Classification Ch. 4 Exercises

Lucy L.

2024-01-24

```
library(ISLR2)
library(MASS)
library(e1071)
library(class)
set.seed(1)
```

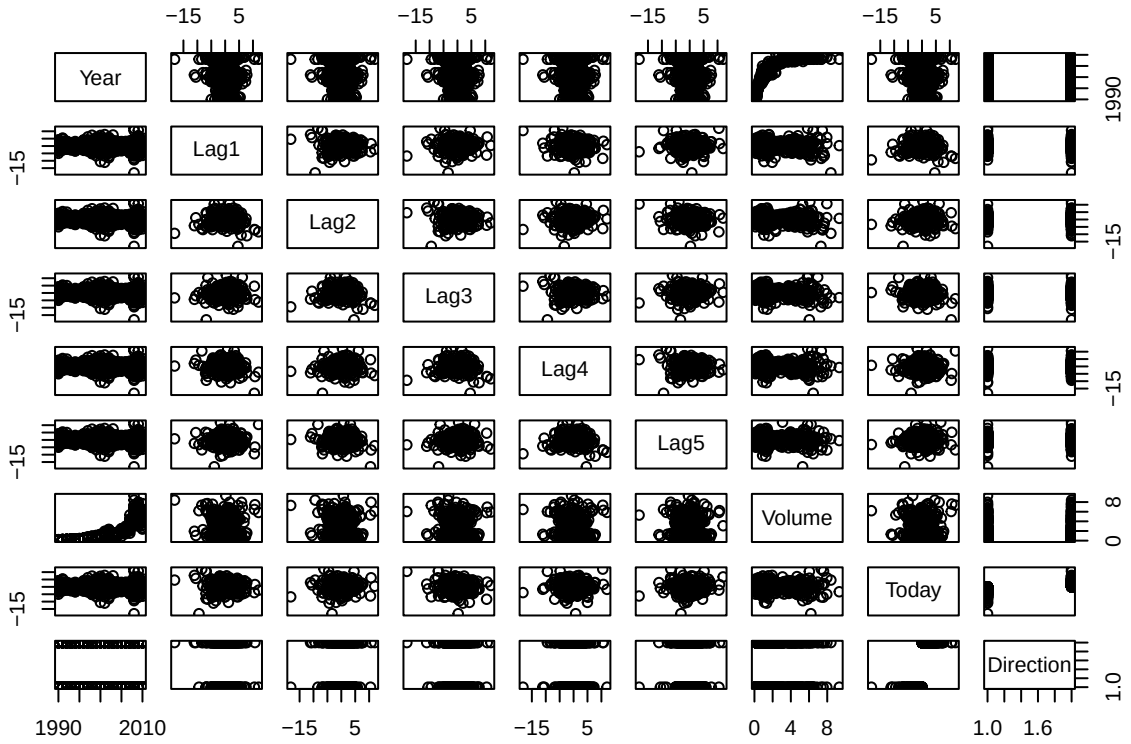
Exercise 13

(a)

```
weekly = Weekly
# names(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            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)
```



(b)

```r
# Fitting logistic regression model
glm.fits = glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Weekly, family = binomial
)

summary(glm.fits)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## 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 following predictors are statistically significant: Lag2.

(c)

```
attach(Weekly)

# Creating confusion matrix for the outcomes
glm.probs = predict(glm.fits, type = "response")
glm.pred1 = rep("Down", 1089)
glm.pred1[glm.probs > 0.5] = "Up"
table(glm.pred1, Direction)
```

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

```
1 - mean(glm.pred1 == Direction) # test error
```

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

The overall fraction of correct predictions is about 0.564, which is favorable. The confusion matrix indicates that logistic regression is unrealistically optimistic because the entire data were used as a training set.

(d)

```
# Creating training and test sets
train = subset(weekly, subset = Year <= 2008 & Year >= 1990)
weekly_2009_2010 = subset(weekly, subset = Year >= 2009)
dim(weekly_2009_2010)
```

```
## [1] 104   9
```

```
direction_2009_2010 = weekly_2009_2010$Direction

glm.fits2 = glm(Direction ~ Lag2, data = train, family = binomial)

# Creating confusion matrix for the outcomes
glm.probs2 = predict(glm.fits2, weekly_2009_2010, type = "response")
glm.pred2 = rep("Down", 104)
glm.pred2[glm.probs2 > 0.5] = "Up"
table(glm.pred2, direction_2009_2010)
```

```
##           direction_2009_2010
## glm.pred2 Down Up
##      Down    9  5
##      Up     34 56
```

```
1- mean(glm.pred2 == direction_2009_2010) # test error
```

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

(e) Perform the same as (d) using LDA.

