_data)[20] <- 'concave_points_se'
names(BC_data)[21] <- 'symmetry_se'
names(BC_data)[22] <- 'fractal_dimension_se'
names(BC_data)[23] <- 'radius_worst'
names(BC_data)[24] <- 'texture_worst'
names(BC_data)[25] <- 'perimeter_worst'
names(BC_data)[26] <- 'area_worst'
names(BC_data)[27] <- 'smoothness_worst'
names(BC_data)[28] <- 'compactness_worst'
names(BC_data)[29] <- 'concavity_worst'
names(BC_data)[30] <- 'concave_points_worst'
names(BC_data)[31] <- 'symmetry_worst'
names(BC_data)[32] <- 'fractal_dimension_worst'

BC_data_final <- BC_data[,-c(23:32)]

BC_data_final$id_number <- NULL

BC_data <- BC_data_final
```

Q 1 a & b

Do certain exploratory analysis first. (a) Check the scatterplots as well as correlations between the predictors. It would be reasonable only to compare those predictors with their own metrics, i.e., means, standard errors, and worst cases.
(b) Check whether there are strong multicollinearity effects among the predictors.

Answer 1 a & b

```
BC_data_relation <- (BC_data[,-1])

pairs(BC_data_relation)
```



```
round(cor(BC_data_relation),3)
```

```
##                         radius_mean texture_mean perimeter_mean area_mean
## radius_mean                   1.000        0.324          0.998     0.987
## texture_mean                  0.324        1.000          0.330     0.321
## perimeter_mean                0.998        0.330          1.000     0.987
## area_mean                     0.987        0.321          0.987     1.000
## smoothness_mean               0.171       -0.023          0.207     0.177
## compactness_mean              0.506        0.237          0.557     0.499
## concavity_mean                0.677        0.302          0.716     0.686
## concave_points_mean           0.823        0.293          0.851     0.823
## symmetry_mean                 0.148        0.071          0.183     0.151
## fractal_dimension_mean       -0.312       -0.076         -0.261    -0.283
## radius_se                     0.679        0.276          0.692     0.733
## texture_se                   -0.097        0.386         -0.087    -0.066
## perimeter_se                  0.674        0.282          0.693     0.727
## area_se                       0.736        0.260          0.745     0.800
## smoothness_se                -0.223        0.007         -0.203    -0.167
## compactness_se                0.206        0.192          0.251     0.213
## concavity_se                  0.194        0.143          0.228     0.208
## concave_points_se             0.376        0.164          0.407     0.372
## symmetry_se                  -0.104        0.009         -0.082    -0.072
## fractal_dimension_se         -0.043        0.054         -0.006    -0.020
##                         smoothness_mean compactness_mean concavity_mean
## radius_mean                       0.171            0.506          0.677
## texture_mean                     -0.023            0.237          0.302
## perimeter_mean                    0.207            0.557          0.716
## area_mean                         0.177            0.499          0.686
## smoothness_mean                   1.000            0.659          0.522
## compactness_mean                  0.659            1.000          0.883
## concavity_mean                    0.522            0.883          1.000
## concave_points_mean               0.554            0.831          0.921
## symmetry_mean                     0.558            0.603          0.501
## fractal_dimension_mean            0.585            0.565          0.337
## radius_se                         0.301            0.497          0.632
## texture_se                        0.068            0.046          0.076
## perimeter_se                      0.296            0.549          0.660
## area_se                           0.247            0.456          0.617
## smoothness_se                     0.332            0.135          0.099
## compactness_se                    0.319            0.739          0.670
## concavity_se                      0.248            0.571          0.691
## concave_points_se                 0.381            0.642          0.683
## symmetry_se                       0.201            0.230          0.178
## fractal_dimension_se              0.284            0.507          0.449
##                         concave_points_mean symmetry_mean fractal_dimension_mean
## radius_mean                           0.823         0.148                 -0.312
## texture_mean                          0.293         0.071                 -0.076
## perimeter_mean                        0.851         0.183                 -0.261
## area_mean                             0.823         0.151                 -0.283
## smoothness_mean                       0.554         0.558                  0.585
## compactness_mean                      0.831         0.603                  0.565
## concavity_mean                        0.921         0.501                  0.337
## concave_points_mean                   1.000         0.462                  0.167
## symmetry_mean                         0.462         1.000                  0.480
## fractal_dimension_mean                0.167         0.480                  1.000
```

```
## radius_se                 0.698      0.303                 0.000
## texture_se                0.021      0.128                 0.164
## perimeter_se              0.711      0.314                 0.040
## area_se                   0.690      0.224                -0.090
## smoothness_se             0.028      0.187                 0.402
## compactness_se            0.490      0.422                 0.560
## concavity_se              0.439      0.343                 0.447
## concave_points_se         0.616      0.393                 0.341
## symmetry_se               0.095      0.449                 0.345
## fractal_dimension_se      0.258      0.332                 0.688
##                       radius_se texture_se perimeter_se area_se smoothness_se
## radius_mean               0.679     -0.097        0.674   0.736        -0.223
## texture_mean              0.276      0.386        0.282   0.260         0.007
## perimeter_mean            0.692     -0.087        0.693   0.745        -0.203
## area_mean                 0.733     -0.066        0.727   0.800        -0.167
## smoothness_mean           0.301      0.068        0.296   0.247         0.332
## compactness_mean          0.497      0.046        0.549   0.456         0.135
## concavity_mean            0.632      0.076        0.660   0.617         0.099
## concave_points_mean       0.698      0.021        0.711   0.690         0.028
## symmetry_mean             0.303      0.128        0.314   0.224         0.187
## fractal_dimension_mean    0.000      0.164        0.040  -0.090         0.402
## radius_se                 1.000      0.213        0.973   0.952         0.165
## texture_se                0.213      1.000        0.223   0.112         0.397
## perimeter_se              0.973      0.223        1.000   0.938         0.151
## area_se                   0.952      0.112        0.938   1.000         0.075
## smoothness_se             0.165      0.397        0.151   0.075         1.000
## compactness_se            0.356      0.232        0.416   0.285         0.337
## concavity_se              0.332      0.195        0.362   0.271         0.269
## concave_points_se         0.513      0.230        0.556   0.416         0.328
## symmetry_se               0.241      0.412        0.266   0.134         0.414
## fractal_dimension_se      0.228      0.280        0.244   0.127         0.427
##                       compactness_se concavity_se concave_points_se
## radius_mean                    0.206        0.194             0.376
## texture_mean                   0.192        0.143             0.164
## perimeter_mean                 0.251        0.228             0.407
## area_mean                      0.213        0.208             0.372
## smoothness_mean                0.319        0.248             0.381
## compactness_mean               0.739        0.571             0.642
## concavity_mean                 0.670        0.691             0.683
## concave_points_mean            0.490        0.439             0.616
## symmetry_mean                  0.422        0.343             0.393
## fractal_dimension_mean         0.560        0.447             0.341
## radius_se                      0.356        0.332             0.513
## texture_se                     0.232        0.195             0.230
## perimeter_se                   0.416        0.362             0.556
## area_se                        0.285        0.271             0.416
## smoothness_se                  0.337        0.269             0.328
## compactness_se                 1.000        0.801             0.744
## concavity_se                   0.801        1.000             0.772
## concave_points_se              0.744        0.772             1.000
## symmetry_se                    0.395        0.309             0.313
## fractal_dimension_se           0.803        0.727             0.611
##                       symmetry_se fractal_dimension_se
## radius_mean                -0.104               -0.043
```

