Abhishek HW3

```r
# This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter’s lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.
# (a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

library(ISLR)
head(Weekly)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

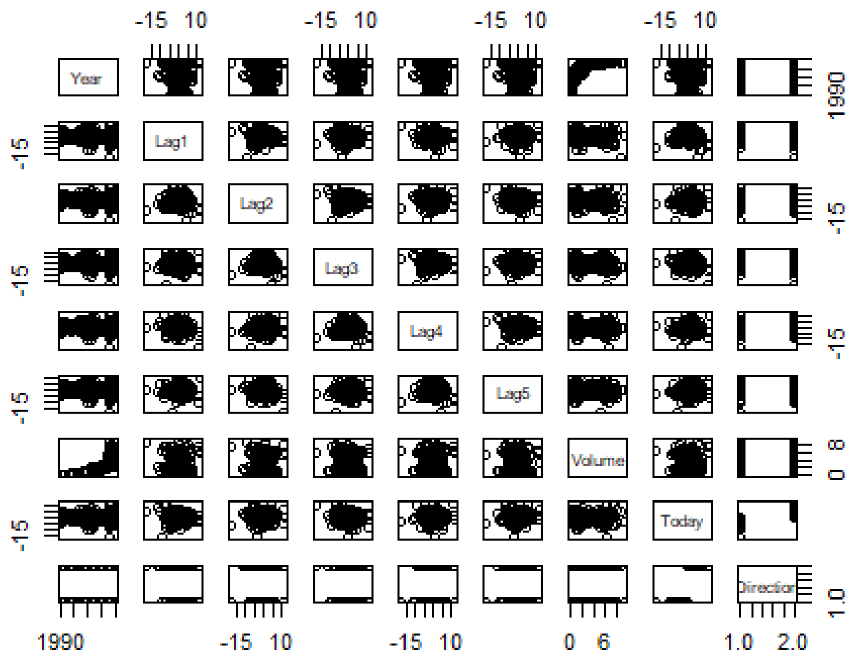
```r
summary(Weekly)
```

```
##       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
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today             Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260
```

```r
nrow(Weekly)
```

```
## [1] 1089
```

```r
pairs(Weekly)
```




```
# There is a pattern between Volume vs Year, Volume increases over time .

# (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?

log_fit = glm(Direction ~ . -Year -Today, data=Weekly, family = "binomial")
summary(log_fit)

##
## Call:
## glm(formula = Direction ~ . - Year - Today, 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
```

```
# The only predictor with a p-value less than 0.05 is “Lag2”. Hence "Lag2"is
the only predictor which is  statistically significant.

# (c) Compute the confusion matrix and performance measures (accuracy, error
rate, sensitivity, specificity). Explain what the confusion matrix is telling
you about the types of mistakes made by logistic regression. Does the error
rate represent the performance of logistic regression in prediction? (hint:
is it training error rate or test error rate?)

log_prob = predict(log_fit, type="response")
log_predict = rep("Down", 1089)
log_predict[log_prob > 0.5] = "Up"
conf_mat <- table(Weekly$Direction, log_predict)
conf_mat
```

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

```
#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (54 + 557)/(54 + 430 + 48 + 557)
= 56.1%
#Sensitivity = TP/(TP + FN) = 557/(557 + 48) = 92.06%
#Specificity = TN/(TN + FP) = 54/(54 + 430) = 11.1%
#The model predicts well for Up direction (92.06% of the time), but it
predict poorly for the Down direction (11.1% of the time).

# (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 performance measures (accuracy, error rate, sensitivity, specificity) for
the held out data (that is, the data from 2009 and 2010).

Weekly_train = Weekly[Weekly$Year <= 2008,]
Weekly_test = Weekly[Weekly$Year > 2008,]
Weekly_test$Direction
```

```
##   [1] Down Down Down Down Up   Down Down Down Down Up   Up   Up   Up   Up
##  [15] Up   Down Up   Up   Down Up   Up   Up   Up   Down Down Down Down Up
##  [29] Up   Up   Up   Down Up   Up   Down Up   Up   Down Down Up   Up
```

