Homework 3 Chapter 4:

Logistic Regression: Exercise 13 (complete exercise)

==============================

13. This question should be answered using the Weekly data set, which is part of the ISLR2 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.

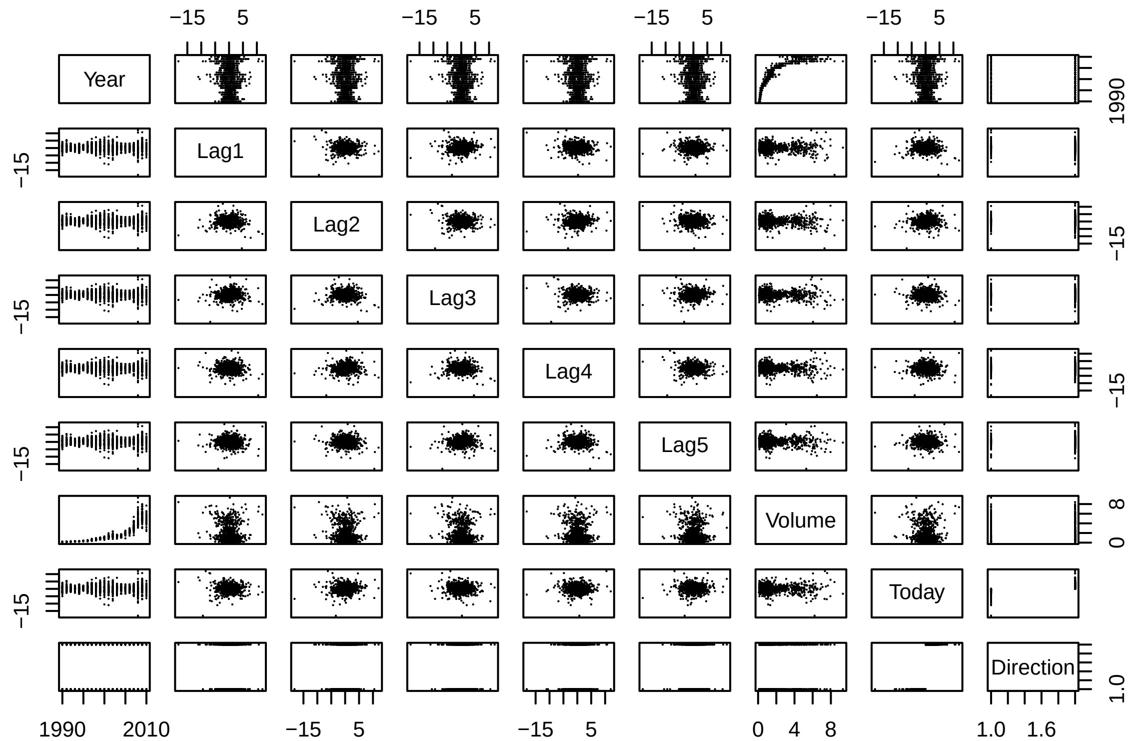
(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

The `summary(Weekly)` command produces numerical summaries, and `plot(Weekly)` function produces the graphical summary (pairwise scatterplots) of all variables in the data. There are no immediately obvious patterns, except maybe that volume seems to have increased in the last few years.

```r
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
##
##
##
##
```

```r
plot(Weekly, cex=0.01)
```



(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?

Only one predictor, `Lag2`, appears significant at 0.05% significance level:

```
# logistic regression
attach(Weekly)
DirectionUp = as.numeric(Direction=="Up")
glm.fit = glm(DirectionUp ~ Volume+Lag1+Lag2+Lag3+Lag4+Lag5,
              data=Weekly, family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = DirectionUp ~ Volume + Lag1 + Lag2 + Lag3 + Lag4 +
##     Lag5, 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 **
## Volume      -0.02274    0.03690  -0.616   0.5377
## 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
## ---
## 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
```

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

```
glm.probs = predict(glm.fit, type="response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != DirectionUp)
```

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

```
table(glm.pred, DirectionUp)
```

```
##         DirectionUp
## glm.pred   0   1
##        0  54  48
##        1 430 557
```

The overall misclassification error rate is 43.89% which can be computed as the number of misclassifications (48+430) divided by the total data sample size (1089). The complement of that is the overall fraction of correct predictions (1-43.89% = 56.11%).