```
## texture_mean                 0.009                    0.054
## perimeter_mean              -0.082                   -0.006
## area_mean                   -0.072                   -0.020
## smoothness_mean              0.201                    0.284
## compactness_mean             0.230                    0.507
## concavity_mean               0.178                    0.449
## concave_points_mean          0.095                    0.258
## symmetry_mean                0.449                    0.332
## fractal_dimension_mean       0.345                    0.688
## radius_se                    0.241                    0.228
## texture_se                   0.412                    0.280
## perimeter_se                 0.266                    0.244
## area_se                      0.134                    0.127
## smoothness_se                0.414                    0.427
## compactness_se               0.395                    0.803
## concavity_se                 0.309                    0.727
## concave_points_se            0.313                    0.611
## symmetry_se                  1.000                    0.369
## fractal_dimension_se         0.369                    1.000
```

Yes there is multicollinearity amongst the predictors.

Q 2

Split data by using the following commands library(caret) set.seed(12) tr.ind = createDataPartition(BC_data$diagnosis, p = 0.7, list = F) BC.tr = BC_data[tr.ind,] BC.te = BC_data[-tr.ind,]

Answer 2

```
#library(caret)
set.seed(12)

#tr.ind = createDataPartition(BC_data$diagnosis, p = 0.7, list = F)

tr.ind = sample(seq_len(nrow(BC_data)), 0.7*nrow(BC_data))
BC.tr = BC_data[tr.ind,]
BC.te = BC_data[-tr.ind,]
```

Actually, I was having issue with the command that Key had provided, and so used *tr.ind = sample(seq_len(nrow(BC_data)), 0.7*nrow(BC_data))* per Key's direction.

Q 3

Run a logistic regression model for the data by using the diagnosis column on all the predictors. Report confusion matrices for the test data as well as training data predictions. Display ROC curve for the test data prediction, along with reporting the AUC.

Answer 3

$Logistic\ Regression$

```
## Logistic regression with diagnosis column as response and all other columns as predictors.

library("pROC")
log.reg.bc <- glm(as.factor(diagnosis)~ ., data = BC.tr, family = binomial)
summary(log.reg.bc)
```

```
##
## Call:
## glm(formula = as.factor(diagnosis) ~ ., family = binomial, data = BC.tr)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.59590  -0.09456  -0.02235   0.00006   2.98139
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -34.85532   23.98727  -1.453   0.1462
## radius_mean               4.10019    7.53010   0.545   0.5861
## texture_mean              0.44326    0.10704   4.141 3.46e-05 ***
## perimeter_mean           -0.73169    1.07390  -0.681   0.4957
## area_mean                 0.01420    0.02718   0.522   0.6014
## smoothness_mean          -8.60714   63.10149  -0.136   0.8915
## compactness_mean        -69.37448   55.03986  -1.260   0.2075
## concavity_mean          105.04859   44.72742   2.349   0.0188 *
## concave_points_mean      77.03145   69.97055   1.101   0.2709
## symmetry_mean            53.14925   24.80982   2.142   0.0322 *
## fractal_dimension_mean  295.18481  186.09607   1.586   0.1127
## radius_se               -13.34872   19.52178  -0.684   0.4941
## texture_se               -1.88901    1.04509  -1.808   0.0707 .
## perimeter_se             -0.18925    1.66642  -0.114   0.9096
## area_se                   0.25109    0.15922   1.577   0.1148
## smoothness_se           215.52077  190.33639   1.132   0.2575
## compactness_se          -44.96002   96.98235  -0.464   0.6429
## concavity_se           -100.11968   74.53156  -1.343   0.1792
## concave_points_se       -61.70314  190.00129  -0.325   0.7454
## symmetry_se            -109.20538   71.45722  -1.528   0.1264
## fractal_dimension_se   -235.38753  524.01766  -0.449   0.6533
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 519.798  on 397  degrees of freedom
## Residual deviance:  72.261  on 377  degrees of freedom
## AIC: 114.26
##
## Number of Fisher Scoring iterations: 10
```

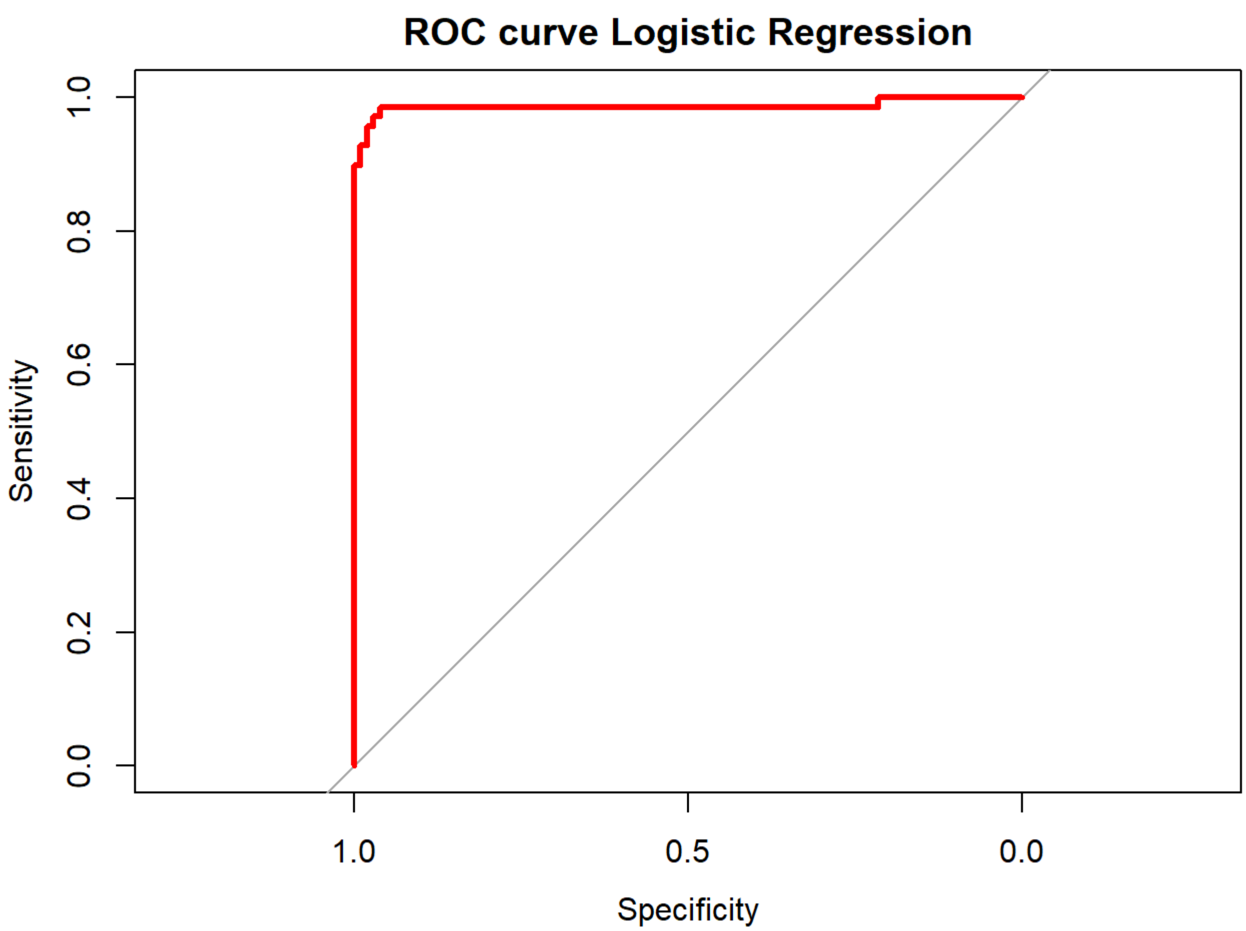
```
test.probs = predict(log.reg.bc, BC.te, type='response')
test.preds = rep("B", nrow(BC.te))
test.preds[test.probs > 0.5] = "M"
table(test.preds, BC.te$diagnosis)
```