```
Down
##  [43] Down Up   Up   Down Up   Up   Up   Down Up   Down Up   Down Down
Down
##  [57] Down Up   Up   Down Up   Up   Up   Up   Up   Up   Down Up   Down
Down
##  [71] Up   Down Up   Down Up   Up   Down Down Up   Down Up   Down Up
Down
##  [85] Down Down Up   Up   Up   Up   Down Up   Up   Up   Up   Up   Down Up
##  [99] Down Up   Up   Up   Up   Up
## Levels: Down Up

nrow(Weekly_train)

## [1] 985

nrow(Weekly_test)

## [1] 104

log_fit = glm(Direction ~ Lag2, data=Weekly_train, family = "binomial")
log_prob = predict(log_fit, Weekly_test, type="response")
log_predict = rep("Down", 104)
log_predict[log_prob > 0.5] = "Up"
conf_mat <- table(Weekly_test$Direction, log_predict)
conf_mat

##       log_predict
##        Down Up
##   Down    9 34
##   Up      5 56

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (56 + 9)/(56 + 34 + 9 + 5) =
62.5%
#Error rate = (1 - Accuracy) = 37.5%
#Sensitivity = TP/(TP + FN) = 56/(56 + 5) = 91.8%
#Specificity = TN/(TN + FP) = 9/(9 + 34) = 20.9%

# (e) Repeat (d) using LDA.
library(MASS)
lda_fit = lda(Direction ~ Lag2, data=Weekly_train)
lda_pred = predict(lda_fit, Weekly_test)
lda_pred$class

##   [1] Up   Up   Down Down Up   Up   Up   Down Down Down Down Up   Up   Up
##  [15] Up   Up   Up   Up   Up   Up   Down Up   Up   Up   Up   Up   Up   Up
##  [29] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [43] Up   Up   Down Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [57] Down Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [71] Up   Down Up   Down Up   Up   Up   Up   Down Down Up   Up   Up   Up
##  [85] Up   Down Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [99] Up   Up   Up   Up   Up   Up
## Levels: Down Up
```

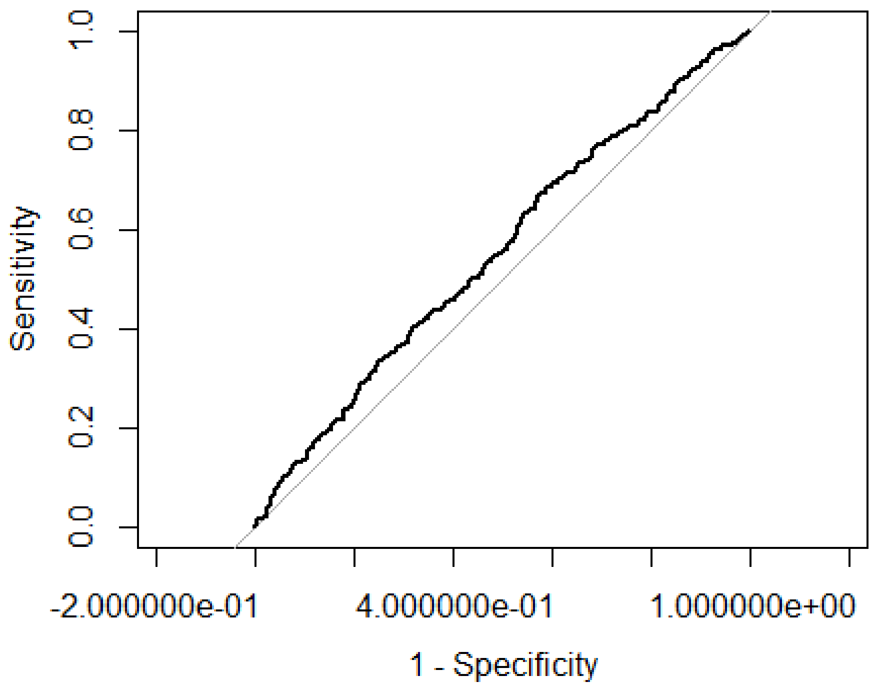
```
conf_mat = table(Weekly_test$Direction, lda_pred$class)
conf_mat

##
##        Down Up
##   Down    9 34
##   Up      5 56

#The model makes correct predictions for 62.5% of the test data. The results
are pretty close to those obtained in the logistic regression model.

