MA679_HW3_Jianhao

Jianhaoyan
2/6/2019

```
## Loading required package: ParamHelpers
## ── Attaching packages ──────────────────────────────────────────────────
tidyverse 1.2.1 ──
## ✔ ggplot2 3.0.0     ✔ purrr   0.2.5
## ✔ tibble  1.4.2     ✔ dplyr   0.7.7
## ✔ tidyr   0.8.1     ✔ stringr 1.3.1
## ✔ readr   1.1.1     ✔ forcats 0.3.0
## ── Conflicts ───────────────────────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## Loading required package: MASS
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following object is masked from 'package:tidyr':
##
##     expand
## Loading required package: lme4
##
## arm (Version 1.10-1, built: 2018-4-12)
## Working directory is /Users/yanjianhao/Desktop/MA679HW/MA671HW
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
## The following object is masked from 'package:mlr':
##
##     train
```

6.Suppose we collect data for a group of students in a statistics class with variables X1 = hours studied, X2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\hat\beta_0 = -6, \hat\beta_1 = 0.05, \hat\beta_2 = 1$.

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

```
P = invlogit(-6+0.05*40+1*3.5)
print(paste("The probability of getting a A is", P))
## [1] "The probability of getting a A is 0.377540668798145"
```

(b) How many hours would the student in part (a) need to study to have a 50 % chance of getting an A in the class?

```
Hours= (logit(0.5)+6-3.5)/0.05
print(paste("The student needs to study", Hours, "everyday."))
## [1] "The student needs to study 50 everyday."
```

8.Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20 % on the training data and 30 % on the test data. Next we use 1-nearest neigh- bors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

#####We choose logistic regression. The testing error in KNN is 36% because the K equals 1. Comparing these two eror rates, the logistic regression model seems better than KNN.

9. This problem has to do with odds.

(a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

```r
p1=0.27
print(paste("we have on average a fraction of",p1,"of people defaulting on their credit card payment."))
## [1] "we have on average a fraction of 0.27 of people defaulting on their credit card payment."
```

(b) Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will de- fault?

```r
p2=0.16/(1-0.16)
print(paste("The odds that she will default is then",p2))
## [1] "The odds that she will default is then 0.19047619047619"
```

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 1089 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 ?

```r
library(ISLR)
kableExtra::kable(summary(Weekly))
```

```
 Year </th>
 Lag1 </th>
 Lag2 </th>
 Lag3 </th>
 Lag4 </th>
 Lag5 </th>
Volume </th>
Today </th>
Direction
Min. :1990
Min. :-18.1950
Min. :-18.1950
Min. :-18.1950
Min. :-18.1950
Min. :-18.1950
Min. :0.08747
Min. :-18.1950
Down:484
1st Qu.:1995
1st Qu.: -1.1540
1st Qu.: -1.1540
1st Qu.: -1.1580
1st Qu.: -1.1580
1st Qu.: -1.1660
1st Qu.:0.33202
1st Qu.: -1.1540
Up :605
Median :2000
Median : 0.2410
Median : 0.2410
Median : 0.2410
Median : 0.2380
Median : 0.2340
Median :1.00268
Median : 0.2410
NA
Mean :2000
Mean : 0.1506
Mean : 0.1511
Mean : 0.1472
Mean : 0.1458
Mean : 0.1399
```

Mean :1.57462
Mean : 0.1499
NA
3rd Qu.:2005
3rd Qu.: 1.4050
3rd Qu.: 1.4090
3rd Qu.: 1.4090
3rd Qu.: 1.4090
3rd Qu.: 1.4050
3rd Qu.:2.05373
3rd Qu.: 1.4050
NA
Max. :2010
Max. : 12.0260
Max. : 12.0260
Max. : 12.0260
Max. : 12.0260
Max. : 12.0260
Max. :9.32821
Max. : 12.0260
NA