```
##
## test.preds  B  M
##          B 99  2
##          M  3 67
```

```
mean(test.preds == BC.te$diagnosis)
```

```
## [1] 0.9707602
```

```
roc.test.log <- roc(response=BC.te$diagnosis, factor(test.probs, ordered = TRUE))
plot(roc.test.log, col="red", lwd=3, main="ROC curve Logistic Regression")
```




```
auc_test_log<-auc(roc.test.log)
auc_test_log
```

```
## Area under the curve: 0.9868
```

```
print(paste("The AUC for the Logistic Regression is:", round(auc_test_log,3)))
```

```
## [1] "The AUC for the Logistic Regression is: 0.987"
```

```
vif(log.reg.bc)
```

```
##             radius_mean            texture_mean          perimeter_mean
##             1819.940622                3.007705             1601.849929
##               area_mean         smoothness_mean        compactness_mean
##              178.829902                5.818491               47.644527
##          concavity_mean     concave_points_mean           symmetry_mean
##               46.813744               16.507485                5.234056
## fractal_dimension_mean               radius_se              texture_se
##               15.314602               51.554197                2.558286
##            perimeter_se                 area_se           smoothness_se
##               21.344324               36.232159                7.138260
##          compactness_se            concavity_se       concave_points_se
##               27.685931               44.726954               15.390530
##             symmetry_se    fractal_dimension_se
##                3.740315               16.563038
```

There is strong multicollinearity amongst predictors.

Q 4

Redo part 3. for LDA and QDA, as well as Naive Bayes methods, respectively.

Answer 4

LDA

```
## LDA

library(MASS)
lda.fit <- lda(diagnosis~., data=BC.tr, family=binomial)
summary(lda.fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   40     -none- numeric
## scaling 20     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     4     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

