Homework 2

Zili Ma, Xiao-Qun Wang

Q1. Using a little bit of algebra, prove that (4.2) is equivalent to (4.3). In other words, the logistic function representation and logit representation for the logistic regression model are equivalent.

A1.

Given $$P(x) = \frac{\exp(\beta_0+\beta_1 x)}{1+\exp(\beta_0+\beta_1 x)} \quad (4.2).$$

$$\frac{P(x)}{1-P(x)} = \exp(\beta_0+\beta_1 x) \quad (4.3)$$, substitude (4.3) into RHS of (4.2).

then. RHS $$\frac{\frac{P(x)}{1-P(x)}}{1+\frac{P(x)}{1-P(x)}} = \frac{P(x)}{1-P(x)+P(x)} = P(x) = LHS \quad //$$

Q2. We now examine the differences between LDA and QDA

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

A2a: LDA is better in the test set, QDA is better in the training set but less good in the test data. For the linear boundary, LDA can provide an effective classification with both low variance and bias of model, while QDA may introduce more variance and overfit the training set.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

A2b: QDA is better on both training and test sets than LDA.

(c) In general, as the sample size *n* increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

A2c: Generally, QDA will be better when the training set is very large, then the variance of the classifier will not be a major concern.

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

A2d: False. QDA will introduce a larger variance of the model for the test data and increase the test error, if overfitting occurs.

Q3. 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 neighbors (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?

A3. In case of 1-nearest neighbor classifier, the train error should equal to 0, because of the training observation has been classified already and posteriorly.Therefore, the test error rate generated by the 1NN method is actually equal to 36%, which is higher than the former method. So, we should choose logistic regression instead of 1NN.

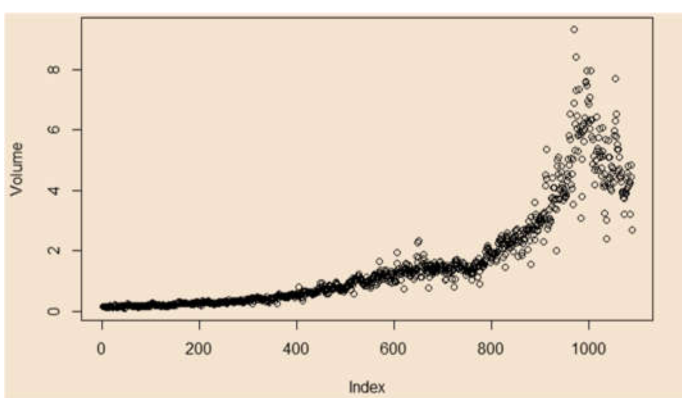
Q4. This question should be answered using the *Weekly* data set, which is part of the *ISLR* package. This data is similar in nature to the *Smarket* data from this chapter's lab, except that it contains 1, 089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

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

```
> library(ISLR)
> 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
```

```
> 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
```

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



(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?

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

A4b: As shown in above result, only ***Lag2*** is shown as a significant predictor with p-value smaller than 0.05.

(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")
> dim(Weekly)
[1] 1089    9
> contrasts(Direction)
     Up
Down  0
Up    1
> glm.pred=rep("Down",1089)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred,Direction)
        Direction
glm.pred Down  Up
    Down   54  48
    Up    430 557
```

A4c: As shown in above result, the error rate of the training data is (430+48)/1089*100% which equals to 43.9%, The percentage of correct prediction is 1-43.9% equaling to 56.1%. For weeks when the market goes up, the probability for the model predicting the right direction is 557/(557+48) equaling to 92%. For weeks when the market goes down, the probability for the model predicting the right direction is 55/(54+430) equaling to 11.6%.

(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)
Weekly.2009=Weekly[!train,]
Direction.2009=Direction[!train]
glm.fit1=glm(Direction~Lag2,family=binomial,data=Weekly,subset=train)
glm.probs1=predict(glm.fit1,Weekly.2009,type="response")
glm.pred1=rep("Down",length(glm.probs1))
glm.pred1[glm.probs1>.5]="Up"
summary(glm.fit1)
table(glm.pred1,Direction.2009)
```

```
Call:
glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
    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
```

```
> table(glm.pred1,Direction.2009)
         Direction.2009
glm.pred1 Down Up
     Down    9  5
     Up     34 56
> mean(glm.pred1==Direction.2009)
[1] 0.625
```

A4d: The overall fraction of correct predictions for the held out data is (9+56)/(9+5+34+56) equaling to 62.5%.

