Homework 02

Mingxue (Jacqueline) Li

2019-02

Exercises from Section 4.7 of ISLR

Exercise 7

Suppose that we wish to predict whether a given stock will issue a dividend this year ('Yes' or 'No') based on X, last year's percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\bar{X} = 10$, while the mean for those that didn't was $\bar{X} = 0$. In addition, the variance of X for these two sets of companies was $\hat{\sigma}^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was $X = 4$ last year.

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

Answers:

According to Bayes' theorem,

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

along with the normal density formula,

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

we can compute:

$$p_{yes}(x) = \frac{\pi_{yes} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu_{yes})^2/2\sigma^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu_l)^2/2\sigma^2}} = \frac{\pi_{yes} e^{-(x-\mu_{yes})^2/2\sigma^2}}{\pi_{yes} e^{-(x-\mu_{yes})^2/2\sigma^2} + \pi_{no} e^{-(x-\mu_{no})^2/2\sigma^2}}$$

$$p_{yes}(4) = \frac{0.8 e^{-(4-10)^2/2*36}}{0.8 e^{-(4-10)^2/2*36} + 0.2 e^{-(4-0)^2/2*36}} \approx 75.2\%$$

Exercise 10

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.

Solutions:

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

```
library(ISLR)
dim(Weekly)
```

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

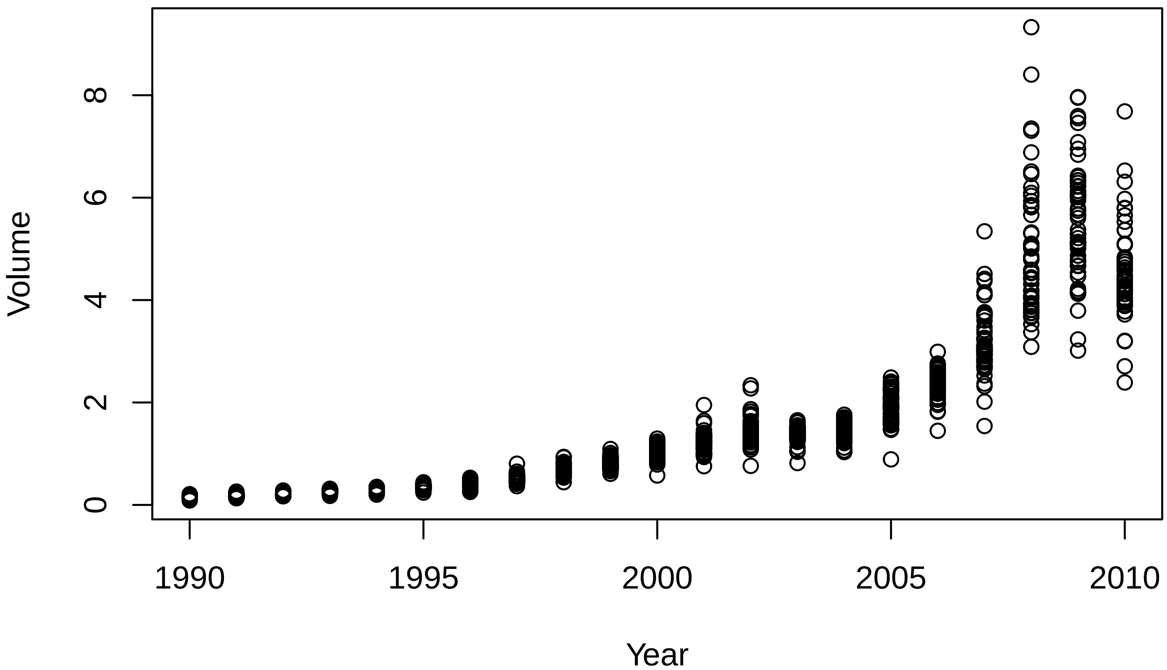
```r
summary(Weekly)
```

```
##       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
data = Weekly
cor(data[,-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
```

```r
attach(Weekly)
plot(Year, Volume)
```



Conclusions: `Year` and `Volume` are highly positively correlated.

(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?

```
glm.fit1 = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data = Weekly, family = binomial)
summary(glm.fit1)
```

```
##
## 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
```

Conclusions: Lag2 is proven to be statistically significant.