# (f) Repeat (d) using QDA.
library(MASS)
qda_fit = qda(Direction ~ Lag2, data=Weekly_train)
qda_pred = predict(qda_fit, Weekly_test)
qda_pred$class

##   [1] Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up
##  [24] Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up
##  [47] Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up
##  [70] Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up
##  [93] Up Up Up Up Up Up Up Up Up Up Up Up
## Levels: Down Up

conf_mat = table(Weekly_test$Direction, qda_pred$class)
conf_mat

##
##        Down Up
##   Down    0 43
##   Up      0 61

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (61)/(61 + 0 + 0 + 43) = 58.6%
#Sensitivity = TP/(TP + FN) = 61/(0 + 61) = 100%
#Specificity = TN/(TN + FP) = 0/(0 + 43) = 0%
# The model makes correct predictions for 58.6% of the test data. The model
predicts well for Up direction (100% of the time), but it predicts poorly for
the Down direction (0% of the time).

# (g) Repeat (d) using KNN with K = 1.
library(class)
knn_pred = knn(as.matrix(Weekly_train$Lag2), as.matrix(Weekly_test$Lag2),
as.matrix(Weekly_train$Direction),  k = 1)
table(Weekly_test$Direction, knn_pred)

##       knn_pred
##        Down Up
##   Down   21 22
##   Up     29 32

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (32+21)/(21 + 22 + 32 + 29) =
53/104 = 50.9%
```

```r
#Sensitivity = TP/(TP + FN) = 32/(29 + 32) = 52.45%
#Specificity = TN/(TN + FP) = 21/(21 + 22) = 48.8%
# The model makes correct predictions for 50.9% of the test data. The model
predicts right 52.45% of the time for the Up Direction and predicts right
48.8% of the time for the Down direction.

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

# Accuracy with Logistic Regression : 62.5%
# Accuracy with LDA : 62.5%
# Accuracy with QDA : 58.6%
# Accuracy with KNN : 52.45%
# Logistic regression and LDA have the maximum Accuracy.

# (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 classifiers.

# Logistic Regression
log_fit = glm(Direction ~ Lag2:Lag1, data=Weekly_train, family = "binomial")
log_prob = predict(log_fit, Weekly_test, type="response")
log_predict = rep("Down", 104)
log_predict[log_prob > 0.5] = "Up"
table(Weekly_test$Direction, log_predict)

##       log_predict
##        Down Up
##   Down    1 42
##   Up      1 60

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (1+60)/(1 + 60 + 1 + 42) = 61/104
= 58.6%



# LDA
lda_fit = lda(Direction ~ Lag2:Lag1, data=Weekly_train)
lda_pred = predict(lda_fit, Weekly_test)
lda_pred$class

##   [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up   Up
##  [15] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [29] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [43] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [57] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [71] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [85] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [99] Up   Up   Up   Up   Up   Up
## Levels: Down Up
```

```r
conf_mat = table(Weekly_test$Direction, lda_pred$class)
conf_mat

##
##         Down Up
##   Down     0 43
##   Up       1 60

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (0+60)/(0 + 60 + 1 + 43) = 60/104
= 57.7%



# QDA
qda_fit = qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data=Weekly_train)
qda_pred = predict(qda_fit, Weekly_test)
qda_pred$class

##   [1] Down Up   Down Down Down Up   Up   Down Down Down Down Up   Up   Up
##  [15] Up   Up   Up   Up   Up   Up   Down Up   Up   Up   Up   Down Up
Down
##  [29] Down Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Down Up
##  [43] Up   Up   Down Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [57] Down Down Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [71] Down Down Up   Down Up   Down Up   Up   Down Down Up   Up   Up   Up
##  [85] Up   Down Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up
##  [99] Down Up   Up   Up   Up   Up
## Levels: Down Up

conf_mat = table(Weekly_test$Direction, qda_pred$class)
conf_mat

##
##         Down Up
##   Down    12 31
##   Up      13 48

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (48+12)/(48 + 12 + 13 + 31) =
60/104 = 57.7%

# K = 5.
library(class)
knn_pred = knn(as.matrix(Weekly_train$Lag2), as.matrix(Weekly_test$Lag2),
as.matrix(Weekly_train$Direction),  k = 5)
table(Weekly_test$Direction, knn_pred)