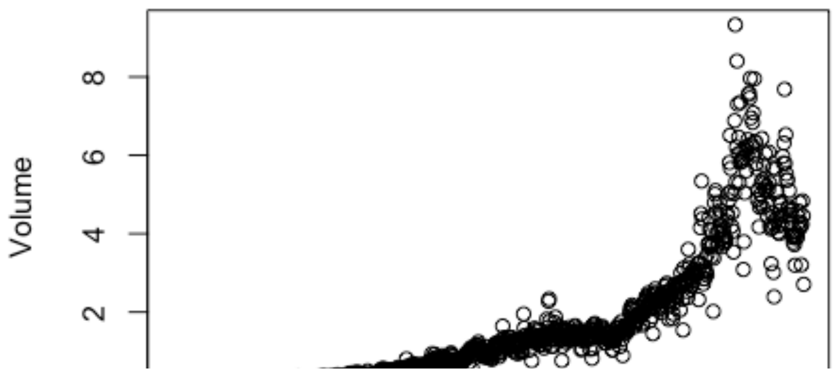
```
str(Weekly)
## 'data.frame':    1089 obs. of  9 variables:
##  $ Year     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
##  $ Lag1     : num  0.816 -0.27 -2.576 3.514 0.712 ...
##  $ Lag2     : num  1.572 0.816 -0.27 -2.576 3.514 ...
##  $ Lag3     : num  -3.936 1.572 0.816 -0.27 -2.576 ...
##  $ Lag4     : num  -0.229 -3.936 1.572 0.816 -0.27 ...
##  $ Lag5     : num  -3.484 -0.229 -3.936 1.572 0.816 ...
##  $ Volume   : num  0.155 0.149 0.16 0.162 0.154 ...
##  $ Today    : num  -0.27 -2.576 3.514 0.712 1.178 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
kableExtra::kable(cor(Weekly[,-9]))
```

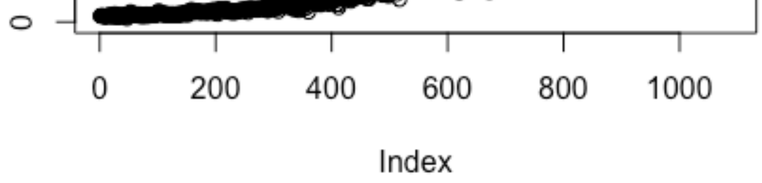
Year
Lag1
Lag2
Lag3
Lag4
Lag5
Volume
Today
Year
1.0000000
-0.0322893
-0.0333900
-0.0300065
-0.0311279
-0.0305191
0.8419416
-0.0324599
Lag1
-0.0322893
1.0000000
-0.0748531
0.0586357
-0.0712739
-0.0081831
-0.0649513
-0.0750318
Lag2
-0.0333900
-0.0748531
1.0000000
-0.0757209
0.0583815
-0.0724995
-0.0855131

0.0591667
Lag3
-0.0300065
0.0586357
-0.0757209
1.0000000
-0.0753959
0.0606572
-0.0692877
-0.0712436
Lag4
-0.0311279
-0.0712739
0.0583815
-0.0753959
1.0000000
-0.0756750
-0.0610746
-0.0078259
Lag5
-0.0305191
-0.0081831
-0.0724995
0.0606572
-0.0756750
1.0000000
-0.0585174
0.0110127
Volume
0.8419416
-0.0649513
-0.0855131
-0.0692877
-0.0610746
-0.0585174
1.0000000
-0.0330778
Today
-0.0324599
-0.0750318
0.0591667
-0.0712436
-0.0078259
0.0110127
-0.0330778
1.0000000

We can find the correlations of lag and today are really small.

```
attach(Weekly)
plot(Volume)
```





#####The most obvious relationship among all of the variables is between volume and Year. And from the graph, we can find volume increase over the time.

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 ?

```
model_1 = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family
= binomial)
summary(model_1)
```

```
##
## 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
```

Only intercept and lag2 are 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.

```
probs <- predict(model_1, type = "response")
pred.glm <- rep("Down", length(probs))
pred.glm[probs > 0.5] <- "Up"
table(pred.glm, Direction)
```

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

We may conclude that the percentage of correct predictions on the training data is (54+557)/1089wich is equal to 56.1065197%. In other words 43.8934803% is the training error rate, which is often overly optimistic. We could also say that for weeks when the market goes up, the model is right 92.0661157% of the time (557/(48+557)). For weeks when the market goes down, the model is right only 11.1570248% of the time (54/(54+430)).