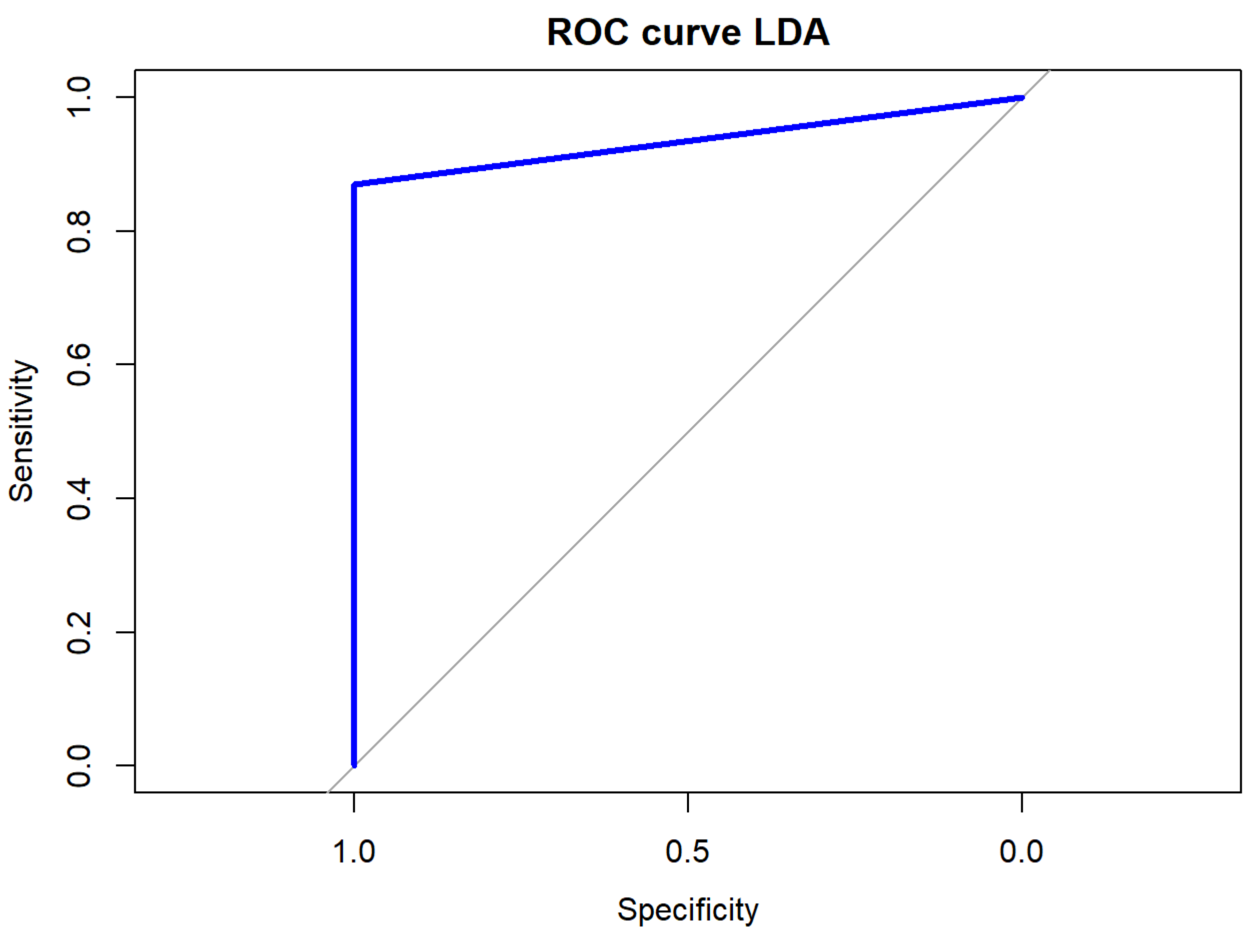
```
test.preds = predict(lda.fit, BC.te)
table(test.preds$class, BC.te$diagnosis)
```

```
##
##       B   M
##   B 102   9
##   M   0  60
```

```
mean(test.preds$class == BC.te$diagnosis)
```

```
## [1] 0.9473684
```

```
roc.test.lda <- roc(response=BC.te$diagnosis, factor(test.preds$class, ordered = TRUE))
plot(roc.test.lda, col="blue", lwd=3, main="ROC curve LDA")
```




```
auc_test_lda<-auc(roc.test.lda)

print(paste("The AUC for the LDA is:", round(auc_test_lda,3)))
```

```
## [1] "The AUC for the LDA is: 0.935"
```

QDA

```
## QDA

qda.fit <-qda(diagnosis~., data=BC.tr, family=binomial)
summary(qda.fit)
```

```
##         Length Class  Mode
## prior     2    -none- numeric
## counts    2    -none- numeric
## means    40    -none- numeric
## scaling 800    -none- numeric
## ldet      2    -none- numeric
## lev       2    -none- character
## N         1    -none- numeric
## call      4    -none- call
## terms     3    terms  call
## xlevels   0    -none- list
```

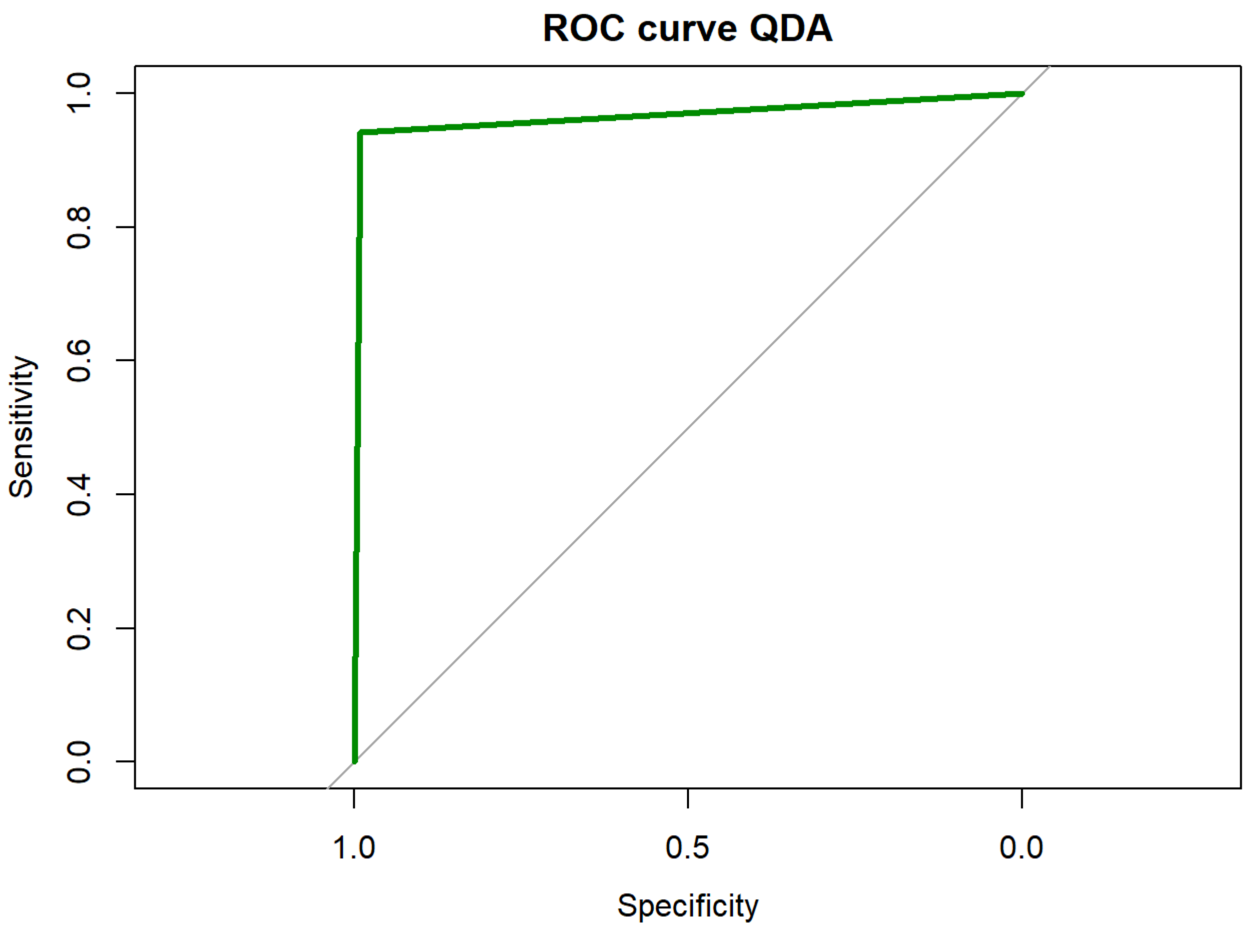
```
test.preds = predict(qda.fit, BC.te)
table(test.preds$class, BC.te$diagnosis)
```

```
##
##       B   M
##   B 101   4
##   M   1  65
```

```
mean(test.preds$class == BC.te$diagnosis)
```

```
## [1] 0.9707602
```

```
roc.test.qda <- roc(response=BC.te$diagnosis, factor(test.preds$class, ordered = TRUE))
plot(roc.test.qda, col="green4", lwd=3, main="ROC curve QDA")
```




```
auc_test_qda<-auc(roc.test.qda)

print(paste("The AUC for the QDA is:", round(auc_test_qda,3)))
```

```
## [1] "The AUC for the QDA is: 0.966"
```