The confusion matrix tells us that 48 times the model wrongly predicted that the market would go down while in reality it went up; and 430 times it wrongly predicted that the market would go up but it went down. The model however correctly predicted that predicted that the market would go up on 557 days, and that it would go down on 54 days, for a total of $557 + 54 = 611$ correct predictions. The `mean()` function can be used to compute the fraction of days for which the prediction was correct. In this case, logistic regression correctly predicted the movement of the market in 611 out of 1089 days, which is 56.11% 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.**

```
train = (Year <= 2008)
Weekly.train = Weekly[train , ]
DirUp.train = DirectionUp[train]
Weekly.test = Weekly[!train, ]
DirUp.test = DirectionUp[!train]

glm.train.fit = glm(DirectionUp ~ Lag2, subset=train, family=binomial)
summary(glm.train.fit)
```

```
##
## Call:
## glm(formula = DirectionUp ~ Lag2, family = binomial, subset = train)
```

```
##
## 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
```

```r
# glm.train.fit = glm(DirUp.train ~ Lag2[train], family=binomial)
# summary(glm.train.fit)
```

Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```r
glm.test.probs = predict(glm.train.fit, Weekly.test, type="response")
glm.test.pred = rep(0, length(glm.test.probs))
glm.test.pred[glm.test.probs > 0.5] = 1
mean(glm.test.pred != DirUp.test)
```

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

```r
table(glm.test.pred, DirUp.test)
```

```
##              DirUp.test
## glm.test.pred  0  1
##             0  9  5
##             1 34 56
```

The overall misclassification error rate is 37.5% which can be computed as the number of misclassifications (34+5) divided by the total test data sample size (104). The complement of that is the overall fraction of correct predictions (1-37.5% = 62.5%).

The confusion matrix tells us that 5 days the model wrongly predicted that the market would go down while in reality it went up; and 34 times it wrongly predicted that the market would go up but it really went down. The model however correctly predicted that predicted that the market would go up on 56 days, and that it would go down on 9 days, for a total of 56 + 9 = 65 correct predictions. The `mean()` function can be used to compute the fraction of days for which the prediction was correct. In this case, logistic regression correctly predicted the movement of the market in 65 out of 104 days, which is 62.5% of the time.

Beginning of HW 4

(e) Repeat (d) using LDA.

```r
# LDA
library(MASS)
```

```
lda.train.fit = lda(DirectionUp ~ Lag2, subset=train)
lda.train.fit
```

```
## Call:
## lda(DirectionUp ~ Lag2, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4477157 0.5522843
##
## Group means:
##          Lag2
## 0 -0.03568254
## 1  0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```
lda.pred = predict(lda.train.fit, Weekly.test)
mean(lda.pred$class != DirUp.test)
```

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

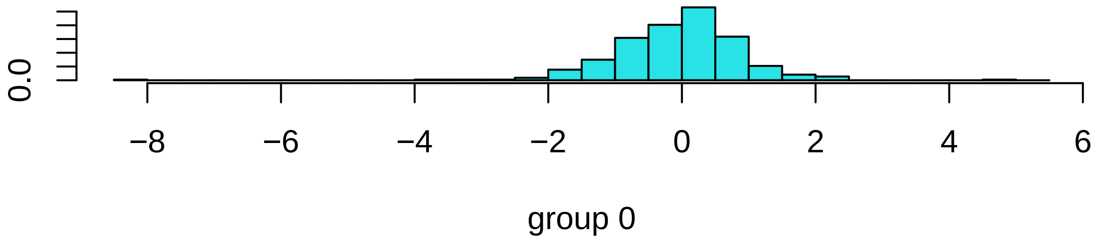
```
table(lda.pred$class, DirUp.test)
```

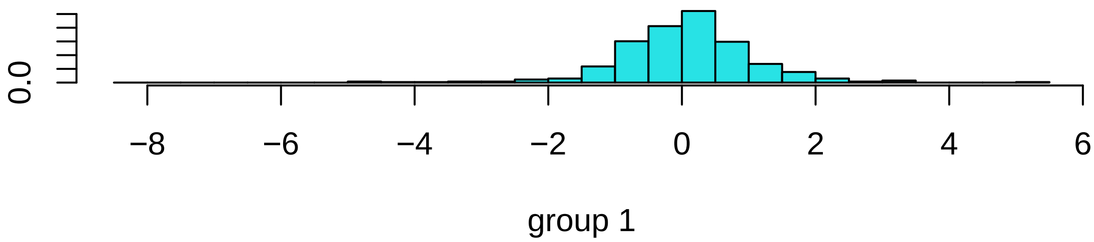
```
##    DirUp.test
##      0  1
##   0  9  5
##   1 34 56
```

Using the 50% threshold, the overall misclassification error rate is 37.5%, identical to the logistic regression misclassification rate. LDA's misclassification rate can be computed as the number of misclassifications (5+34) divided by the total test data sample size (104). The complement of that is the overall fraction of correct predictions (1-37.5% = 62.5%).

As in the case of the logistic regression, the confusion matrix tells us that 5 days the LDA model wrongly predicted that the market would go down while in reality it went up; and 34 times it wrongly predicted that the market would go up but it really went down. The model however correctly predicted that predicted that the market would go up on 56 days, and that it would go down on 9 days, for a total of 56 + 9 = 65 correct predictions. The `mean()` function can be used to compute the fraction of days for which the prediction was correct. In this case, LDA correctly predicted the movement of the market in 65 out of 104 days, which is 62.5% of the time.