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).

```
db_train =Weekly[,c(1,3,9)]%>%
 filter(Year>=1990&Year<=2008)
db_test= Weekly[,c(1,3,9)]%>%
 filter(Year>=2009&Year<=2010)
model_2<-glm(Direction~Lag2,data = db_train, family = binomial)
```

```
probs2 <- predict(model_2, db_test, type = "response")
pred.glm2 <- rep("Down", length(probs2))
pred.glm2[probs2 > 0.5] <- "Up"
table(pred.glm2, db_test$Direction)
##
## pred.glm2 Down Up
##      Down    9  5
##      Up     34 56
```

Repeat (d) using LDA.

```
library(MASS)
fit.lda <- lda(Direction ~ Lag2, data = db_train)
fit.lda
## Call:
## lda(Direction ~ Lag2, data = db_train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
pred.lda <- predict(fit.lda, db_test)
table(pred.lda$class, db_test$Direction)
##
##        Down Up
##   Down    9  5
##   Up     34 56
```

In this case, we may conclude that the percentage of correct predictions on the test data is 62.5%. In other words 37.5% is the test error rate. We could also say that for weeks when the market goes up, the model is right 91.8032787% of the time. For weeks when the market goes down, the model is right only 20.9302326% of the time. These results are very close to those obtained with the logistic regression model which is not surpising.

(f) Repeat (d) using QDA.

```
fit.qda <- qda(Direction~Lag2, data = db_train)
fit.qda
## Call:
## qda(Direction ~ Lag2, data = db_train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
pred.qda<-predict(fit.qda,db_test)
table(pred.qda$class,db_test$Direction)
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

n this case, we may conclude that the percentage of correct predictions on the test data is 58.6538462%. In other words 41.3461538% is the test error rate. We could also say that for weeks when the market goes up, the model is right 100% of the time. For weeks when the market goes down, the model is right only 0% of the time. We may note, that QDA achieves a correctness of 58.6538462% even though the model chooses "Up" the whole time !

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

```
library(class)
train.x = as.matrix(db_train$Lag2)
test.x = as.matrix(db_test$Lag2)
train.direction<-factor(db_train$Direction)
fit.knn = knn(train.x,test.x,train.direction,k=1)
table(fit.knn, db_test$Direction)
```

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

In this case, we may conclude that the percentage of correct predictions on the test data is 50%. In other words 50% is the test error rate. We could also say that for weeks when the market goes up, the model is right 50.8196721% of the time. For weeks when the market goes down, the model is right only 48.8372093% of the time.

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

If we compare the test error rates, we see that logistic regression and LDA have the minimum error rates, followed by QDA and KNN.

(i) Experiment with different combinations of predictors, includ- ing possible transformations and interactions, for each of the methods. Report the variables, method, and associated confu- sion 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.