```
lda.fit = lda(Direction ~ Lag2, data = train)
lda.pred = predict(lda.fit, weekly_2009_2010)

lda.class = lda.pred$class
table(lda.class, weekly_2009_2010$Direction)
```

```
##
## lda.class Down Up
##      Down    9  5
##      Up     34 56
```

```
mean(lda.class == weekly_2009_2010$Direction)
```

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

(f) Repeat (d) using QDA.

```
qda.fit = qda(Direction ~ Lag2, data = train)
qda.pred = predict(qda.fit, weekly_2009_2010)

qda.class = qda.pred$class
table(qda.class, weekly_2009_2010$Direction)
```

```
##
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```
mean(qda.class == weekly_2009_2010$Direction)
```

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

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

```
train.X = data.frame(train$Lag2)
test.X = data.frame(weekly_2009_2010$Lag2)
train.Direction = train$Direction

# k = 1
knn.pred = knn(train.X, test.X, train.Direction, k=1)
table(knn.pred, weekly_2009_2010$Direction)
```

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

```
mean(knn.pred == weekly_2009_2010$Direction)
```

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

(h) Repeat (d) using Naive Bayes.

```
nb.fit = naiveBayes(Direction ~ Lag2, data = train)
nb.class = predict(nb.fit, weekly_2009_2010)
table(nb.class, weekly_2009_2010$Direction)
```

```
##
## nb.class Down Up
##     Down    0  0
##     Up     43 61
```

```
mean(nb.class == weekly_2009_2010$Direction)
```

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

(i) For this dataset, LDA has the best results, followed by logistic regression, QDA and Naive Bayes, and finally KNN (with K = 1).

Exercise 14

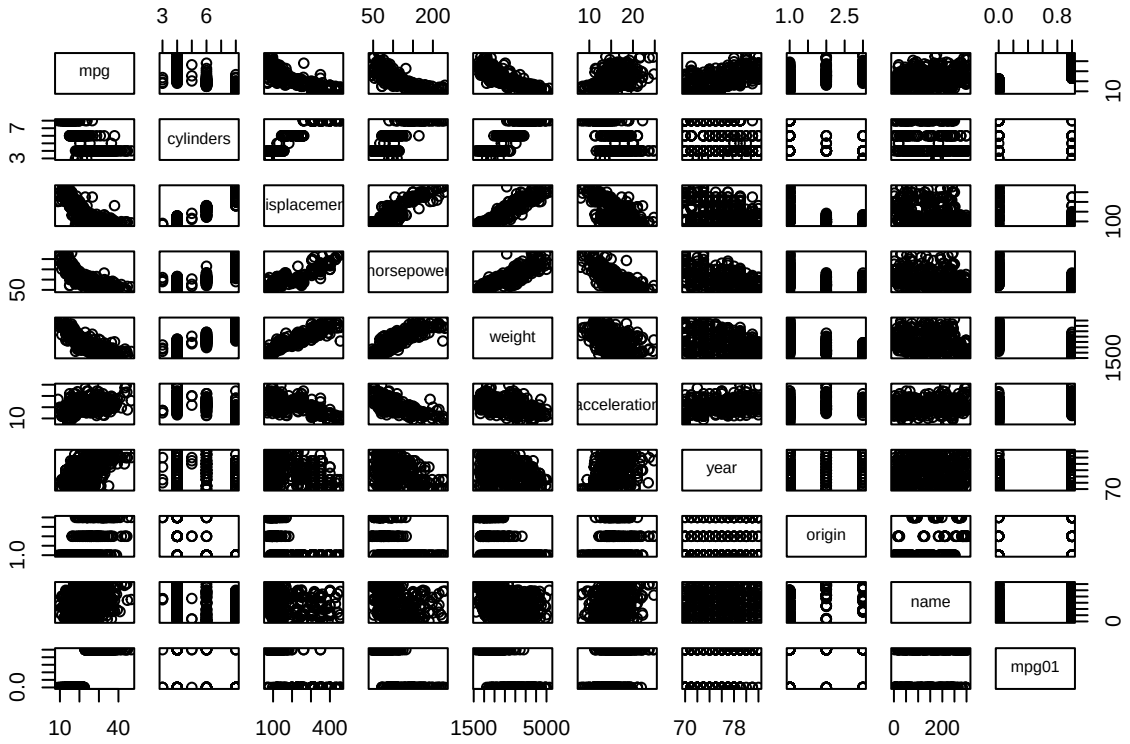
```
attach(Auto)
```

(a)