##        knn_pred
##         Down Up
##   Down    15 28
##   Up      21 40
```

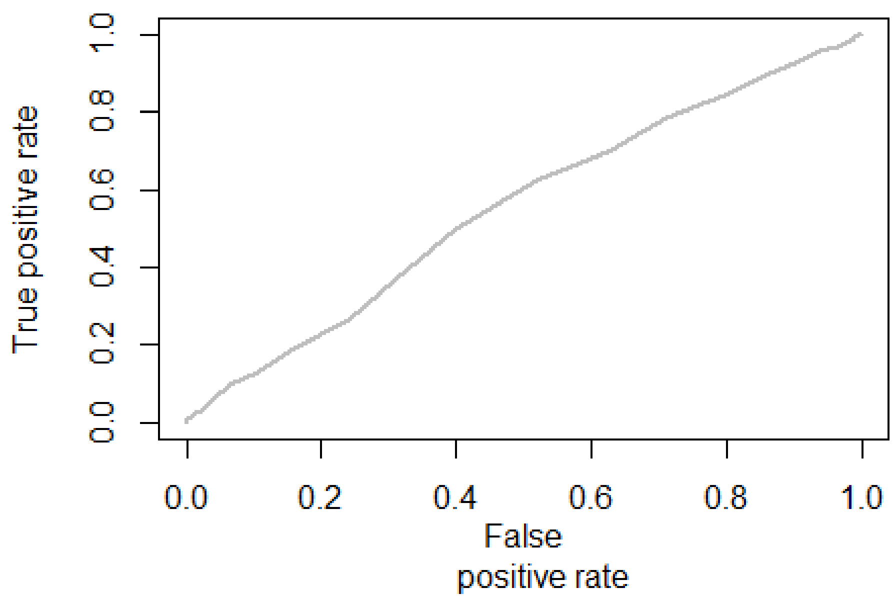
```r
#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (40+16)/(40 + 16 + 27 + 21) =
56/104 = 53.8%

# K = 10
library(class)
knn_pred = knn(as.matrix(Weekly_train$Lag2), as.matrix(Weekly_test$Lag2),
as.matrix(Weekly_train$Direction),  k = 10)
table(Weekly_test$Direction, knn_pred)

##       knn_pred
##        Down Up
##   Down   18 25
##   Up     20 41

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (17+42)/(17 + 42 + 19 + 26) =
59/104 = 56.7%

# K = 15
library(class)
knn_pred = knn(as.matrix(Weekly_train$Lag2), as.matrix(Weekly_test$Lag2),
as.matrix(Weekly_train$Direction),  k = 15)
table(Weekly_test$Direction, knn_pred)

##       knn_pred
##        Down Up
##   Down   20 23
##   Up     20 41

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (20+41)/(20 + 41 + 20 + 23) =
61/104 = 58.6%


# Among the experiments done above, logistic regression with interaction term
seems to perform better. But if we go back, the old model for logistic
regression and LDA gave us more accuracy.

# Q2) Perform ROC analysis and present the results for logistic regression
and LDA used for the best model chosen in Question 1(i).

# Logistic Regression

library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
log_fit = glm(Direction ~ . -Year -Today, data=Weekly, family = "binomial")
log_prob = predict(log_fit, type="response")
a = roc(Direction~log_prob, data=Weekly)
plot(a, legacy.axes=T)
```




```r
# Area under the curve is around 0.53 and it's not a good fit.

