HW3_ISEN 613

Gourav Ghoshal

October 12, 2018

Prob. 1

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)
```

```
## Warning: package 'ISLR' was built under R version 3.4.2
```

```
names(Weekly)
```

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

```
dim(Weekly)
```

```
## [1] 1089    9
```

```
head(Weekly)
```

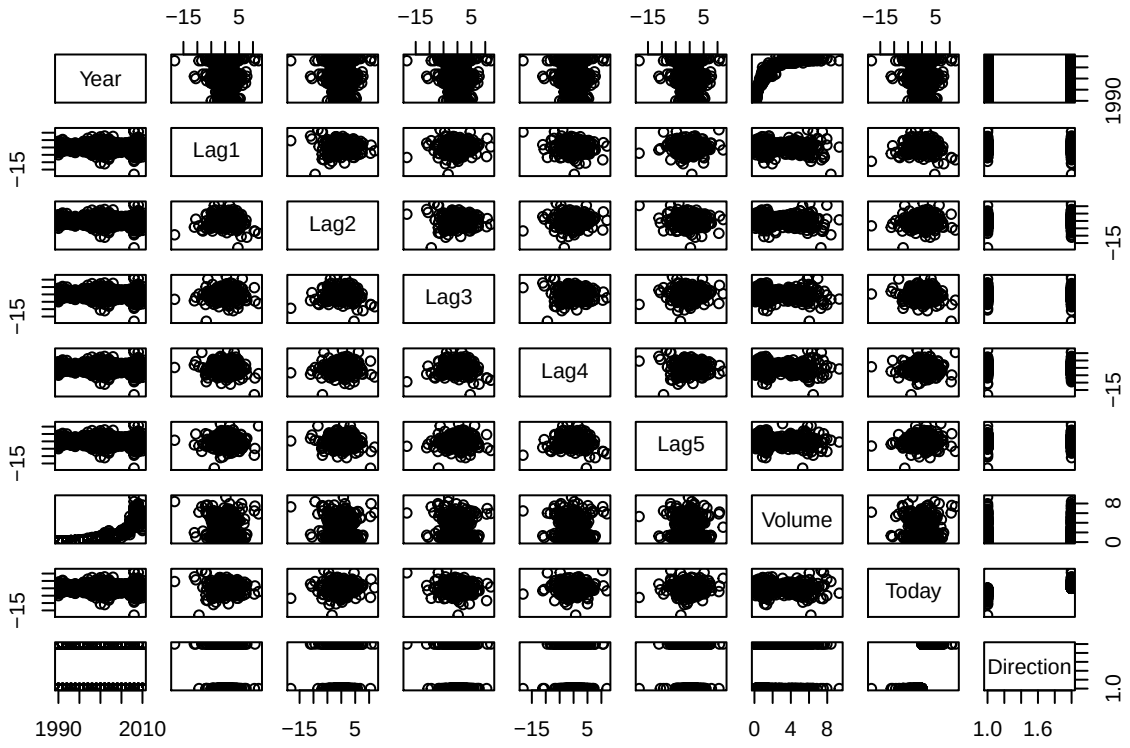
```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  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
```

```
attach(Weekly)
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
```

```
pairs(Weekly)
```




```
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
```

There is a strong relationship between year and volume - as the year increased the volume increased significantly and relationship is non-linear (possibly exponential).

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?

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

```
##
## 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
```

Yes.

Only the Lag2 is statistically significant, in addition to the intercept term.

(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?)

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

```
pred_probs = predict(lr_fit, type = 'response')
pred_class = rep('Down', dim(Weekly)[1])
pred_class[pred_probs > 0.5] = 'Up'
table(pred_class, Direction)
```

```
##           Direction
## pred_class Down  Up
##       Down   54  48
##       Up    430 557
```

```
mean(pred_class == Direction)
```

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

The confusion matrix shows that the logistic regression model predicts 54 times correctly that the market will go down and 557 times correctly that the market will up. But, it also predicts 48 time that the market will go down, when it goes up and 430 time that market will go up when it goes down. Sensitivity = 54/(54+430) = 0.1115 Specificity = 557/(557+48) = 0.9206

No, the error does not represent the correct performance because it the training error and not the test error.

(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).

Train-test split