```
Auto$mpg01 <- ifelse(Auto$mpg > median(mpg), 1, 0)
```

(b)

```
pairs(Auto)
```



Displacement, horsepower and weight appear to be the most likely to be efficient for predicting `mpg01`.

(c)

```
# split into training and test sets
train_num = 1:300
train = Auto[train_num, ]
test = Auto[-train_num, ]
```

(d)

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

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

```
##
## lda.class  0  1
##         0  5  9
##         1  0 78
```

```r
mean(lda.class == test$mpg01)
```

```
## [1] 0.9021739
```

```r
(1 - mean(lda.class == test$mpg01)) # test error
```

```
## [1] 0.09782609
```

(e)

```r
qda.fit = qda(mpg01 ~  displacement + horsepower + weight,
              data = train)
qda.pred = predict(qda.fit, test)

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

```
##
## qda.class  0  1
##         0  5 14
##         1  0 73
```

```r
mean(qda.class == test$mpg01)
```

```
## [1] 0.8478261
```

```r
(1 - mean(qda.class == test$mpg01)) # test error
```

```
## [1] 0.1521739
```

(f)

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

```
## [1] 92 10
```

```r
glm.pred1 = rep(0, 92)
glm.pred1[glm.probs > 0.5] = 1
table(glm.pred1, test$mpg01)
```

```
##
## glm.pred1  0  1
##         0  5 18
##         1  0 69
```

```
1 - mean(glm.pred1 == test$mpg01) # test error
```

```
## [1] 0.1956522
```

(g)

```
nb.fit = naiveBayes(mpg01 ~  displacement + horsepower + weight,
              data = train)
nb.class = predict(nb.fit, test)
table(nb.class, test$mpg01)
```

```
##
## nb.class  0  1
##        0  5  9
##        1  0 78
```

```
1 - mean(nb.class == test$mpg01) # test error
```

```
## [1] 0.09782609
```

(h)

```
train.X = data.frame(train$horsepower, train$displacement, train$weight)
test.X = data.frame(test$horsepower, test$displacement, test$weight)
train.Y = train$mpg01
test.Y = test$mpg01

k_s = c(1,3,5,7,10,15)

for(j in k_s){
 knn.pred = knn(train.X, test.X, train.Y, k=j)
 table(knn.pred, test.Y)
 print(1 - mean(knn.pred == test.Y) )
}
```

```
## [1] 0.1956522
## [1] 0.2391304
## [1] 0.2065217
## [1] 0.2065217
## [1] 0.1847826
## [1] 0.1847826
```

The test error appears to be lowest for `K = 15`, followed by `K = 1` out of `K = 1, 3, 5, 7, 10, 15`.