Naive Bayes

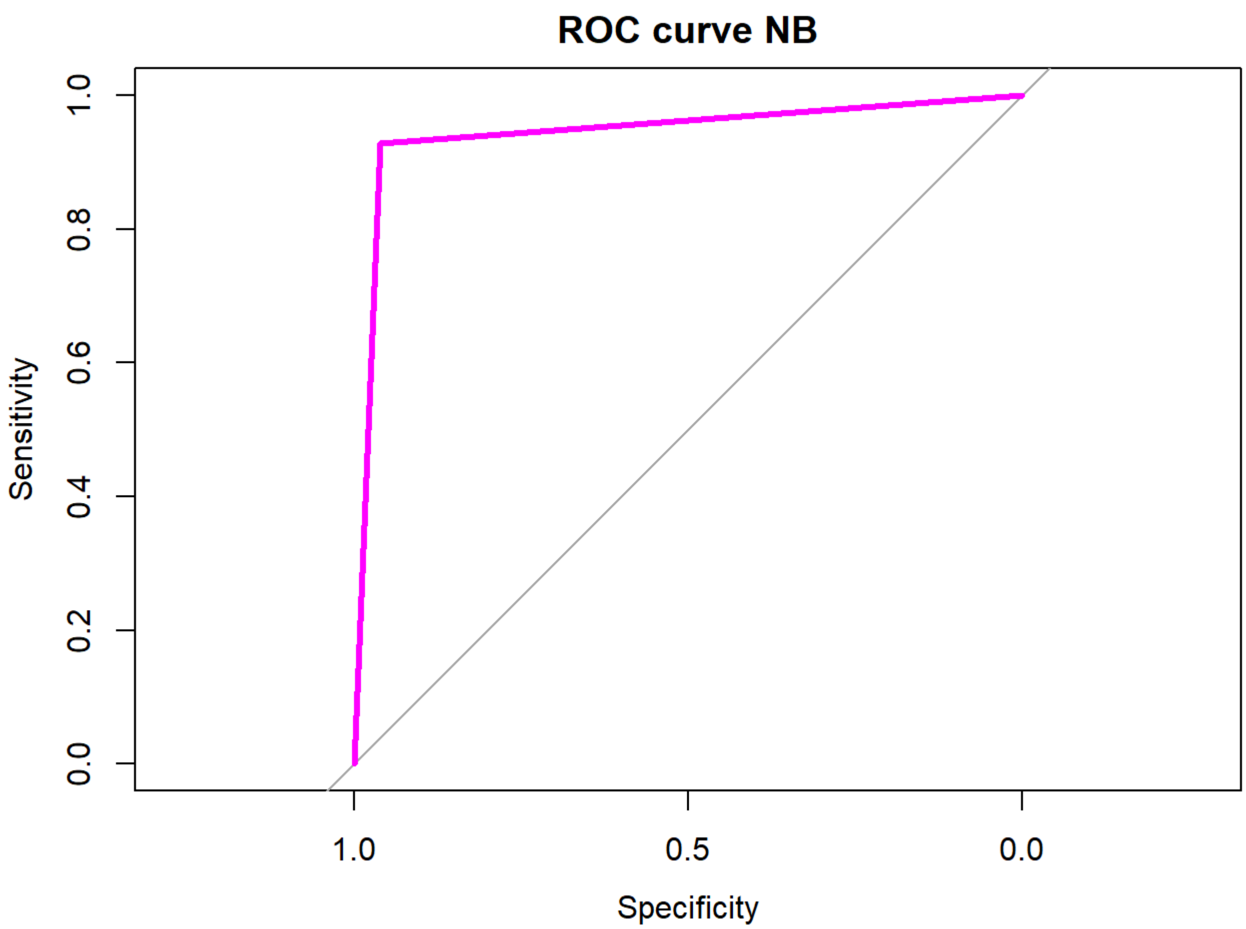
```
## Naive Bayes

library("naivebayes")
nb.fit = naive_bayes(diagnosis~., data=BC.tr, usekernel = T)

test.preds = predict(nb.fit, BC.te)
table(test.preds, BC.te$diagnosis)
```

```
##
## test.preds  B  M
##          B 98  5
##          M  4 64
```

```
mean(test.preds == BC.te$diagnosis)
```

```
## [1] 0.9473684
```

```
roc.test.nb <- roc(response=BC.te$diagnosis, factor(test.preds, ordered = TRUE))
plot(roc.test.nb, col="magenta1", lwd=3, main="ROC curve NB")
```




```
auc_test_nb<-auc(roc.test.nb)


print(paste("The AUC for the Naive Bayes is:", round(auc_test_nb,3)))
```

```
## [1] "The AUC for the Naive Bayes is: 0.944"
```

Q 5

Do part 3. by using the KNN classification method for k=1, and k=7.