(e) Repeat (d) using LDA.

```
library(MASS)
lda.fit=lda(Direction~Lag2,data=Weekly,subset=train)
lda.fit
```

```
Call:
lda(Direction ~ Lag2, data = Weekly, subset = 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
```

```
lda.pred=predict(lda.fit,Weekly.2009)
table(lda.pred$class,Direction.2009)
mean(lda.pred$class==Direction.2009)
```

```
> table(lda.pred$class,Direction.2009)
      Direction.2009
       Down Up
  Down    9  5
  Up     34 56
> mean(lda.pred$class==Direction.2009)
[1] 0.625
```

A4e: The overall fraction of correct predictions for the held out data is (9+56)/(9+5+34+56) equaling to 62.5%.

(f) Repeat (d) using QDA.

```
qda.fit=qda(Direction~Lag2,data=Weekly,subset=train)
qda.fit
```

```
Call:
qda(Direction ~ Lag2, data = Weekly, subset = train)

Prior probabilities of groups:
     Down        Up
0.4477157 0.5522843

Group means:
            Lag2
Down -0.03568254
Up    0.26036581
```

```
qda.pred=predict(qda.fit,Weekly.2009)
table(qda.pred$class,Direction.2009)
mean(qda.pred$class==Direction.2009)
```

```
> table(qda.pred$class,Direction.2009)
      Direction.2009
       Down Up
  Down    0  0
  Up     43 61
> mean(qda.pred$class==Direction.2009)
[1] 0.5865385
```

A4f: The overall fraction of correct predictions for the held out data is equaling to 58.6%. We may note, that QDA achieves a correctness of 58.7% even though the model chooses "Up" the whole time.

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

```
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.2009)
```

```
> table(knn.pred,Direction.2009)
        Direction.2009
knn.pred Down Up
    Down   21 30
    Up     22 31
> mean(knn.pred==Direction.2009)
[1] 0.5
```

A4g: 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 on this data?

A4h: 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, 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.

A4i:

(1) Logistic regression with Lag2 interaction with Lag1.

```
#Logistic regression with Lag2:Lag1
glm.fit2=glm(Direction~Lag2:Lag1,data=Weekly,family=binomial,subset=train)
glm.probs2=predict(glm.fit2,Weekly.2009,type="response")
glm.pred2=rep("Down",length(glm.probs2))
glm.pred2[glm.probs2>.5]="Up"
table(glm.pred2,Direction.2009)
mean(glm.pred2==Direction.2009)
```

```
> table(glm.pred2,Direction.2009)
         Direction.2009
glm.pred2 Down Up
     Down    1  1
     Up     42 60
> mean(glm.pred2==Direction.2009)
[1] 0.5865385
```

(2) LDA with Lag2 interaction with Lag1

```
#LDA with Lag2:Lag1
lda.fit2=lda(Direction~Lag2:Lag1,data=Weekly,subset=train)
lda.pred2=predict(lda.fit2,Weekly.2009)
table(lda.pred2$class,Direction.2009)
mean(lda.pred2$class==Direction.2009)
```

```
> table(lda.pred2$class,Direction.2009)
      Direction.2009
       Down Up
  Down    0  1
  Up     43 60
> mean(lda.pred2$class==Direction.2009)
[1] 0.5769231
```

(3) QDA with square root of the absolute value of Lag2

```
# QDA with sqrt(abs(Lag2))
qda.fit2 <- qda(Direction ~ Lag2 + sqrt(abs(Lag2)),
                data = Weekly, subset = train)
qda.pred2 <- predict(qda.fit2, Weekly.2009)
table(qda.pred2$class, Direction.2009)
mean(qda.pred2$class==Direction.2009)
```

```
> table(qda.pred2$class, Direction.2009)
      Direction.2009
       Down Up
  Down   12 13
  Up     31 48
> mean(qda.pred2$class==Direction.2009)
[1] 0.5769231
```

(4) KNN with K=10

```
#KNN K=10
knn.pred2=knn(train.X,test.X,train.Direction,k=10)
table(knn.pred2,Direction.2009)
mean(knn.pred2==Direction.2009)
```

```
> table(knn.pred2,Direction.2009)
         Direction.2009
knn.pred2 Down Up
     Down   17 18
     Up     26 43
> mean(knn.pred2==Direction.2009)
[1] 0.5769231
```

(5) KNN with K=50

```
#KNN K=50
knn.pred3=knn(train.X,test.X,train.Direction,k=50)
table(knn.pred3,Direction.2009)
mean(knn.pred3==Direction.2009)
```

```
> table(knn.pred3,Direction.2009)
         Direction.2009
knn.pred3 Down Up
     Down   20 22
     Up     23 39
> mean(knn.pred3==Direction.2009)
[1] 0.5673077
```

From above results of combinations, the original logistic regression and LDA have the best performance in terms of test error rates.