```
train <- (Year < 2009)
Weekly.20092010 <- Weekly[!train, ]
Direction.20092010 <- Direction[!train]
train.X <- as.matrix(Lag2[train])
test.X <- as.matrix(Lag2[!train])
train.Direction <- Direction[train]
fit.glm3 <- glm(Direction ~ Lag2:Lag1, data = Weekly, family = binomial, subset = train)
probs3 <- predict(fit.glm3, Weekly.20092010, type = "response")
pred.glm3 <- rep("Down", length(probs3))
pred.glm3[probs3 > 0.5] = "Up"
table(pred.glm3, Direction.20092010)
##          Direction.20092010
## pred.glm3 Down Up
##      Down    1  1
##      Up     42 60
mean(pred.glm3 == Direction.20092010)
## [1] 0.5865385
# LDA with Lag2 interaction with Lag1
fit.lda2 <- lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train)
pred.lda2 <- predict(fit.lda2, Weekly.20092010)
mean(pred.lda2$class == Direction.20092010)
## [1] 0.5769231
# QDA with sqrt(abs(Lag2))
fit.qda2 <- qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = train)
pred.qda2 <- predict(fit.qda2, Weekly.20092010)
table(pred.qda2$class, Direction.20092010)
##       Direction.20092010
##        Down Up
##   Down   12 13
##   Up     31 48
mean(pred.qda2$class == Direction.20092010)
## [1] 0.5769231
# KNN k =10
pred.knn2 <- knn(train.X, test.X, train.Direction, k = 10)
table(pred.knn2, Direction.20092010)
##          Direction.20092010
## pred.knn2 Down Up
##      Down   18 21
##      Up     25 40
mean(pred.knn2 == Direction.20092010)
## [1] 0.5576923
# KNN k = 100
pred.knn3 <- knn(train.X, test.X, train.Direction, k = 100)
table(pred.knn3, Direction.20092010)
##          Direction.20092010
## pred.knn3 Down Up
##      Down    9 12
##      Up     34 49
mean(pred.knn3 == Direction.20092010)
## [1] 0.5576923
```

4.11

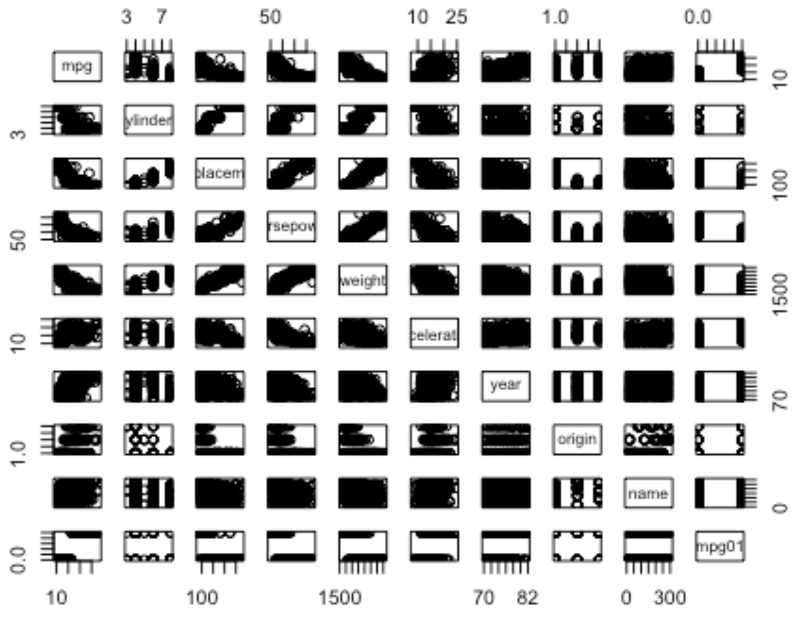
In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the "Auto" data set.

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.

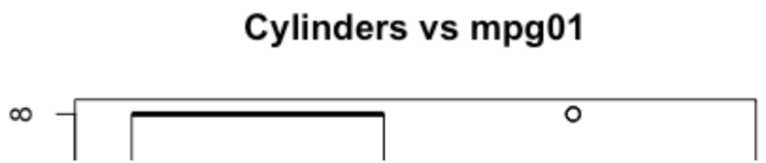
```
library(ggplot2)
attach(Auto)
## The following object is masked from package:ggplot2:
##
##     mpg
mpg01 <- rep(0, length(mpg))
mpg01[mpg > median(mpg)] <- 1
Auto <- data.frame(Auto, mpg01)
```

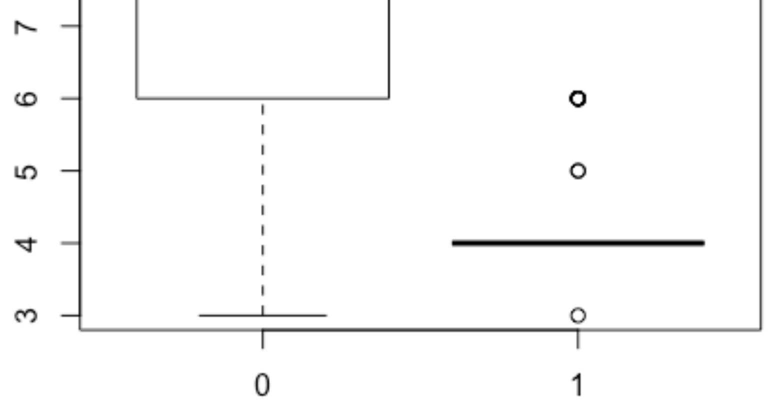
Explore the data graphically in order to investigate the association between "mpg01" and the other features. Which of the other features seem most likely to be useful in predicting "mpg01" ? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
cor(Auto[, -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
pairs(Auto)
```