```
indx = (Year > 2009)
train = Weekly[-indx,]
dim(train)
```

```
## [1] 1088    9
```

```
test = Weekly[indx,]
dim(test)
```

```
## [1] 52  9
```

Model fitting

```
lr_fit2 <- glm(Direction ~ Lag2, data = train, family = binomial)
summary(lr_fit2)
```

```
##
```

```
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.567  -1.268   1.007   1.085   1.388
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.21683    0.06126   3.539 0.000401 ***
## Lag2         0.06338    0.02638   2.402 0.016285 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1488.7  on 1086  degrees of freedom
## AIC: 1492.7
##
## Number of Fisher Scoring iterations: 4
```

```
pred_probs2 = predict(lr_fit2, test, type = 'response')
pred_class2 = rep('Down', dim(test)[1])
pred_class2[pred_probs2 > 0.5] = 'Up'
table(pred_class2, test$Direction)
```

```
##
## pred_class2 Down Up
##        Down    5  1
##        Up     15 31
```

```
mean(pred_class2 == test$Direction)
```

```
## [1] 0.6923077
```

```
mean(pred_class2 != test$Direction)
```

```
## [1] 0.3076923
```

Accuracy = 0.625 Sensitivity = 9/(9+34) = 0.209 Specificity = 56/(56+5) = 0.918 Error rate = 0.375

(e) Repeat (d) using LDA.

```
library(MASS)
lda_fit <- lda(Direction ~ Lag2, data = train)
summary(lda_fit)
```

```
##         Length Class  Mode
## prior   2      -none- numeric
## counts  2      -none- numeric
## means   2      -none- numeric
## scaling 1      -none- numeric
## lev     2      -none- character
## svd     1      -none- numeric
```

```
## N       1      -none- numeric
## call    3      -none- call
## terms   3      terms  call
## xlevels 0      -none- list
```

```
pred_lda = predict(lda_fit, test)$class
table(pred_lda, test$Direction)
```

```
##
## pred_lda Down Up
##     Down    5  1
##     Up     15 31
```

```
mean(pred_lda == test$Direction)
```

```
## [1] 0.6923077
```

```
mean(pred_lda != test$Direction)
```

```
## [1] 0.3076923
```

LDA predicts that the result is Down for all. Accuracy = 0.4134 Sensitivity = 1 Specificity = 0 Error rate = 0.5865

(f) Repeat (d) using QDA.

```
library(MASS)
qda_fit <- qda(Direction ~ Lag2, data = train)
qda_fit
```

```
## Call:
## qda(Direction ~ Lag2, data = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4439338 0.5560662
##
## Group means:
##            Lag2
## Down -0.0437619
## Up    0.3042810
```

```
pred_qda = predict(qda_fit, test)$class
pred_qda
```

```
##  [1] Up 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 Up
## [47] Up Up Up Up Up Up
## Levels: Down Up
```

```
table(pred_qda, test$Direction)
```

```
##
## pred_qda Down Up
##     Down    0  0
##     Up     20 32
```

```
mean(pred_qda == test$Direction)
```

```
## [1] 0.6153846
```

```
mean(pred_qda != test$Direction)
```

```
## [1] 0.3846154
```

QDA predicts that the result is Up for all. Accuracy = 0.5865 Sensitivity = 0 Specificity = 1 Error rate = 0.4134

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

```
library(class)
train_x = cbind(Lag2)[indx,]
train_y = Direction[indx]
test_x = cbind(Lag2)[!indx,]
knn_fit = knn(train = Weekly[3], test = Weekly[3], Weekly$Direction, k =
1)
table(knn_fit, Weekly$Direction)
```