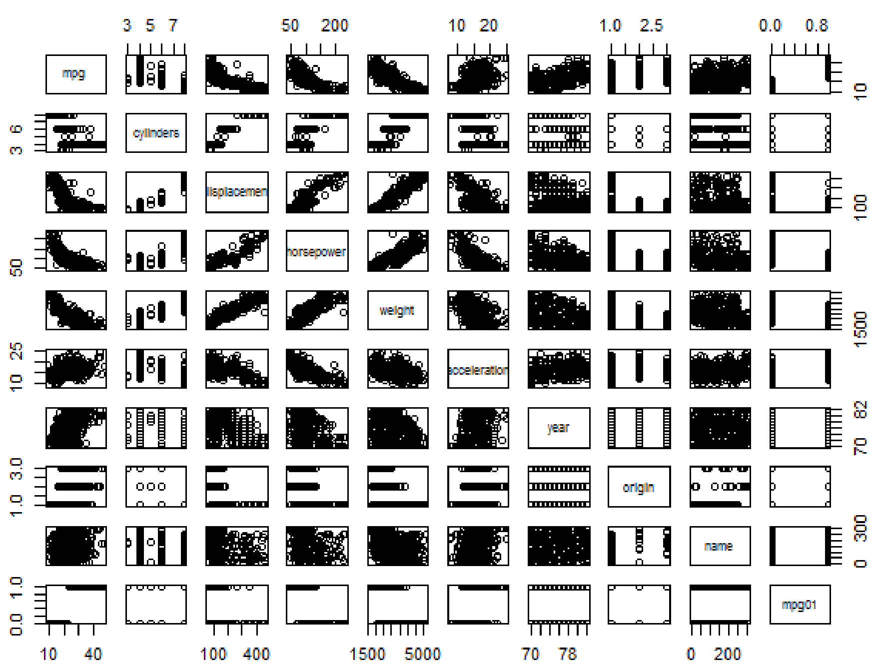
Q5. In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the *Auto* data set.

(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 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.

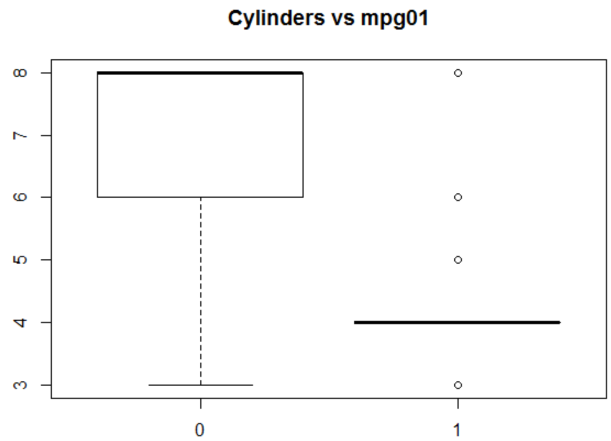
```
data(Auto)
mpg01=rep(0,length(mpg))
mpg01[mpg>median(mpg)]=1
Auto<-data.frame(Auto,mpg01)
```

(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.