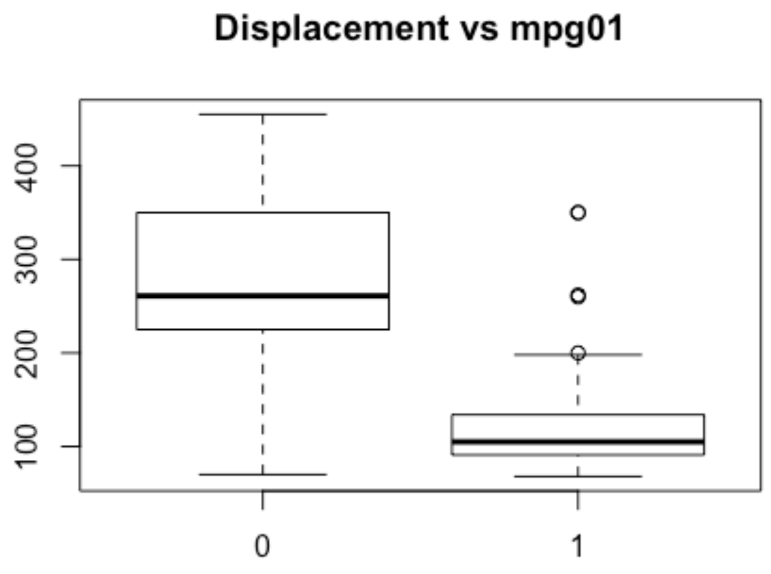


```
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")
```

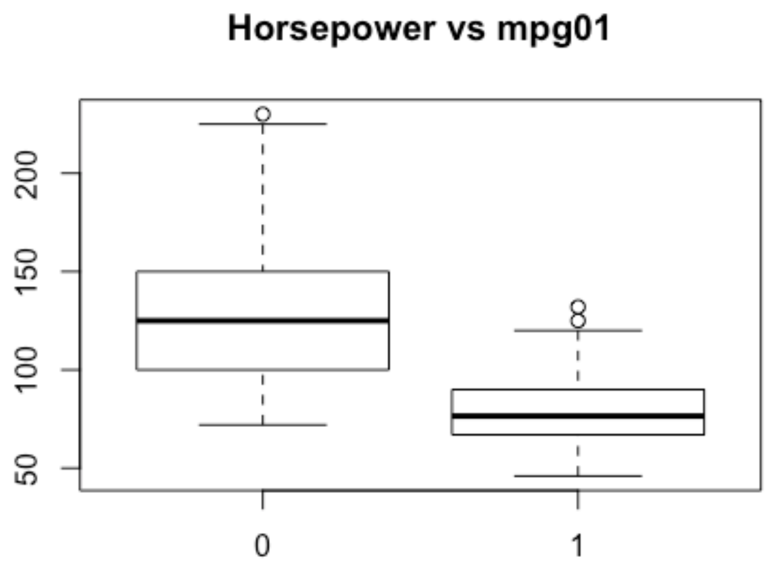




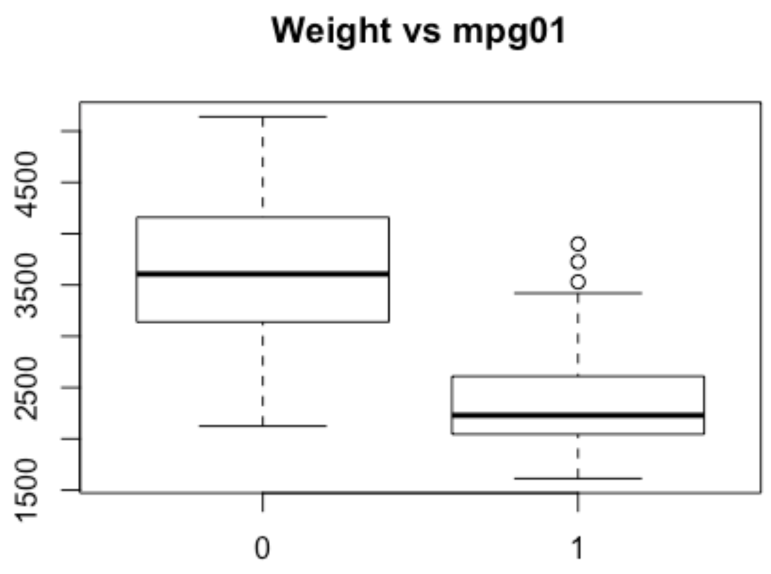
```
boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")
```