```
##
## knn_fit Down  Up
##    Down  464  23
##    Up     20 582
```

```
mean(knn_fit == Weekly$Direction)
```

```
## [1] 0.9605142
```

```
mean(knn_fit != Weekly$Direction)
```

```
## [1] 0.03948577
```

Accuracy = 0.96 Sensitivity = 461/(461+23) = 0.9524 Specificity = 586/(586+19) = 0.9685 Error rate =0.0385

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

knn gives the best result with highest scores in every metrics

(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.

Different KNN models

```
knn_2 <- knn(train = Weekly[1:8], test = Weekly[1:8], Weekly$Direction, k = 2)
table(knn_2, Weekly$Direction)
```

```
##
## knn_2  Down  Up
```

```
##   Down  440  55
##   Up     44 550
```

```
knn_3 <- knn(train = Weekly[1:8], test = Weekly[1:8], Weekly$Direction, k = 3)
table(knn_3, Weekly$Direction)
```

```
##
## knn_3  Down  Up
##   Down  448  26
##   Up     36 579
```

```
knn_4 <- knn(train = Weekly[1:8], test = Weekly[1:8], Weekly$Direction, k = 4)
table(knn_4, Weekly$Direction)
```

```
##
## knn_4  Down  Up
##   Down  434  40
##   Up     50 565
```

```
knn_5 <- knn(train = Weekly[1:8], test = Weekly[1:8], Weekly$Direction, k = 5)
table(knn_5, Weekly$Direction)
```

```
##
## knn_5  Down  Up
##   Down  448  30
##   Up     36 575
```

Different Logistic Regression models

```
lr_model2 = glm(Direction ~ polym(Year+Lag1+Lag2+Lag3+Lag4+Lag5+Volume+Today, degree = 2), data = train
summary(lr_model2)
```

```
##
## Call:
## glm(formula = Direction ~ polym(Year + Lag1 + Lag2 + Lag3 + Lag4 +
##     Lag5 + Volume + Today, degree = 2), family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7812  -1.2158   0.8894   1.0886   1.4392
##
## Coefficients:
##                                                                                 Estimate
## (Intercept)                                                                      0.23287
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)1 11.15754
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)2  0.48354
##                                                                                 Std. Error
## (Intercept)                                                                        0.06196
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)1    2.11994
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)2    2.22707
##                                                                                 z value
## (Intercept)                                                                       3.758
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)1   5.263
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)2   0.217
##                                                                                 Pr(>|z|)
## (Intercept)                                                                     0.000171
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)1 1.42e-07
```

```
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)2 0.828117
##
## (Intercept)                                                                  ***
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)1 ***
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 2)2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1465.5  on 1085  degrees of freedom
## AIC: 1471.5
##
## Number of Fisher Scoring iterations: 4
```

The AIC for 2nd degree polynomial fit is = 106.53

```
lr_model3 = glm(Direction ~ polym(Year+Lag1+Lag2+Lag3+Lag4+Lag5+Volume+Today, degree = 3), data = train
summary(lr_model3)
```

```
##
## Call:
## glm(formula = Direction ~ polym(Year + Lag1 + Lag2 + Lag3 + Lag4 +
##     Lag5 + Volume + Today, degree = 3), family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3602  -1.2498   0.9228   1.0793   1.9804
##
## Coefficients:
##                                                                              Estimate
## (Intercept)                                                                   0.23855
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)1 12.50562
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)2  2.37848
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)3  7.06694
##                                                                              Std. Error
## (Intercept)                                                                     0.06252
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)1    2.41088
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)2    2.88185
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)3    3.48578
##                                                                              z value
## (Intercept)                                                                    3.815
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)1   5.187
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)2   0.825
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)3   2.027
##                                                                              Pr(>|z|)
## (Intercept)                                                                  0.000136
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)1 2.14e-07
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)2 0.409182
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)3 0.042626
##
## (Intercept)                                                                  ***
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)1 ***
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)2
```

```
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 3)3 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1460.2  on 1084  degrees of freedom
## AIC: 1468.2
##
## Number of Fisher Scoring iterations: 4
```

AIC slightly improved to 106.53