Exercise 15

(a)

```
Power = function(x){
  return(x **3)
  }

Power(2)
```

```
## [1] 8
```

(b)

```
Power2 = function(x, a){
  return(x ** a)
}

Power2(3,8)
```

```
## [1] 6561
```

(c)

```
Power2(10, 3)
```

```
## [1] 1000
```

```
Power2(8, 17)
```

```
## [1] 2.2518e+15
```

```
Power2(131, 3)
```

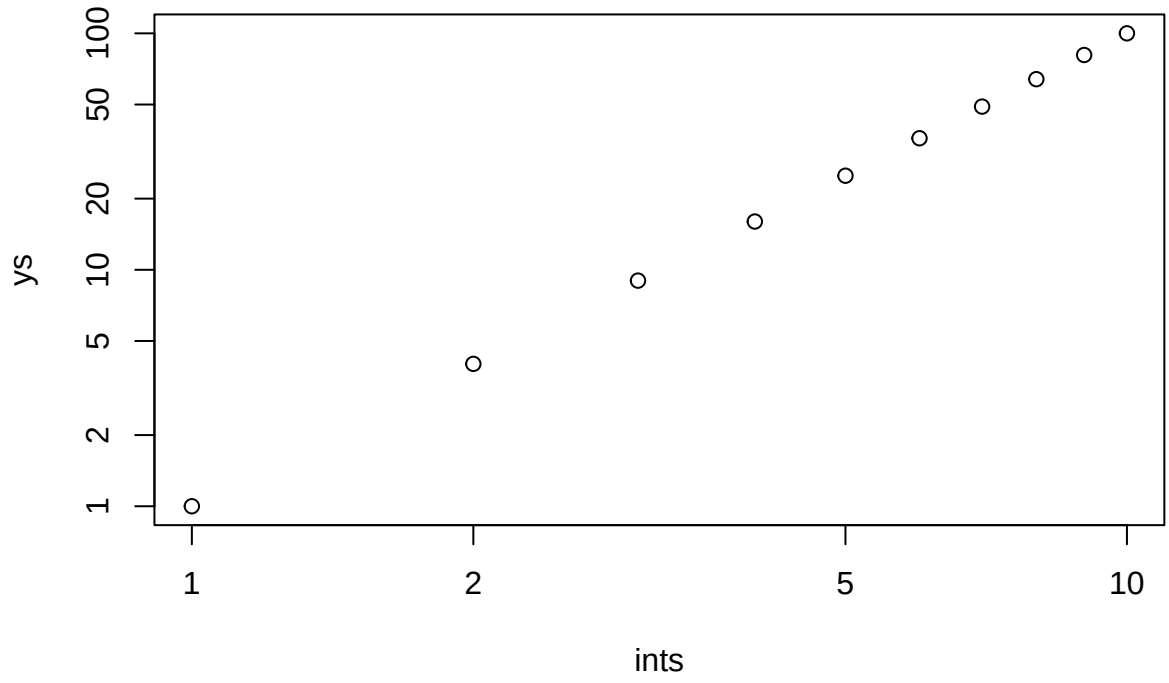
```
## [1] 2248091
```

(d)

```
Power3 = function(x, a){
  result = x ** a
  return(result)
}
```

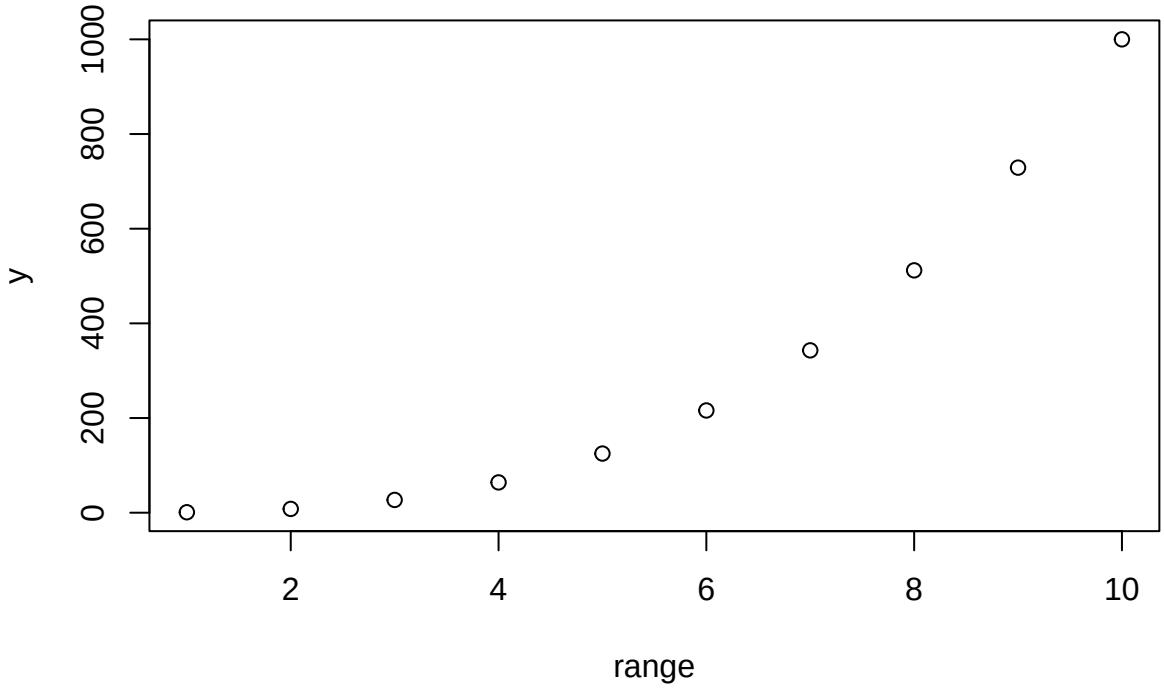
(e)

```
ints = 1:10
ys = c()

for (i in (1:10)){
  ys[[i]] = Power3(i, 2)
}

plot(ints, ys, log = "xy")
```



(f)

```
PlotPower = function(range, a){
  y = c()
  for (i in range){
  y[[i]] = Power3(i, a)
}
  plot(range, y)
  }

PlotPower(1:10, 3)
```



Exercise 16

```
Boston$above_median <- ifelse(Boston$crim > median(Boston$crim), 1, 0)
# pairs(Boston)

# split into training and test sets
train_num = 1:400
train = Boston[train_num, ]
test = Boston[-train_num, ]

# Logistic Regression
glm.fits = glm(above_median ~  indus + dis + ptratio,
               data = train, family = binomial)
glm.probs = predict(glm.fits, test, type = "response")
# dim(test)

glm.pred1 = rep(0, 106)
glm.pred1[glm.probs > 0.5] = 1
# table(glm.pred1, test$above_median)
paste("Logistic regression test error: ",
      1 - mean(glm.pred1 == test$above_median))
```

```
## [1] "Logistic regression test error:  0.141509433962264"
```