Answer 5

$KNN\, k = 1$

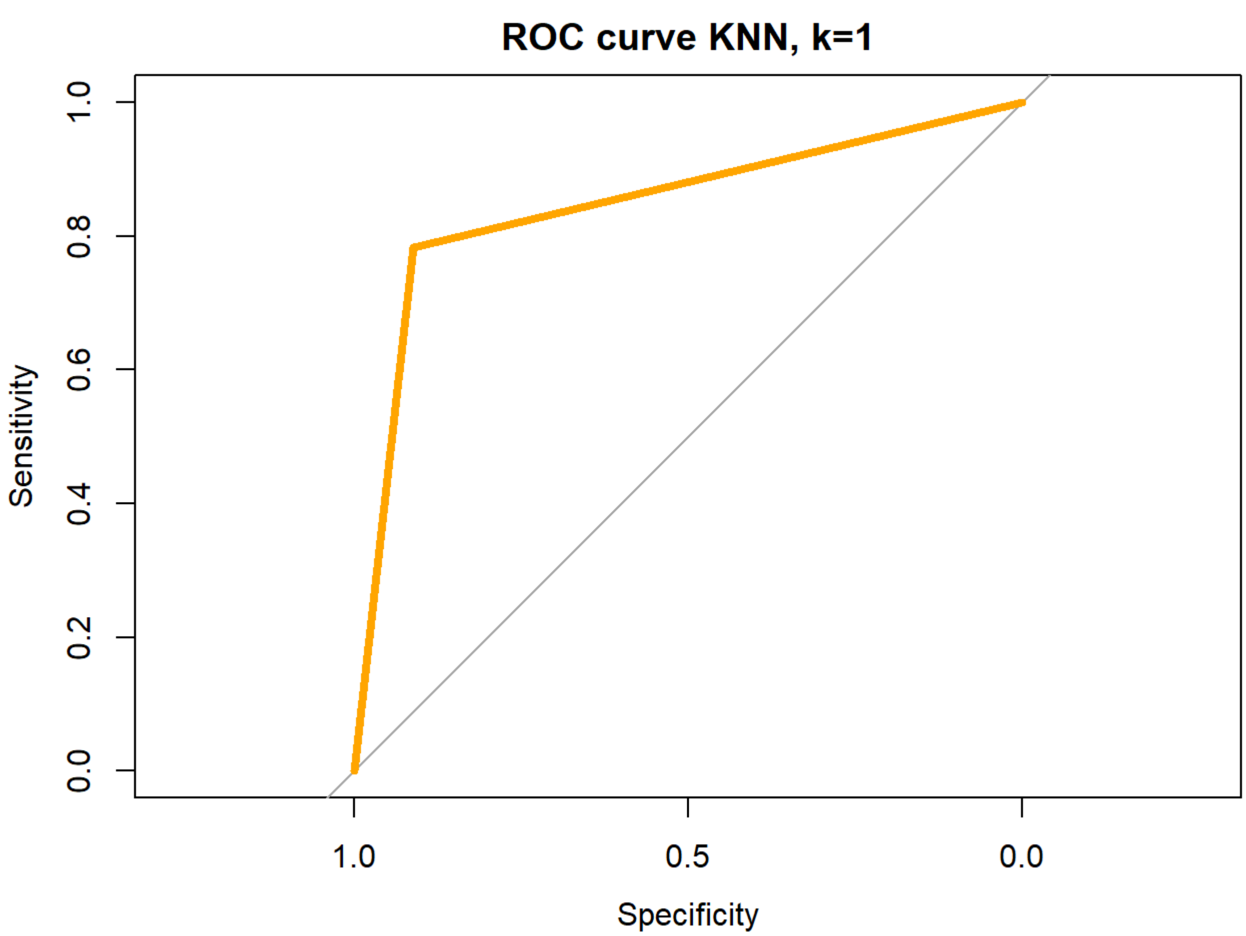
```
## KNN with k = 1

library(ISLR)
library(class)
train.knn = as.matrix(BC.tr[, names(BC.tr) != "diagnosis"])
test.knn  = as.matrix(BC.te[, names(BC.te) != "diagnosis"])


knn.fit = knn(train.knn, test.knn, BC.tr$diagnosis, k=1)
table(knn.fit, BC.te$diagnosis)
```

```
##
## knn.fit  B  M
##       B 93 15
##       M  9 54
```

```
mean(knn.fit == BC.te$diagnosis)
```

```
## [1] 0.8596491
```

```
roc.test.knn1 <- roc(response=BC.te$diagnosis, factor(knn.fit, ordered = TRUE))
plot(roc.test.knn1, col="orange1", lwd=4, main="ROC curve KNN, k=1")
```



```r
auc_test_knn1<-auc(roc.test.knn1)


print(paste("The AUC for the KNN (with k = 1) is:", round(auc_test_knn1,3)))
```

```
## [1] "The AUC for the KNN (with k = 1) is: 0.847"
```