```
lr_model3 = glm(Direction ~ (Year+Lag1+Lag2+Lag3+Lag4+Lag5+Volume+Today)^3, data = train, family = bino
```

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

```
lr_model4 = glm(Direction ~ polym(Year+Lag1+Lag2+Lag3+Lag4+Lag5+Volume+Today, degree = 4), data = train
summary(lr_model4)
```

```
##
## Call:
## glm(formula = Direction ~ polym(Year + Lag1 + Lag2 + Lag3 + Lag4 +
##     Lag5 + Volume + Today, degree = 4), family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1194  -1.2418   0.8938   1.0985   2.3283
##
## Coefficients:
##                                                                                 Estimate
## (Intercept)                                                                      0.22999
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)1 12.20342
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)2 -0.08086
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)3  5.77411
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)4 -4.64509
##                                                                                 Std. Error
## (Intercept)                                                                        0.06254
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)1    2.31050
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)2    2.82849
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)3    3.00031
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)4    3.18776
##                                                                                 z value
## (Intercept)                                                                       3.677
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)1   5.282
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)2  -0.029
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)3   1.925
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)4  -1.457
##                                                                                 Pr(>|z|)
## (Intercept)                                                                     0.000236
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)1 1.28e-07
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)2 0.977192
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)3 0.054291
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)4 0.145070
##
```

```
## (Intercept)                                                                    ***
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)1 ***
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)2
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)3 .
## polym(Year + Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, degree = 4)4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1458.6  on 1083  degrees of freedom
## AIC: 1468.6
##
## Number of Fisher Scoring iterations: 4
```

AIC increased, hence polynomial of degree 3 is best fit for logistic regression

```
pred_lr3 = predict(lr_model3, test)
pred_lr3_cls = ifelse(pred_lr3<0.5, 'Down', 'Up')
table(pred_lr3_cls, test$Direction)
```

```
##
## pred_lr3_cls Down Up
##         Down   20  0
##         Up      0 32
```

```
mean(pred_lr3_cls == test$Direction)
```

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

```
mean(pred_lr3_cls != test$Direction)
```

```
## [1] 0
```

Test error = 0.46 Sensitivity = 295/(295+157) = 0.652 Specificity = 249/(249+310) = 0.445

Different LDA models

```
lda_model2 <- lda(train$Direction~(Year+Lag1+Lag2+Lag3+Lag4+Lag5+Volume+Today)^2,
                  data=train)
pred_class <- predict(lda_model2, test)$class
table(pred_class, test$Direction)
```

```
##
## pred_class Down Up
##       Down   17  0
##       Up      3 32
```

accuracy = (293+409)/(293+159+150+409) = 0.694 test error = (150+159)/(293+159+150+409) = 0.3056
Sensitivity = 293/(293+159) = 0.6482 Specificity = 409/(150+409) = 0.7316

Problem 3

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

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