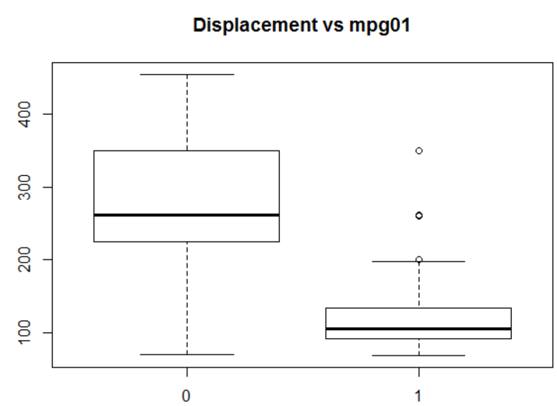
```
plot(Auto)
```



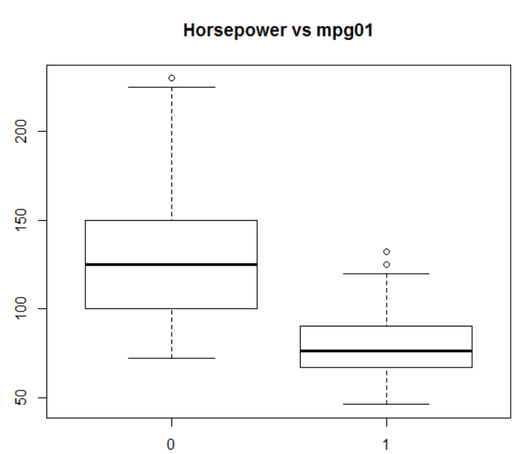
```
boxplot(cylinders~mpg01,data=Auto,main="Cylinders vs mpg01")
```



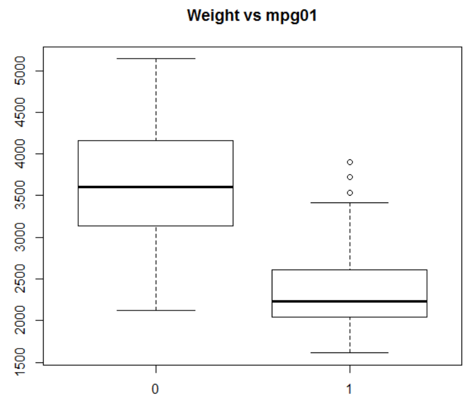

```
boxplot(displacement~mpg01,data=Auto,main="Displacement vs mpg01")
```



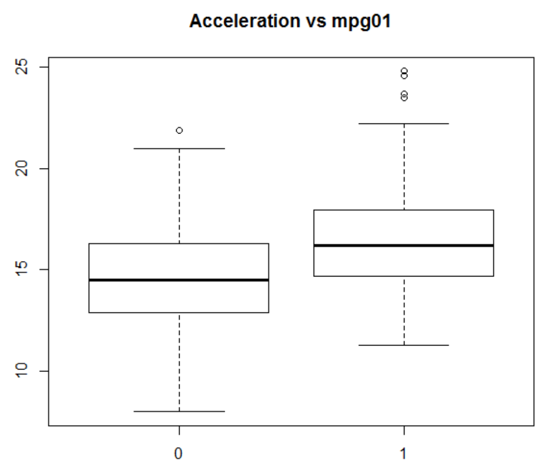

```
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")
```



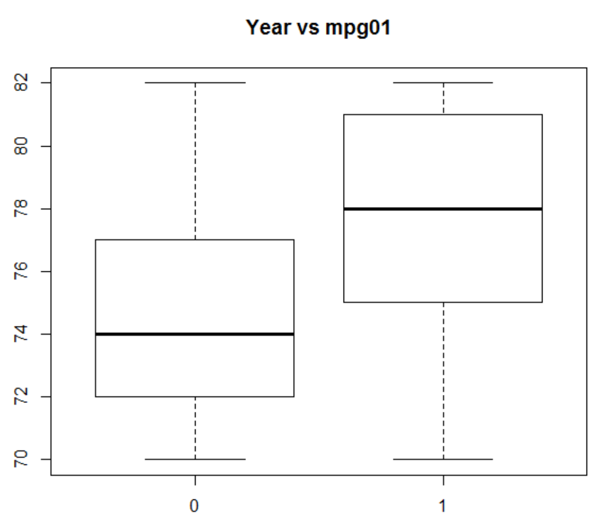
```
boxplot(weight ~ mpg01, data = Auto, main = "Weight vs mpg01")
```