$KNN\, k = 7$

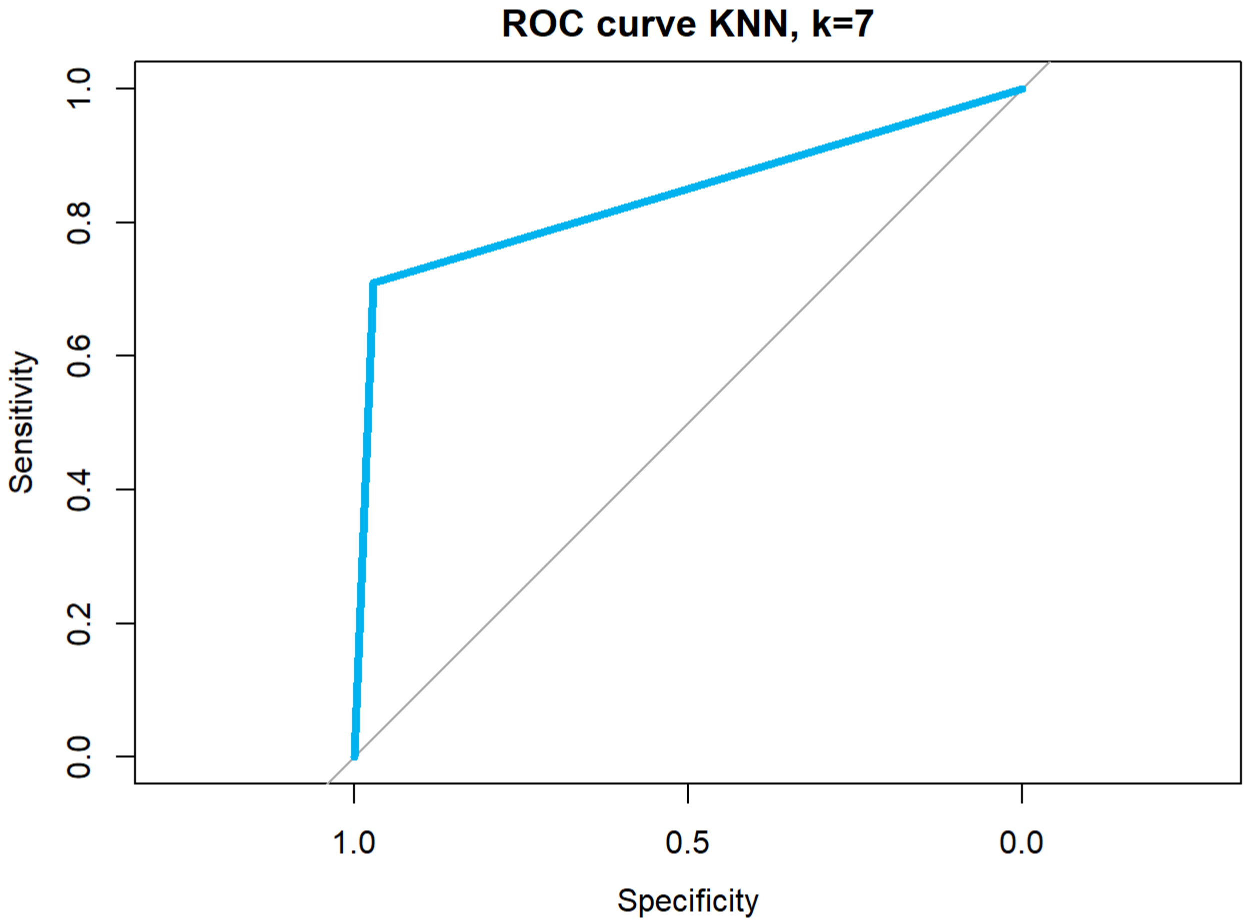
```r
## KNN with k = 1

train.knn = as.matrix(BC.tr[, names(BC.tr) != "diagnosis"])
test.knn  = as.matrix(BC.te[, names(BC.te) != "diagnosis"])


knn.fit7 = knn(train.knn, test.knn, BC.tr$diagnosis, k=7)
table(knn.fit7, BC.te$diagnosis)
```

```
##         
## knn.fit7  B  M
##        B 99 20
##        M  3 49
```

```r
mean(knn.fit7 == BC.te$diagnosis)
```

```
## [1] 0.8654971
```

```r
roc.test.knn7 <- roc(response=BC.te$diagnosis, factor(knn.fit7, ordered = TRUE))
plot(roc.test.knn7, col="deepskyblue2", lwd=4, main="ROC curve KNN, k=7")
```




```
auc_test_knn7<-auc(roc.test.knn7)


print(paste("The AUC for the KNN (with k = 7) is:", round(auc_test_knn7,3)))
```

```
## [1] "The AUC for the KNN (with k = 7) is: 0.84"
```

Q 6

Since KNN uses Euclidean distances to calculate the *distance*, different predictors may affect each other. Scale all the predictors for the original data, i.e., combined with training and test data. Then split the date by "$tr.ind$". Redo part **5.** for this scaled data.

Answer 6

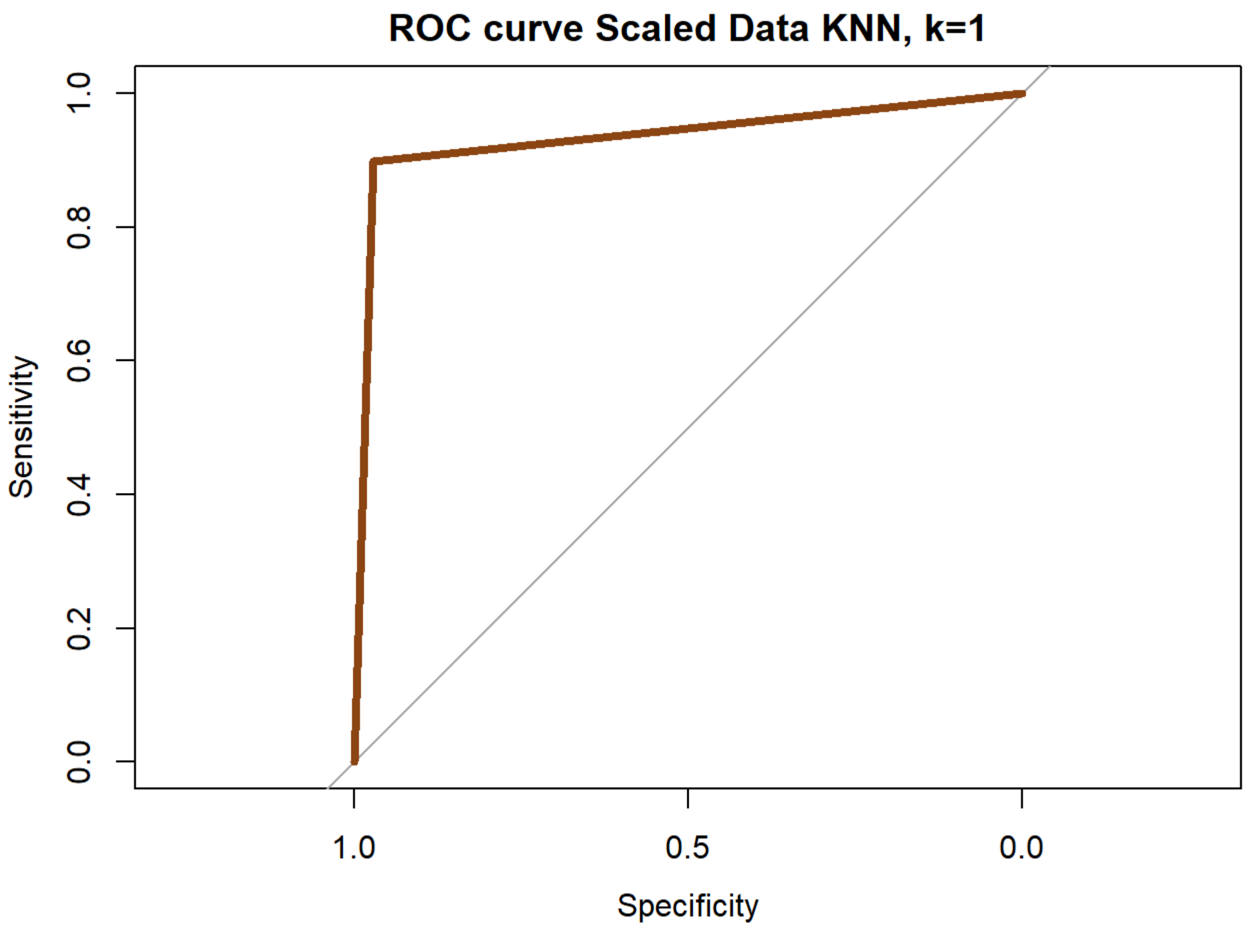
```
BC_data[, names(BC_data) != "diagnosis"] <- scale(BC_data[, names(BC_data) != "diagnosis"])

BC.tr = BC_data[tr.ind,]
BC.te = BC_data[-tr.ind,]

## KNN with k = 1

train.knn = as.matrix(BC.tr[, names(BC.tr) != "diagnosis"])
test.knn  = as.matrix(BC.te[, names(BC.te) != "diagnosis"])


knn.fit61 = knn(train.knn, test.knn, BC.tr$diagnosis, k=1)
table(knn.fit61, BC.te$diagnosis)
```

```
##
## knn.fit61  B  M
##         B 99  7
##         M  3 62
```

```
mean(knn.fit61 == BC.te$diagnosis)
```

```
## [1] 0.9415205
```

```
roc.test.knn61 <- roc(response=BC.te$diagnosis, factor(knn.fit61, ordered = TRUE))
plot(roc.test.knn61, col="chocolate4", lwd=4, main="ROC curve Scaled Data KNN, k=1")
```




```
auc_test_knn61<-auc(roc.test.knn61)


print(paste("The AUC for the KNN (with k = 1) for scaled data is:", round(auc_test_knn61,3)))
```

```
## [1] "The AUC for the KNN (with k = 1) for scaled data is: 0.935"
```