```
library(ISLR)
attach(Auto)
data <- data.frame(Auto)
data$mpg01 <- as.integer(mpg > median(mpg))
head(data) # since median = 22.75
```

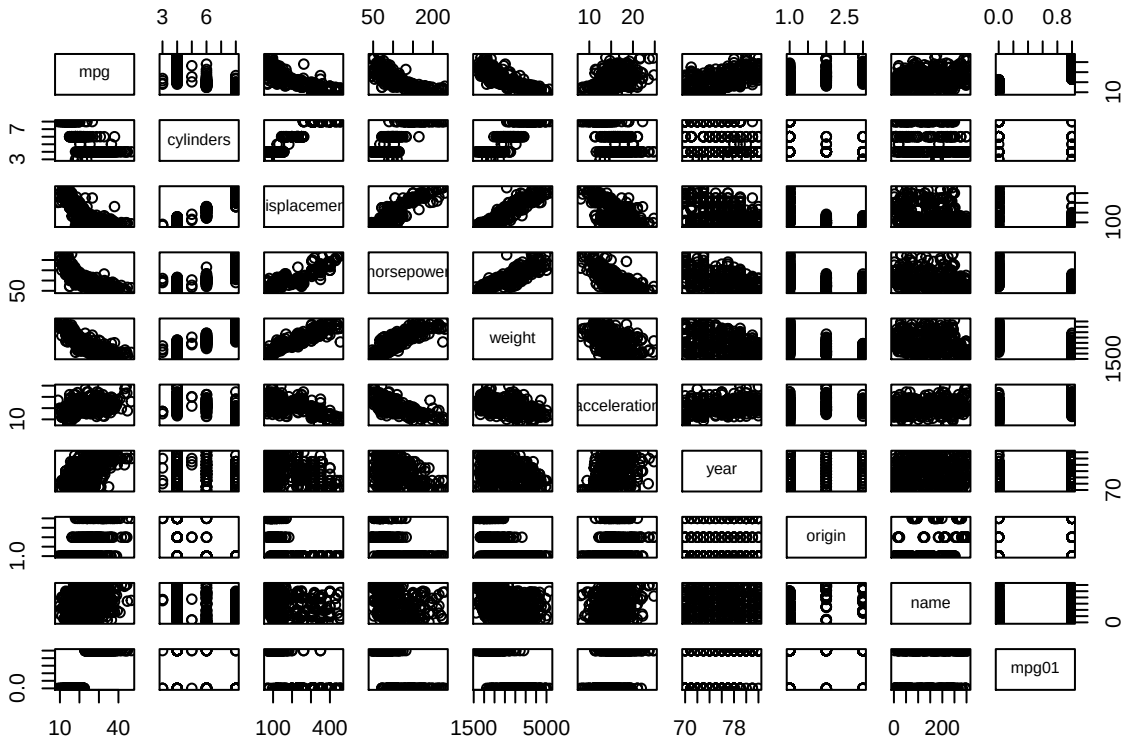
```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                       name mpg01
## 1 chevrolet chevelle malibu     0
## 2         buick skylark 320     0
## 3        plymouth satellite     0
## 4             amc rebel sst     0
## 5               ford torino     0
## 6          ford galaxie 500     0
```

```
tail(data)
```

```
##     mpg cylinders displacement horsepower weight acceleration year origin
## 392  27         4          151         90   2950         17.3   82      1
## 393  27         4          140         86   2790         15.6   82      1
## 394  44         4           97         52   2130         24.6   82      2
## 395  32         4          135         84   2295         11.6   82      1
## 396  28         4          120         79   2625         18.6   82      1
## 397  31         4          119         82   2720         19.4   82      1
##                  name mpg01
## 392 chevrolet camaro     1
## 393  ford mustang gl     1
## 394        vw pickup     1
## 395    dodge rampage     1
## 396      ford ranger     1
## 397       chevy s-10     1
```

(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(data)
```



Since mpg01 depends on mpg, which in turn depends on Displacement, horsepower, weight, acceleration and year, therefore, these could be significant explanatory variables

```
attach(data)
```