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




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



```
> cov(Auto[,-9])
                     mpg    cylinders displacement  horsepower      weight
mpg            60.918142  -10.3529281   -657.58521  -233.85793  -5517.4407
cylinders     -10.352928    2.9096965    169.72195    55.34824   1300.4244
displacement -657.585207  169.7219486  10950.36755  3614.03374  82929.1001
horsepower   -233.857926   55.3482436   3614.03374  1481.56939  28265.6202
weight      -5517.440704 1300.4243632  82929.10014 28265.62023 721484.7090
acceleration    9.115514   -2.3750522   -156.99444   -73.18697   -976.8153
year           16.691477   -2.1719296   -142.57213   -59.03643   -967.2285
origin          3.553510   -0.7817344    -51.80079   -14.11274   -400.2660
mpg01           3.270332   -0.6483376    -39.47379   -12.85422   -322.2315
             acceleration         year       origin       mpg01
mpg             9.1155144   16.6914766    3.5535101   3.2703325
cylinders      -2.3750522   -2.1719296   -0.7817344  -0.6483376
displacement -156.9944354 -142.5721332  -51.8007921 -39.4737852
horsepower    -73.1869670  -59.0364320  -14.1127407 -12.8542199
weight       -976.8152526 -967.2284566 -400.2660499 -322.2314578
acceleration    7.6113312    2.9504619    0.4727882   0.4790281
year            2.9504619   13.5699149    0.5386502   0.7928389
origin          0.4727882    0.5386502    0.6488595   0.2071611
mpg01           0.4790281    0.7928389    0.2071611   0.2506394
```

A5b: There exists some association between “mpg01” and “cylinders”, “weight”, “displacement” and “horsepower”.

(c) 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]
```

(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 + weight + displacement + horsepower,
              data = Auto, subset = train)
lda.fit
```

```
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
```

```
lda.pred=predict(lda.fit, Auto.test)
table(lda.pred$class, mpg01.test)
mean(lda.pred$class!=mpg01.test)
```

```
> table(lda.pred$class, mpg01.test)
   mpg01.test
     0  1
  0 86  9
  1 14 73
> mean(lda.pred$class!=mpg01.test)
[1] 0.1263736
```

A5d: The test error of the model obtained is 12.6%.

(e) 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?

```
qda.fit=qda(mpg01 ~ cylinders + weight + displacement + horsepower,
            data = Auto, subset = train)
qda.fit
```

```
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
```

```
qda.pred=predict(qda.fit, Auto.test)
table(qda.pred$class, mpg01.test)
mean(qda.pred$class!=mpg01.test)
```

```
> table(qda.pred$class, mpg01.test)
   mpg01.test
     0  1
  0 89 13
  1 11 69
> mean(qda.pred$class!=mpg01.test)
[1] 0.1318681
```

A5e: The test error of the model obtained is 13.2%.

(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 + weight + displacement + horsepower,
            data = Auto, subset = train)
glm.fit
```

```
Call:  glm(formula = mpg01 ~ cylinders + weight + displacement + horsepower,
    data = Auto, subset = train)

Coefficients:
 (Intercept)     cylinders        weight  displacement    horsepower
   1.9936920    -0.1570599    -0.0002671     0.0001044     0.0013754

Degrees of Freedom: 209 Total (i.e. Null);  205 Residual
Null Deviance:      52.11
Residual Deviance: 16.54        AIC: 74.26
```

```
glm.probs=predict(glm.fit, Auto.test)
glm.pred=rep(0,length(glm.probs))
glm.pred[glm.probs>.5]=1
table(glm.pred,mpg01.test)
mean(glm.pred!=mpg01.test)
```

```
> table(glm.pred,mpg01.test)
        mpg01.test
glm.pred  0  1
       0 86  9
       1 14 73
> mean(glm.pred!=mpg01.test)
[1] 0.1263736
```

A5f: The test error of the model obtained is 12.6%.

(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?

K=1

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

```
> 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
```

K=10

```
pred.knn <- knn(train.X, test.X, train.mpg01, k = 10)
table(pred.knn, mpg01.test)
mean(pred.knn != mpg01.test)
```

```
> 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
```

K=100

```
pred.knn <- knn(train.X, test.X, train.mpg01, k = 100)
table(pred.knn, mpg01.test)
mean(pred.knn != mpg01.test)
```