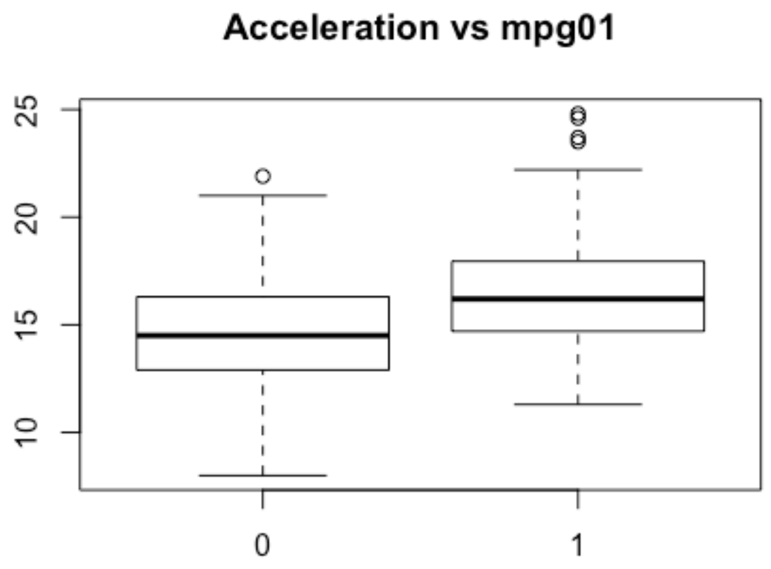
```
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")
```



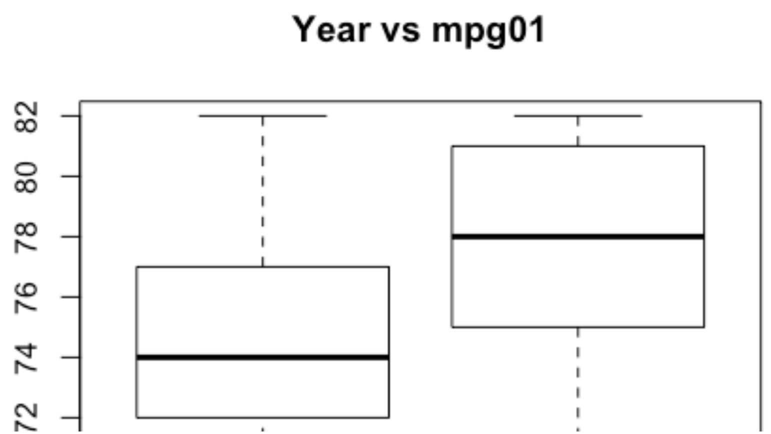
```r
boxplot(weight ~ mpg01, data = Auto, main = "Weight vs mpg01")
```



```r
boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs mpg01")
```



```r
boxplot(year ~ mpg01, data = Auto, main = "Year vs mpg01")
```



#####We may conclude that there exists some association between "mpg01" and "cylinders", "weight", "displacement" and "horsepower".

Split the data into a training set and a test set.

```
train <- (year %% 2 == 0)
Auto.train <- Auto[train, ]
Auto.test <- Auto[!train, ]
mpg01.test <- mpg01[!train]
```

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 ?

```
fit.lda <- lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = 
train)
fit.lda
## Call:
## lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
##     subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4571429 0.5428571
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.812500 3604.823     271.7396  133.14583
## 1  4.070175 2314.763     111.6623   77.92105
##
## Coefficients of linear discriminants:
##                        LD1
## cylinders    -0.6741402638
## weight       -0.0011465750
## displacement  0.0004481325
## horsepower    0.0059035377
pred.lda <- predict(fit.lda, Auto.test)
table(pred.lda$class, mpg01.test)
##    mpg01.test
##      0  1
##   0 86  9
##   1 14 73
mean(pred.lda$class != mpg01.test)
## [1] 0.1263736
```

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 ?

```
fit.qda <- qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = 
train)
fit.qda
## Call:
## qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
##     subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4571429 0.5428571
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.812500 3604.823     271.7396  133.14583
## 1  4.070175 2314.763     111.6623   77.92105
mean(pred.qda$class != mpg01.test)
## Warning in `!=.default`(pred.qda$class, mpg01.test): longer object length
## is not a multiple of shorter object length
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
## [1] 1
## [1] 1
```

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 ?

```
fit.glm <- glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, family = 
binomial, subset = train)
summary(fit.glm)
## 
## Call:
## glm(formula = mpg01 ~ cylinders + weight + displacement + horsepower, 
##     family = binomial, data = Auto, subset = train)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -2.48027  -0.03413   0.10583   0.29634   2.57584  
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  17.658730   3.409012   5.180 2.22e-07 ***
## cylinders    -1.028032   0.653607  -1.573   0.1158    
## weight       -0.002922   0.001137  -2.569   0.0102 *  
## displacement  0.002462   0.015030   0.164   0.8699    
## horsepower   -0.050611   0.025209  -2.008   0.0447 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 289.58  on 209  degrees of freedom
## Residual deviance:  83.24  on 205  degrees of freedom
## AIC: 93.24
## 
## Number of Fisher Scoring iterations: 7
probs <- predict(fit.glm, Auto.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, mpg01.test)
##         mpg01.test
## pred.glm  0  1
##        0 89 11
##        1 11 71
mean(pred.glm != mpg01.test)
## [1] 0.1208791
```