```
plot(lda.train.fit)
```





(f) Repeat (d) using QDA.

```
qda.train.fit = qda(DirectionUp ~ Lag2, subset=train)
qda.train.fit
```

```
## Call:
## qda(DirectionUp ~ Lag2, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4477157 0.5522843
##
## Group means:
##          Lag2
## 0 -0.03568254
## 1  0.26036581
```

```
qda.pred = predict(qda.train.fit, Weekly.test)
mean(qda.pred$class != DirUp.test)
```

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

```
table(qda.pred$class, DirUp.test)
```

```
##    DirUp.test
##      0  1
##   0  0  0
##   1 43 61
```

The overall misclassification error rate for QDA is 41.35%, slightly higher than LDA and the logistic regression misclassification rate. QDA's misclassification rate can be computed as the number of misclassifications (0+43) divided by the total test data sample size (104). The complement of that is the overall fraction of correct predictions (1-41.35% = 58.65%).

As in the case of the logistic regression, the confusion matrix tells us that 0 times the LDA model wrongly predicted that the market would go down while in reality it went up; and 43 times it wrongly predicted that the market would go up but it really went down. The model however correctly predicted that predicted that the market would go up on 61 days, and that it would go down on 0 days, for a total of 61 correct predictions. The `mean()` function can be used to compute the fraction of days for which the prediction was correct. In this case, QDA correctly predicted the movement of the market in 61 out of 104 days, which is 58.65% of the time.

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

```
# KNN
library(class)
train.X = cbind(Lag2[train])
test.X = cbind(Lag2[!train])
train.DirUp = DirUp.train
set.seed(1)
# KNN(k=1)
knn.pred = knn(train.X, test.X, train.DirUp, k=1)
mean(knn.pred != DirUp.test)
```

```
## [1] 0.5
```

```
table(knn.pred,DirUp.test)
```

```
##         DirUp.test
## knn.pred  0  1
##        0 21 30
##        1 22 31
```

50.0% test error rate.

(h) Repeat (d) using naive Bayes.

```
#install.packages("e1071")
library(e1071)

nb.fit = naiveBayes(DirectionUp ~ Lag2, data=Weekly, subset=train)
nb.fit
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.4477157 0.5522843
##
## Conditional probabilities:
##    Lag2
## Y          [,1]     [,2]
```

```
##   0 -0.03568254 2.199504
##   1  0.26036581 2.317485
```

```
nb.class <- predict(nb.fit, Weekly.test)
mean(nb.class != DirUp.test)
```

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

```
table(nb.class, DirUp.test)
```

```
##         DirUp.test
## nb.class  0  1
##        0  0  0
##        1 43 61
```

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

LDA was the best (37.5% misclassification rate); QDA and NB were second best (41% misclassification rate); KNN had the worst misclassification rate, 50%.

(j) 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.

This is an open question, and any answer will suffice - as long as there is one additional model per method (so one extra model for LDA, QDA, NB, and KNN). Note that the model has to be consistent across different methods (LDA, QDA, NB and KNN) for the comparison to make sense.

Examples of tweaks to try:

```
# KNN(k=20), scaled
train.X = scale(cbind(Lag2[train]))
test.X = scale(cbind(Lag2[!train]))
train.DirUp = DirUp.train

knn.pred = knn(train.X, test.X, train.DirUp, k=20)
mean(knn.pred != DirUp.test)
```

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

```
table(knn.pred,DirUp.test)
```

```
##         DirUp.test
## knn.pred  0  1
##        0 17 18
##        1 26 43
```

```
# KNN(k=10)
knn.pred = knn(train.X, test.X, train.DirUp, k=10)
mean(knn.pred != DirUp.test)
```

```
## [1] 0.4038462
```

```
table(knn.pred,DirUp.test)
```

```
##         DirUp.test
## knn.pred  0  1
##        0 19 18
##        1 24 43
```

```r
# KNN(k=20)
knn.pred = knn(train.X, test.X, train.DirUp, k=20)
mean(knn.pred != DirUp.test)
```

```
## [1] 0.4519231
```

```r
table(knn.pred,DirUp.test)
```

```
##          DirUp.test
## knn.pred  0  1
##         0 17 21
##         1 26 40
```