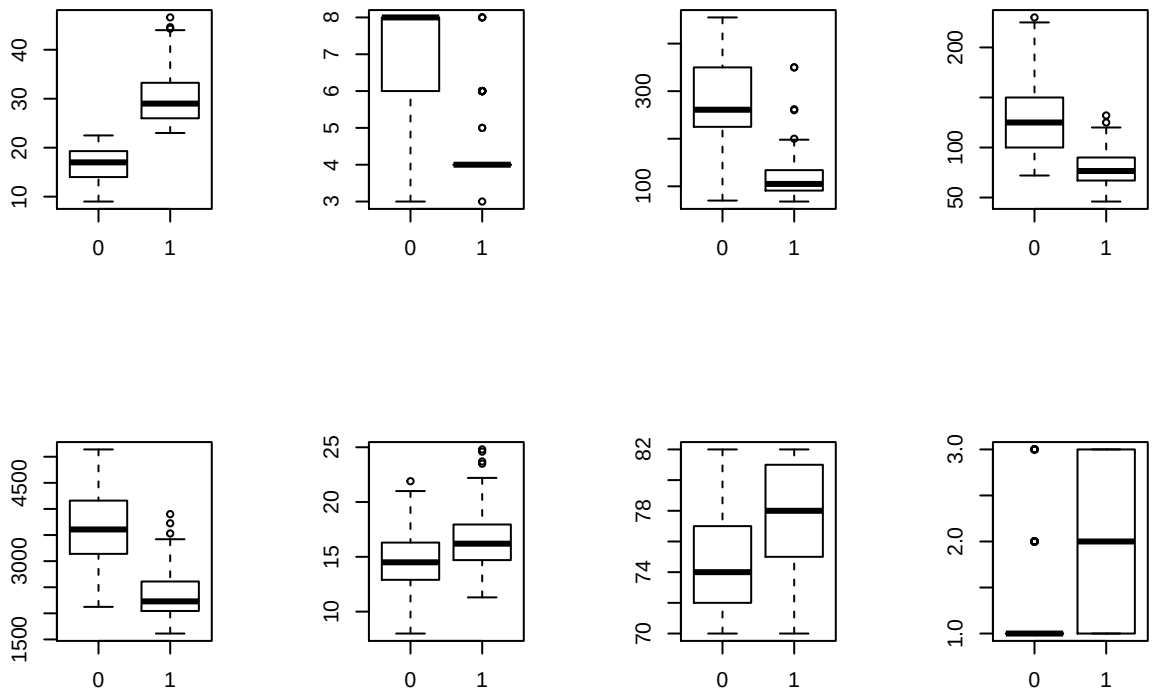
```
## The following objects are masked from Auto:
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year
```

```
names(data)
```

```
##  [1] "mpg"          "cylinders"    "displacement" "horsepower"
##  [5] "weight"       "acceleration" "year"         "origin"
##  [9] "name"         "mpg01"
```

```
par(mfrow = c(2,4))
boxplot(data$mpg~data$mpg01,data=data)
boxplot(data$cylinders~data$mpg01,data=data)
boxplot(data$displacement~data$mpg01,data=data)
boxplot(data$horsepower~data$mpg01,data=data)
```

```
boxplot(data$weight~data$mpg01,data=data)
boxplot(data$acceleration~data$mpg01,data=data)
boxplot(data$year~data$mpg01,data=data)
boxplot(data$origin~data$mpg01,data=data)
```



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

```
n = dim(data)[1]
indx = sample.int(n, floor(0.20*n))
train = data[-indx,]
test = data[indx,]
```

Random 80-20 spliting done

(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?

Since Displacement, horsepower, weight, acceleration and year seems most effective, I am fitting a model using these variables.

```
attach(data)
```

```
## The following objects are masked from data (pos = 3):
```

```
##
##     acceleration, cylinders, displacement, horsepower, mpg, mpg01,
##     name, origin, weight, year
```

```
## The following objects are masked from Auto:
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year
```

```
lda_fit <- lda(mpg01 ~ displacement+horsepower+weight+acceleration+year, data = train)
summary(lda_fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   10     -none- numeric
## scaling  5     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
pred_lda = predict(lda_fit, test)$class
table(pred_lda, test$mpg01)
```

```
##
## pred_lda  0  1
##        0 38  1
##        1  7 32
```

```
mean(pred_lda == test$mpg01)
```

```
## [1] 0.8974359
```

```
mean(pred_lda != test$mpg01)
```

```
## [1] 0.1025641
```

LDA is a very good fit for the model. Test error is = 0.1282 Sensitivity = 36/(36+9) = 0.8 Specificity = 32/(32+1) = 0.9696

(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 ~ displacement+horsepower+weight+acceleration+year, data = train)
summary(lda_fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   10     -none- numeric
## scaling  5     -none- numeric
## lev      2     -none- character
```

```
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
pred_qda = predict(qda_fit, test)$class
table(pred_qda, test$mpg01)
```

```
##
## pred_qda  0  1
##        0 40  3
##        1  5 30
```

```
mean(pred_qda == test$mpg01)
```

```
## [1] 0.8974359
```

```
mean(pred_qda != test$mpg01)
```

```
## [1] 0.1025641
```

Test error = 0.1025 Accuracy = 0.8974 sensitivity = 38/(38+7) = 0.844 Specificity = 32/(1+32) = 0.9696

(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?