```
> table(pred.knn, mpg01.test)
        mpg01.test
pred.knn  0  1
       0 81  7
       1 19 75
> mean(pred.knn != mpg01.test)
[1] 0.1428571
```

A5g: We may conclude that we have a test error rate of 14.3% for K=100. So, a K value of 100 seems to perform the best.

Q6. Using basic statistical properties of the variance, as well as single variable calculus, derive (5.6). In other words, prove that *α* given by (5.6) does indeed minimize Var(*αX*+(1−*α*)*Y*).

Ans:

First,

$\mathrm{Var}(\alpha X+(1-\alpha)Y)=\alpha^2\sigma_X^2+(1-\alpha)^2\sigma_Y^2+2\alpha(1-\alpha)\sigma_{XY}$.

We now take the fist derivative of $\mathrm{Var}(\alpha X+(1-\alpha)Y)$ *relative to* α*:*

$\frac{\partial \mathrm{Var}(\alpha X+(1-\alpha)Y)}{\partial\alpha}=2\alpha\sigma_X^2-2\sigma_Y^2+2\alpha\sigma_Y^2+2\sigma_{XY}-4\alpha\sigma_{XY}$.

We then set this equation to zero in order to pursue a minimized $\mathrm{Var}(\alpha X+(1-\alpha)Y)$. So we can get:

$2\alpha\sigma_X^2-2\sigma_Y^2+2\alpha\sigma_Y^2+2\sigma_{XY}-4\alpha\sigma_{XY}=0$

Then we get:

$\alpha=\frac{\sigma_Y^2-\sigma_{XY}}{\sigma_X^2+\sigma_Y^2-2\sigma_{XY}}$

Also we can check whether *the second derivative is positive to assure that it is indeed minimized value.*

$\frac{\partial^2 \mathrm{Var}(\alpha X+(1-\alpha)Y)}{\partial\alpha^2}=2\sigma_X^2+2\sigma_Y^2-4\sigma_{XY}=2\ \mathrm{Var}(X\text{-}Y)\geq 0$

Q7. We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of *n* observations.

a. **What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.**

Ans: *1−1/n*

b. What is the probability that the second bootstrap observation is not the jth observation from the original sample ?

Ans: 1−1/n

c. Argue that the probability that the jth observation is not in the bootstrap sample is $\mathbf{(1-1/n)}^{n}$

Ans:

The probability that the jth observation is not in the bootstrap sample is the product of the probabilities that each bootstrap observation is not the jth observation from the original sample :

$$(1-1/n)\cdots(1-1/n)=(1-1/n)^{n}$$

Note that *these probabilities are independant.*

d. When *n*=5, what is the probability that the jth observation is in the bootstrap sample?

Ans : The probability that the jth observation is in the bootstrap sample *is* $1-(1-1/5)^{5}=0.672$

e. When *n*=100, what is the probability that the jth observation is in the bootstrap sample ?

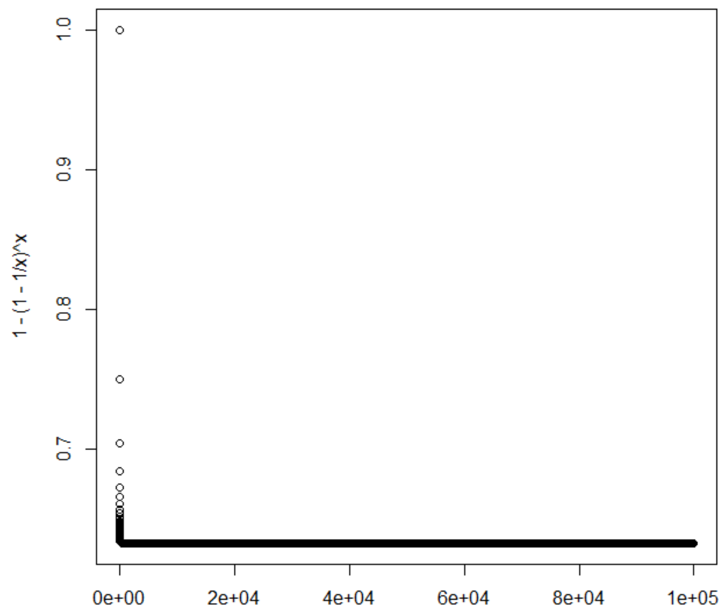
Ans : The probability that the jth observation is in the bootstrap sample *is* $1-(1-1/5)^{100}=0.634$

f. When *n*=10000, what is the probability that the jth observation is in the bootstrap sample ?

Ans : The probability that the jth observation is in the bootstrap sample *is* $1-(1-1/5)^{1000}=0.632$

g. Create a plot that displays, for each integer value of *n* from 1 to 100000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

- x <- 1:100000
- plot(x, 1 - (1 - 1/x)^x)



As shown in above, this plot quickly reaches an asymptote at about y = 0.63.

h. We will now investigate numerically the probability that a bootstrap sample of size *n*=100 contains the jth observation. Here *j*=4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

Ans:

```
> res = 0
> for (i in 1:10000) {
+   if(sum(sample(1:100,rep = TRUE) == 4) > 0){
+       res = res + 1
+       }
+ }
> pro = res/10000
> pro
[1] 0.6315
```

As shown in above, if the times we repeat is large enough, then the probability of a bootstrap sample that contains the jth observation will approach to 0.632. As we known, $\lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n = e^x$. *So the probability will converge to 1−1/e = 0.632 when* $n \to \infty$

Q8. In Chapter 4, we used logisitc regression to predict the probability of "default" using "income" and "balance" on the "Default" data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

a. Fit a logistic regression model that uses "income" and "balance" to predict "default".

Ans:

```
> library(ISLR)
> attach(Default)
> set.seed(1)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
> summary(fit.glm)

Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4725  -0.1444  -0.0574  -0.0211   3.7245

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8
```

b. Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

1. Split the sample set into a training set and a validation set.

Ans: `train <- sample(dim(Default)[1], dim(Default)[1] / 2)`

2. Fit a multiple logistic regression model using only the training observations.

Ans:

```
fit.glm <- glm(default ~ income + balance, data = Default, family
= "binomial", subset = train)
summary(fit.glm)
```

```
Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3583  -0.1268  -0.0475  -0.0165   3.8116

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.208e+01  6.658e-01 -18.148   <2e-16 ***
income       1.858e-05  7.573e-06   2.454   0.0141 *
balance      6.053e-03  3.467e-04  17.457   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1457.0  on 4999  degrees of freedom
Residual deviance:  734.4  on 4997  degrees of freedom
AIC: 740.4

Number of Fisher Scoring iterations: 8
```

3. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the "default" category if the posterior probability is greater than 0.5.

Ans:

```
probs <- predict(fit.glm, newdata = Default[-train, ], type =
"response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
```

4. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

Ans:

```
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0286
```

We have a 2.86% *test error rate with the validation set approach.*

c. Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0236
```

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.028
```

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0268
```

As shown in above, the validation estimate of the test error rate can be variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.

d. Now consider a logistic regression model that predicts the probability of "default" using "income", "balance", and a dummy variable for "student". Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for "student" leads to a reduction in the test error rate.

Ans:

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train)
> pred.glm <- rep("No", length(probs))
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0264
```

As shown in above, *it doesn't seem that adding the "student" dummy variable leads to a reduction in the validation set estimate of the test error rate.*

Q9. We continue to consider the use of a logistic regression model to predict the probability of "default" using "income" and "balance" on the "Default" data set. In particular, we will now computes estimates for the standard errors of the "income" and "balance" logistic regression coefficients in two different ways : (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.

a. Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with "income" and "balance" in a multiple logistic regression model that uses both predictors.

Ans:

```
> set.seed(1)
> attach(Default)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
> summary(fit.glm)

Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4725  -0.1444  -0.0574  -0.0211   3.7245

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8
```

The glm() estimates of the standard errors for the coefficients β0, β1 and β2 are respectively 0.4348, 4.985$*10^{-6}$*and 2.274*$*10^{-4}$.

b. Write a function, boot.fn(), that takes as input the "Default" data set as well as an index of the observations, and that outputs the coefficient estimates for "income" and "balance" in the multiple logistic regression model.

Ans:

```
boot.fn <- function(data, index) {
    fit <- glm(default ~ income + balance, data = data, family =
"binomial", subset = index)
    return (coef(fit))
}
```

c. Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for "income" and "balance".

Ans:

```
library(boot)
```

```
> boot(Default, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01 -8.008379e-03 4.239273e-01
t2*  2.080898e-05  5.870933e-08 4.582525e-06
t3*  5.647103e-03  2.299970e-06 2.267955e-04
```

As shown in above, *the bootstrap estimates of the standard errors for the coefficients β0, β1 and β2 are respectively 0.4239273, 4.582525*$*10^{-6}$*and 2.267955*$*10^{-4}$

d. Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

Ans: *Although the estimated standard errors obtained by standard formula is slightly bigger than by bootstrap, but the two methods are actually very close. The difference for β0, β1 and β2 between two methods are approximately (0.4348-0.4239273)/0.4348=2.5%, (4.985-4.582525)/4.985=8.07% and (2.274-2.267955)/2.274=0.27%*