```r
# LDA
lda.fit = lda(above_median ~  indus + dis + ptratio,
              data = train)
lda.pred = predict(lda.fit, test)

lda.class = lda.pred$class
# table(lda.class, test$above_median)
paste("LDA test error: ",
      1 - mean(lda.class == test$above_median))
```

```
## [1] "LDA test error:  0.141509433962264"
```

```r
# Naive Bayes
nb.fit = naiveBayes(above_median ~  indus + dis + ptratio,
              data = train)
nb.class = predict(nb.fit, test)
# table(nb.class, test$above_median)
paste("Naive Bayes test error: ",
      1 - mean(nb.class == test$above_median))
```

```
## [1] "Naive Bayes test error:  0.141509433962264"
```

```r
# KNN models
train.X = data.frame(train$indus, train$dis, train$ptratio)
test.X = data.frame(test$indus, test$dis, test$ptratio)
train.Y = train$above_median
test.Y = test$above_median

k_s = c(1,3,5,7,10,15)

for(j in k_s){
 knn.pred = knn(train.X, test.X, train.Y, k=j)
 table(knn.pred, test.Y)
 print(paste("KNN test error for k = ", j, ": ",
      1 - mean(knn.pred == test.Y)))
}
```

```
## [1] "KNN test error for k =  1 :  0.0849056603773585"
## [1] "KNN test error for k =  3 :  0.0471698113207547"
## [1] "KNN test error for k =  5 :  0.0471698113207547"
## [1] "KNN test error for k =  7 :  0.0471698113207547"
## [1] "KNN test error for k =  10 :  0.0471698113207547"
## [1] "KNN test error for k =  15 :  0.0943396226415094"
```