```
lr_fit <- glm(mpg01 ~ displacement+horsepower+weight+acceleration+year, data = train,
              family = binomial)
summary(lr_fit)
```

```
##
## Call:
## glm(formula = mpg01 ~ displacement + horsepower + weight + acceleration +
##     year, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2270  -0.1267   0.0320   0.2274   3.1750
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -16.020127   6.226682  -2.573 0.010087 *
## displacement  -0.003236   0.007405  -0.437 0.662112
## horsepower    -0.038560   0.024869  -1.551 0.121010
## weight        -0.004289   0.001244  -3.448 0.000565 ***
## acceleration  -0.042437   0.151516  -0.280 0.779416
## year           0.435522   0.083901   5.191 2.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 434.84  on 313  degrees of freedom
```

```
## Residual deviance: 128.65  on 308  degrees of freedom
## AIC: 140.65
##
## Number of Fisher Scoring iterations: 7
```

```
pred_lr = predict(lr_fit, test)
pred_lr_cls = ifelse(pred_lr<0.5, 0, 1)
table(pred_lr_cls, test$mpg01)
```

```
##
## pred_lr_cls  0  1
##           0 40  2
##           1  5 31
```

```
mean(pred_lr_cls == test$mpg01)
```

```
## [1] 0.9102564
```

```
mean(pred_lr_cls != test$mpg01)
```

```
## [1] 0.08974359
```

Test error = 0.1282 Accuracy = 0.8717 sensitivity = 37/(37+8) = 0.822 Specificity = 31/(31+2) = 0.9393

(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?

```
knn_fit = knn(train[3:7], test[3:7], train$mpg01, k=1)
table(knn_fit,test$mpg01)
```

```
##
## knn_fit  0  1
##       0 38  4
##       1  7 29
```

```
mean(knn_fit == test$mpg01)
```

```
## [1] 0.8589744
```

```
mean(knn_fit != test$mpg01)
```

```
## [1] 0.1410256
```

Test error = 0.1025 Accuracy = 0.8974 sensitivity = 39/(39+6) = 0.866 Specificity = 31/(31+2) = 0.9393

```
knn_fit = knn(train[3:7], test[3:7], train$mpg01, k=2)
table(knn_fit,test$mpg01)
```

```
##
## knn_fit  0  1
##       0 39  4
##       1  6 29
```

```
mean(knn_fit == test$mpg01)
```

```
## [1] 0.8717949
```

```
mean(knn_fit != test$mpg01)
```

```
## [1] 0.1282051
```

```
knn_fit = knn(train[3:7], test[3:7], train$mpg01, k=3)
table(knn_fit,test$mpg01)
```

```
##
## knn_fit  0  1
##       0 41  2
##       1  4 31
```

```
mean(knn_fit == test$mpg01)
```

```
## [1] 0.9230769
```

```
mean(knn_fit != test$mpg01)
```

```
## [1] 0.07692308
```

```
knn_fit = knn(train[3:7], test[3:7], train$mpg01, k=4)
table(knn_fit,test$mpg01)
```

```
##
## knn_fit  0  1
##       0 39  4
##       1  6 29
```

```
mean(knn_fit == test$mpg01)
```

```
## [1] 0.8717949
```

```
mean(knn_fit != test$mpg01)
```

```
## [1] 0.1282051
```

```
knn_fit = knn(train[3:7], test[3:7], train$mpg01, k=5)
table(knn_fit,test$mpg01)
```

```
##
## knn_fit  0  1
##       0 40  4
##       1  5 29
```

```
mean(knn_fit == test$mpg01)
```

```
## [1] 0.8846154
```

```
mean(knn_fit != test$mpg01)
```

```
## [1] 0.1153846
```

Clearly, k=1 gives the best result