(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.probs1 = predict(glm.fit1, type = "response")
glm.pred1 = rep("Down", 1089)
glm.pred1[glm.probs1 > 0.5] = "Up"
table(glm.pred1, Direction)
```

```
##          Direction
## glm.pred1 Down  Up
##      Down   54  48
##      Up    430 557
```

```
mean(glm.pred1==Direction)
```

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

Conclusions: This logistic regression correctly predicted the movement of the returns 56.11% of the time. The false positive rate (Type I Error) of this regression is 430/(430+54) = 88.84% and the true positive rate (Type II Error) of this regression is 557/(557+48) = 92.07%, meaning that the logstic regression is right about returns going up 92.07% of the time and is right about the returns going down for only 54/(430+54)=11.16% 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 the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
train = (Year < 2009)
test =  Weekly[!train, ]
glm.fit2 = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
glm.probs2 = predict(glm.fit2, test, type = 'response')
glm.pred2 = rep('Down', dim(test)[1])
glm.pred2[glm.probs2 > 0.5] = 'Up'
table(glm.pred2, Direction[!train])
```

```
##
## glm.pred2 Down Up
##      Down    9  5
##      Up     34 56
```

```
mean(glm.pred2==Direction[!train])
```

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

(e) Repeat (d) using LDA.

```
library(MASS)
lda.fit1 = lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.pred1 = predict(lda.fit1, test)
names(lda.pred1)
```

```
## [1] "class"     "posterior" "x"
```

```r
table(lda.pred1$class, Direction[!train])
```

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

```r
mean(lda.pred1$class==Direction[!train])
```

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

(f) Repeat (d) using QDA.

```r
qda.fit1 = qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.pred1 = predict(qda.fit1, test)
names(qda.pred1)
```

```
## [1] "class"     "posterior"
```

```r
table(qda.pred1$class, Direction[!train])
```

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

```r
mean(qda.pred1$class==Direction[!train])
```

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

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

```r
library(class)
train.X = as.matrix(Lag2[train])
test.X = as.matrix(Lag2[!train])
train.Direction = Direction[train]
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction[!train])
```

```
##
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```r
mean(knn.pred==Direction[!train])
```

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

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

LDA and Logistic Regression have the best results based on the proportion of observations that are correctly predicted.

Exercise 11

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.

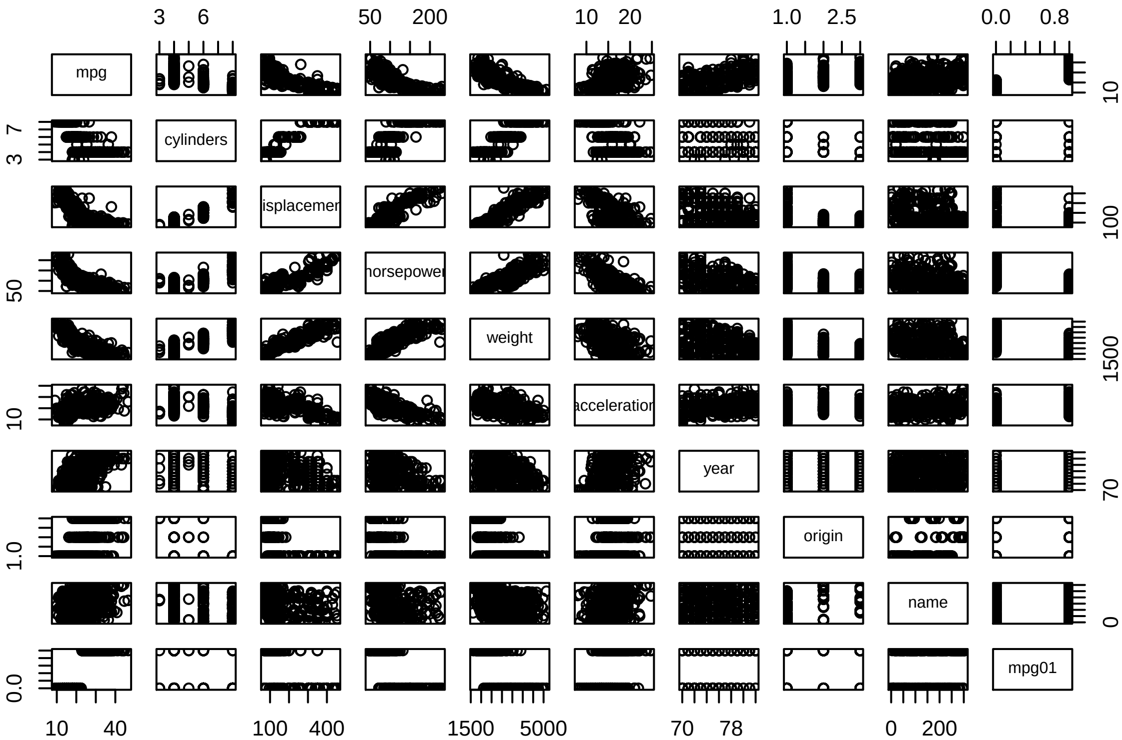
Solutions:

(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 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)
mpg01 = ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
df = 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(df)
```




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

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
```

```
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

Conclusions: `mpg01` is highly correlated with `cylinders`, `displacement`, `horsepower` and `weight`.

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

```
train = df[1:200,]
test = df[201:392,]
```

(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?

```
lda.fit = lda(mpg01 ~ cylinders+displacement+horsepower+weight, data = train)
lda.pred = predict(lda.fit, test)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
table(lda.pred$class, test$mpg01)
```

```
##
##       0   1
##   0  56  12
##   1   8 116
```

```
mean(lda.pred$class!=test$mpg01)
```

```
## [1] 0.1041667
```

Conclusions: The test error of the model obtained is about 10.42%.

(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+displacement+horsepower+weight, data = train)
qda.pred = predict(qda.fit, test)
names(qda.pred)
```

```
## [1] "class"     "posterior"
```

```
table(qda.pred$class, test$mpg01)
```

```
##
##       0   1
##   0  60  22
##   1   4 106
```

```
mean(qda.pred$class!=test$mpg01)
```

```
## [1] 0.1354167
```

Conclusions: The test error of the model obtained is about 13.54%.

(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?

```
glm.fit = glm(mpg01 ~ cylinders+displacement+horsepower+weight, data=train, family=binomial)
glm.probs = predict(glm.fit, test, type="response")
```

```r
glm.pred = ifelse(glm.probs > 0.5, 1, 0)
table(glm.pred, test$mpg01)
```

```
##
## glm.pred  0  1
##        0 61 36
##        1  3 92
```

```r
mean(glm.pred!=test$mpg01)
```

```
## [1] 0.203125
```

Conclusions: The test error of the model obtained is about 20.31%.

(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?

```r
set.seed(1)
train.X = cbind(train$cylinders, train$weight, train$displacement, train$horsepower)
test.X = cbind(test$cylinders, test$weight, test$displacement, test$horsepower)
knn.pred = knn(train.X, test.X, train$mpg01, k=1)
table(knn.pred, test$mpg01)
```

```
##
## knn.pred   0   1
##        0  61  28
##        1   3 100
```

```r
mean(knn.pred != test$mpg01)
```

```
## [1] 0.1614583
```

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train$mpg01, k=5)
table(knn.pred, test$mpg01)
```

```
##
## knn.pred   0   1
##        0  61  24
##        1   3 104
```

```r
mean(knn.pred != test$mpg01)
```

```
## [1] 0.140625
```

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train$mpg01, k=10)
table(knn.pred, test$mpg01)
```

```
##
## knn.pred   0   1
##        0  61  27
##        1   3 101
```

```r
mean(knn.pred != test$mpg01)
```

```
## [1] 0.15625
```

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train$mpg01, k=20)
table(knn.pred, test$mpg01)
```

```
##
## knn.pred   0   1
##        0  61  25
##        1   3 103
```

```r
mean(knn.pred != test$mpg01)
```

```
## [1] 0.1458333
```

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train$mpg01, k=50)
table(knn.pred, test$mpg01)
```

```
##
## knn.pred  0  1
##        0 62 37
##        1  2 91
```

```r
mean(knn.pred != test$mpg01)
```

```
## [1] 0.203125
```

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train$mpg01, k=100)
table(knn.pred, test$mpg01)
```

```
##
## knn.pred   0   1
##        0  61  24
##        1   3 104
```

```r
mean(knn.pred != test$mpg01)
```

```
## [1] 0.140625
```

Conclusions: $K = 5$ and$K = 100$ seem to perform the best on this data set.