# LDA
library(ISLR)
attach(Weekly)
library(MASS)
LDA.fit = lda(Direction~Lag1+Lag2+Lag4,data=Weekly)
LDA.pred0 = predict(LDA.fit,type="response")
LDA.pred = LDA.pred0$posterior[,2]
roc.curve=function(s,print=FALSE){
  Ps=(LDA.pred>s)*1
  FP=sum((Ps==1)*(Direction=="Down"))/sum(Direction=="Down")
  TP=sum((Ps==1)*(Direction=="Up"))/sum(Direction=="Up")
  if(print==TRUE){
    print(table(Observed=Direction,Predicted=Ps))
  }
  vect=c(FP,TP)
  names(vect)=c("FPR","TPR")
  return(vect)
}
threshold=0.5
roc.curve(threshold,print=TRUE)
```

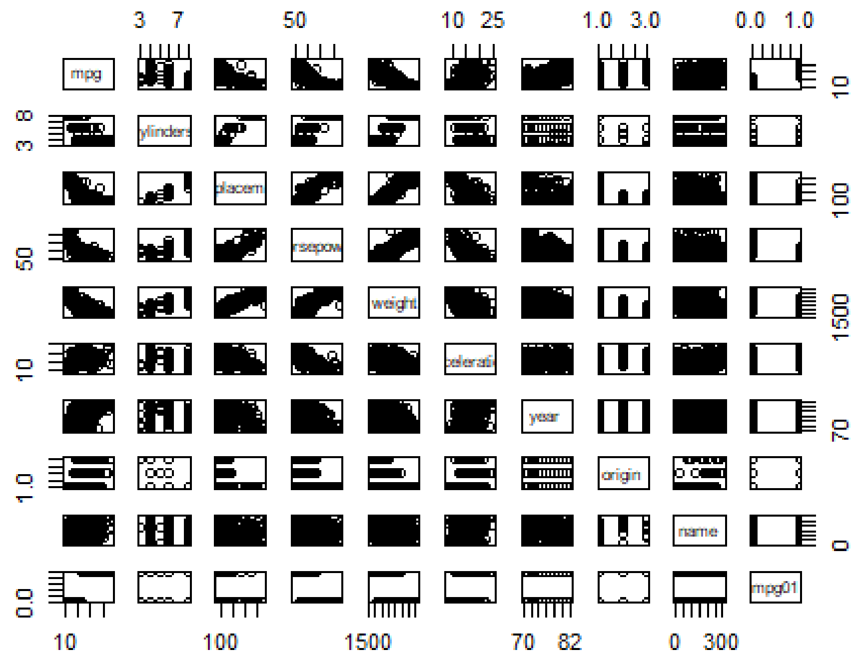
```
##          Predicted
## Observed   0   1
##     Down  44 440
##     Up    40 565

##       FPR       TPR
## 0.9090909 0.9338843

ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=0.01))
plot(M.ROC[1,],M.ROC[2,],col="grey",lwd=2,type="l"
     ,xlab="False
     positive rate"
     ,ylab="True positive rate")
```



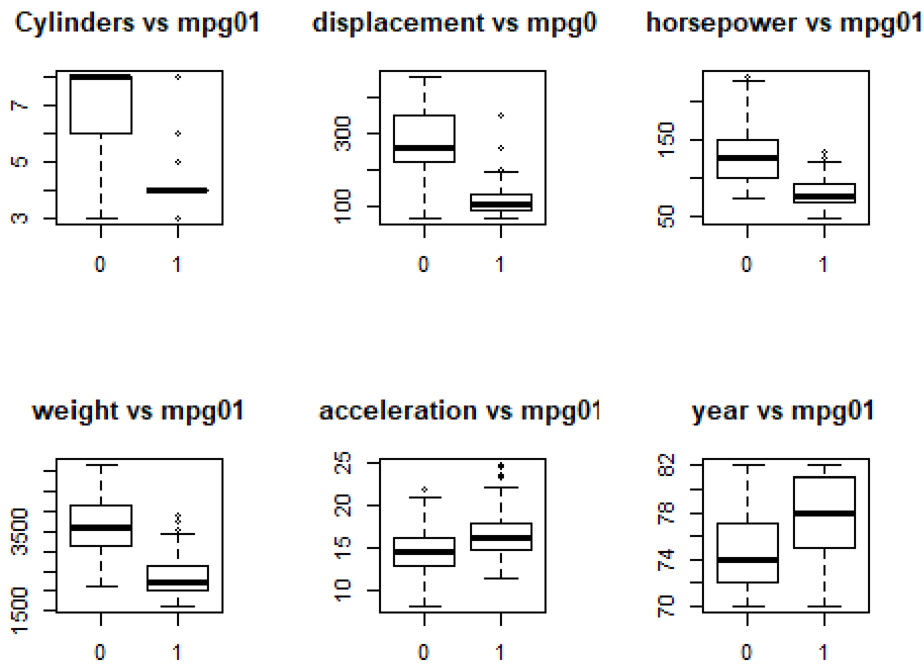

```
# Area under the curve is around 0.5 and it’s a bad fit. This is inferior to the Logistic regression.