Perform KNN on the training data, with several values of K

, in order to predict “mpg01” using 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 ?

```
train.X <- cbind(cylinders, weight, displacement, horsepower)[train, ]
test.X <- cbind(cylinders, weight, displacement, horsepower)[!train, ]
train.mpg01 <- mpg01[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.mpg01, k = 1)
table(pred.knn, mpg01.test)
##         mpg01.test
## pred.knn  0  1
##        0 83 11
##        1 17 71
mean(pred.knn != mpg01.test)
## [1] 0.1538462
pred.knn <- knn(train.X, test.X, train.mpg01, k = 10)
table(pred.knn, mpg01.test)
##         mpg01.test
## pred.knn  0  1
##        0 77  7
##        1 23 75
mean(pred.knn != mpg01.test)
## [1] 0.1648352
```

4.12

Write a function, Power(), that prints out the result of raising 2 to the 3rd power. In other words, your function should compute 23and print out the results.

```
Power <- function() {
```

```
Power <- function() {
  2^3
}

Power()
## [1] 8
```

Create a new function, Power2(), that allows you to pass any two numbers, "x" and "a", and prints out the value of "x^a".

```
Power2 <- function(x,a){
 x^a
}
Power2(2,3)
## [1] 8
```

Using the Power2() function that you just wrote, compute 103, 817, and 1313.

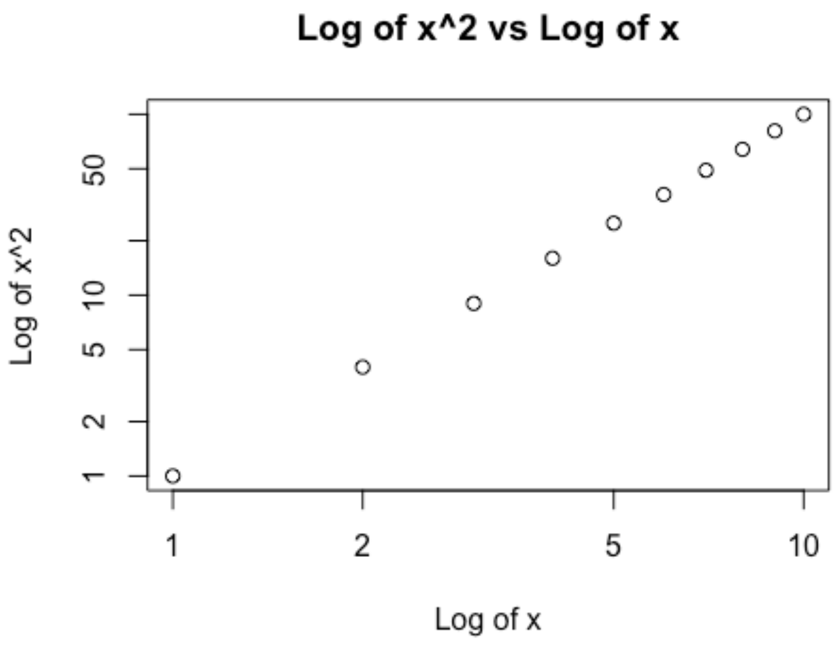
```
Power2(10, 3)
## [1] 1000
Power2(8, 17)
## [1] 2.2518e+15
Power2(131, 3)
## [1] 2248091
```

Now create a new function, Power3(), that actually returns the result "x^a" as an R object, rather than simply printing it to the screen. That is, if you store the value "x^a" in an object called "result" within your function, then you can simply return() this result.

```
Power3 <- function(x , a) {
  result <- x^a
  return(result)
}
```

Now using the Power3() function, create a plot of f(x)=x3. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x2. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either teh x-axis, the y-axis, or both on the log-scale.