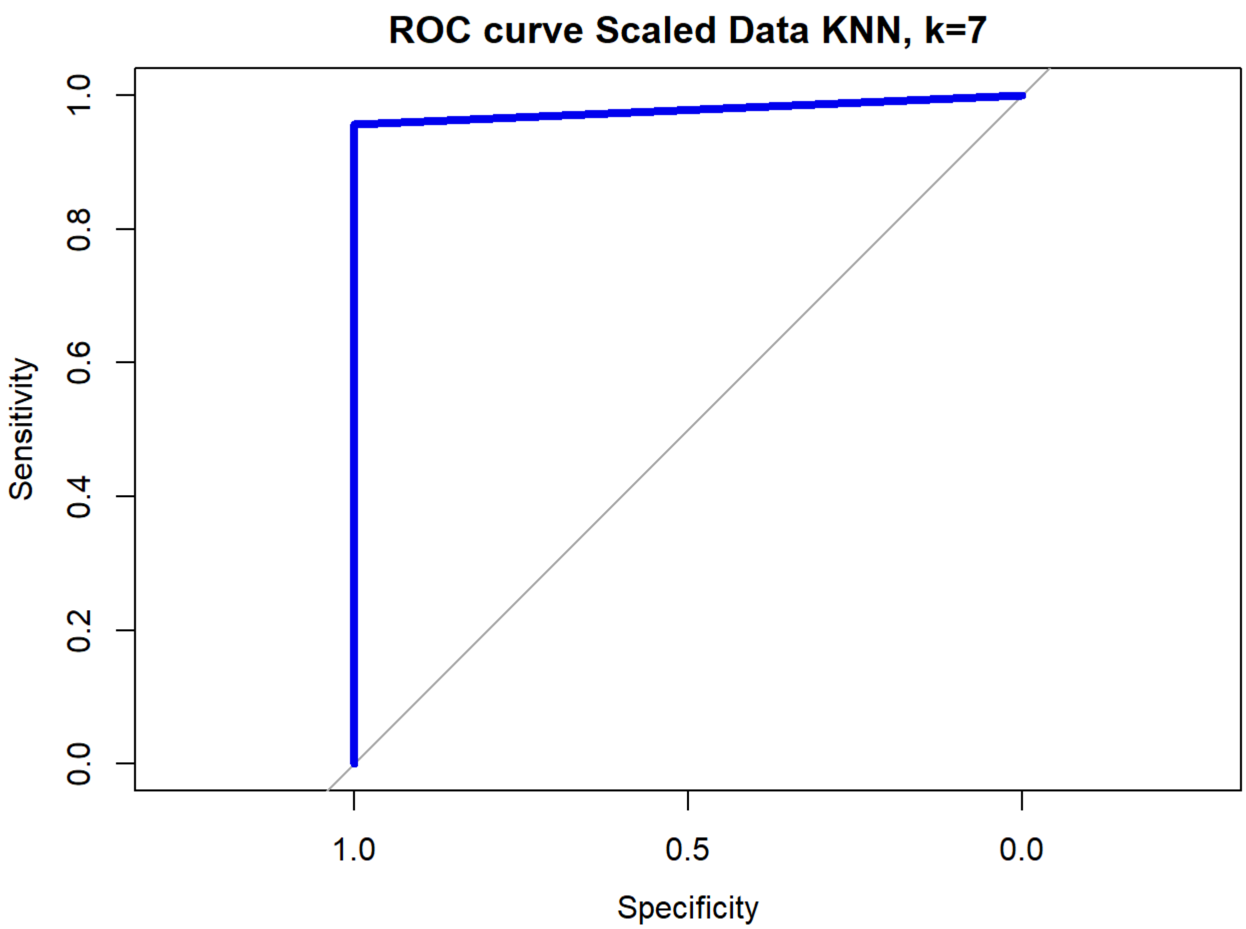
```
## KNN with k = 7

train.knn = as.matrix(BC.tr[, names(BC.tr) != "diagnosis"])
test.knn  = as.matrix(BC.te[, names(BC.te) != "diagnosis"])


knn.fit67 = knn(train.knn, test.knn, BC.tr$diagnosis, k=7)
table(knn.fit67, BC.te$diagnosis)
```

```
##
## knn.fit67   B   M
##          B 102   3
##          M   0  66
```

```
mean(knn.fit67 == BC.te$diagnosis)
```

```
## [1] 0.9824561
```

```
roc.test.knn67 <- roc(response=BC.te$diagnosis, factor(knn.fit67, ordered = TRUE))
plot(roc.test.knn67, col="blue2", lwd=4, main="ROC curve Scaled Data KNN, k=7")
```




```
auc_test_knn67<-auc(roc.test.knn67)


print(paste("The AUC for the KNN (with k = 7) for scaled data is:", round(auc_test_knn67,3)))
```

```
## [1] "The AUC for the KNN (with k = 7) for scaled data is: 0.978"
```

Q 7

Comments all the classification results you've done above.

Answer 7

So, looking at all the classification results, it can be said that Logistic Regression has a better performance at distinguishing between the positive and negative classes as it has the highest AUC. All the other methods , like LDA, QDA and KNN also have high AUC and are close.