# Q3 In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

# (a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median( )

library(ISLR)
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##                  name
##  amc matador       :  5
##  ford pinto        :  5
##  toyota corolla    :  5
##  amc gremlin       :  4
##  amc hornet        :  4
##  chevrolet chevette:  4
##  (Other)           :365
```

```r
attach(Auto)
mpg01 = rep(0, length(mpg))
mpg01[mpg > median(mpg)] = 1
Auto = data.frame(Auto, mpg01)


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

pairs(Auto)
```




```r
par(mfrow=c(2,3))
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")
boxplot(displacement ~ mpg01, data = Auto, main = "displacement vs mpg01")
boxplot(horsepower ~ mpg01, data = Auto, main = "horsepower vs mpg01")
boxplot(weight ~ mpg01, data = Auto, main = "weight vs mpg01")
boxplot(acceleration ~ mpg01, data = Auto, main = "acceleration vs mpg01")
boxplot(year ~ mpg01, data = Auto, main = "year vs mpg01")
```




```
# There exists some association between “mpg01” and “cylinders”, “weight”,
“displacement” and “horsepower”.

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

rows <- sample(x=nrow(Auto), size=.75*nrow(Auto))
auto_trainset <- Auto[rows, ]
auto_testset <- Auto[-rows, ]

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

library(MASS)
lda_fit <- lda(mpg01 ~ cylinders + weight + displacement + horsepower,
data=auto_trainset)
lda_pred <- predict(lda_fit, auto_testset)
table(auto_testset$mpg01, lda_pred$class)

##
##      0  1
##   0 42  9
##   1  2 45

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (43+48)/(43 + 2 + 5 + 48) = 91/98
= 92.8%
```

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

qda_fit <- qda(mpg01 ~ cylinders + weight + displacement + horsepower,
data=auto_trainset)
qda_pred <- predict(qda_fit, auto_testset)
table(auto_testset$mpg01, qda_pred$class)

##
##      0  1
##   0 45  6
##   1  4 43

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (45+47)/(45 + 3 + 3 + 47) = 92/98
= 93.8%

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

log_fit <- glm(mpg01 ~ cylinders + weight + displacement + horsepower,
data=auto_trainset, family="binomial")
log_prob = predict(log_fit, auto_testset, type="response")
log_predict = rep(0, length(auto_testset$mpg))
log_predict[log_prob > 0.5] = 1
table(auto_testset$mpg01, log_predict)

##    log_predict
##      0  1
##   0 42  9
##   1  4 43

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (45+45)/(45 + 3 + 5 + 45) = 90/98
= 91.8%

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

library(class)
knn_pred = knn(as.matrix(auto_trainset[,2:5]), as.matrix(auto_testset[,2:5]),
as.matrix(auto_trainset$mpg01), k = 1)
table(auto_testset$mpg01, knn_pred)

##    knn_pred
##      0  1
##   0 42  9
##   1  5 42
```

```r
#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (44+41)/(41 + 7 + 6 + 44) = 85/98
= 86.7%

knn_pred = knn(as.matrix(auto_trainset[,2:5]), as.matrix(auto_testset[,2:5]),
as.matrix(auto_trainset$mpg01), k = 3)
table(auto_testset$mpg01, knn_pred)

##    knn_pred
##      0  1
##   0 43  8
##   1  4 43

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (44+46)/(44 + 4+ 4 + 46) = 90/98
= 91.8%

knn_pred = knn(as.matrix(auto_trainset[,2:5]), as.matrix(auto_testset[,2:5]),
as.matrix(auto_trainset$mpg01), k = 5)
table(auto_testset$mpg01, knn_pred)

##    knn_pred
##      0  1
##   0 41 10
##   1  4 43

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (44+44)/(44 + 4 + 6 + 44) = 88/98
= 89.8%

knn_pred = knn(as.matrix(auto_trainset[,2:5]), as.matrix(auto_testset[,2:5]),
as.matrix(auto_trainset$mpg01), k = 7)
table(auto_testset$mpg01, knn_pred)

##    knn_pred
##      0  1
##   0 40 11
##   1  2 45

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (45+43)/(43 + 5 + 5 + 45) = 88/98
= 89.8%

knn_pred = knn(as.matrix(auto_trainset[,2:5]), as.matrix(auto_testset[,2:5]),
as.matrix(auto_trainset$mpg01), k = 10)
table(auto_testset$mpg01, knn_pred)

##    knn_pred
##      0  1
##   0 41 10
##   1  3 44

#Accuracy = (TP + TN)/(TP + FN + FP + TN) = (44+42)/(42 + 6 + 6 + 44) = 86/98
= 87.7%
```

```
# Accuracy for K=1, 3, 5, 7 and 10 are 86.7%, 91.8%, 89.8, 89.8, and 87.7
respectively. Of these K=3 performs the best.
```