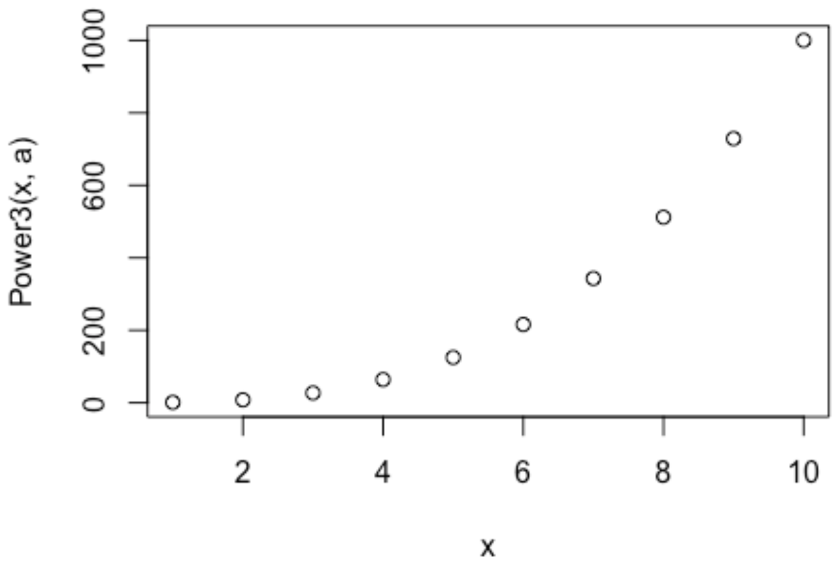
```
x <- 1:10
plot(x, Power3(x, 2), log = "xy", xlab = "Log of x", ylab = "Log of x^2", main = "Log of x^2 vs
Log of x")
```



Create a function, PlotPower(), that allows you to create a plot of "x" against "x^a" for a fixed "a" and for a range of values of "x".

```
PlotPower <- function(x, a) {
  plot(x, Power3(x, a))
}

PlotPower(1:10, 3)
```



13

Using the "Boston" data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore the logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

```
library(MASS)
attach(Boston)
crim01 <- rep(0, length(crim))
crim01[crim > median(crim)] <- 1
Boston <- data.frame(Boston, crim01)

train <- 1:(length(crim) / 2)
test <- (length(crim) / 2 + 1):length(crim)
Boston.train <- Boston[train, ]
Boston.test <- Boston[test, ]
crim01.test <- crim01[test]
fit.glm <- glm(crim01 ~ . - crim01 - crim, data = Boston, family = binomial, subset = train)
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim01.test)
##         crim01.test
## pred.glm   0   1
##        0  68  24
##        1  22 139
mean(pred.glm != crim01.test)
## [1] 0.1818182
```

We may conclude that, for this logistic regression, we have a test error rate of 18.1818182%.

```
fit.glm <- glm(crim01 ~ . - crim01 - crim - chas - nox, data = Boston,family = binomial, subset =
train)
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim01.test)
##         crim01.test
## pred.glm   0   1
##        0  78  28
##        1  12 135
mean(pred.glm != crim01.test)
## [1] 0.1581028
```

We may conclude that, for this logistic regression, we have a test error rate of 15.8102767%.

```
fit.lda <- lda(crim01 ~ . - crim01 - crim, data = Boston, subset = train)
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim01.test)
##    crim01.test
##       0   1
##   0  80  24
##   1  10 139
mean(pred.lda$class != crim01.test)
## [1] 0.1343874
```

We may conclude that, for this LDA, we have a test error rate of 13.4387352%.

```
fit.lda <- lda(crim01 ~ . - crim01 - crim - chas - nox, data = Boston, subset = train)
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim01.test)
##    crim01.test
##       0   1
##   0  82  30
##   1   8 133
mean(pred.lda$class != crim01.test)
## [1] 0.1501976
```

We may conclude that, for this LDA, we have a test error rate of 15.0197628%.

```
train.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[train, ]
test.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[test, ]
train.crim01 <- crim01[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim01, k = 1)
table(pred.knn, crim01.test)
##         crim01.test
## pred.knn   0   1
##        0  85 111
##        1   5  52
mean(pred.knn != crim01.test)
## [1] 0.458498
```

We may conclude that, for this KNN (k=1), we have a test error rate of 45.8498024%.

```
pred.knn <- knn(train.X, test.X, train.crim01, k = 10)
table(pred.knn, crim01.test)
##         crim01.test
## pred.knn   0   1
##        0  83  23
##        1   7 140
mean(pred.knn != crim01.test)
## [1] 0.1185771
```