**DS 502 –Statistical Methods for Data Science**

**Assignment 2**

**Submitted By:**

Siddhant Bhavsar and Jay Trivedi

# Assignment 2

**1. Using 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.**

4.2 equation is $p(X) = (e^{\beta 0 + \beta 1X}) / (1 + e^{\beta 0 + \beta 1X})$

4.3 equation is $p(X) / (1-p(X)) = e^{\beta 0 + \beta 1X}$

$p(X) = (e^{\beta 0 + \beta 1X}) / (1 + e^{\beta 0 + \beta 1X})$

Subtract both side from 1,

$1 - p(X) = 1 - (e^{\beta 0 + \beta 1X}) / (1 + e^{\beta 0 + \beta 1X})$

$1 – p(X) = 1 / (1 + e^{\beta 0 + \beta 1X})$

Now, taking reciprocal on both sides,

$1/ (1 – p(X)) = 1 + e^{\beta 0 + \beta 1X}$

Now multiplying both sides by p(X),

$p(X) / (1 – p(X)) = ((e^{\beta 0 + \beta 1X}) / (1 + e^{\beta 0 + \beta 1X})) * (1 + e^{\beta 0 + \beta 1X})$

So,

$p(X) / (1 – p(X)) = e^{\beta 0 + \beta 1X}$

Hence, Proved.

**2.We now examine the differences between LDA and QDA.**

a. **If the Bayes decision boundary is linear, do we except LDA or QDA to perform better on the training set? On the test set?**
QDA will perform better on training data as the Bayes decision boundary is linear and QDA is more flexible, whereas LDA will perform better than QDA on the test set. LDA by definition is

ought to have lower variance than QDA hence will perform better on test data as QDA might overfit based on the linear Bayes decision boundary.

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?**
In this situation if the Bayes decision boundary is non-linear, then the QDA will perform better on both the set, i.e., training set as well as the test set. LDA usually perform bad on the non-linear Bayes decision boundary as it underfits the data and ends up having on more bias.

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?**
We expect the test prediction accuracy of QDA relative to LDA to improve because QDA has high flexibility which tends to create a good fit. This is since adding more sample reduces variance which is good for QDA as problem often with QDA is variance and not bias thus QDA will end up giving better result.

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.**
False. The variance by using a flexible method such as QDA leads to an inferior test error rate for fewer sample points. It might even cause overfit.

**3.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 and test data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?**

On basis of given data, we can infer that test error rate for K=1 is 36%. This is due to the fact that average error is 18% and for K=1 training error rate is 0% because for any training observation the class is the sample itself making test error rate 36%.

In case of logistic regression, we have lesser test error, thus we should prefer logistic regression in this case. Another reason for preferring logistic regression is for the fact that adding a new sample to the training data will change the whole decision boundary significantly, in other words the variance of this model is very high. Thus, it’s better to use logistic regression model here.

**4.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?**

Code:

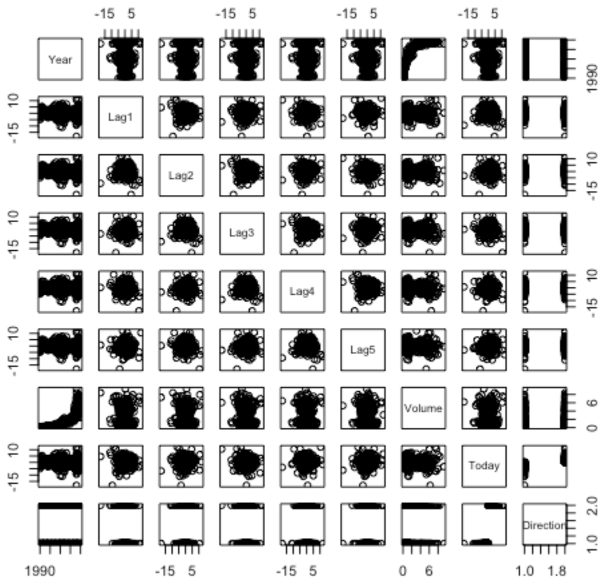
```
library(ISLR)
summary(Weekly)
head(Weekly)
pairs(Weekly)
plot(Weekly$Year,Weekly$Volume, xlab = 'Years', ylab = 'Volume', main = 'Plot of Years vs Volume' )
```

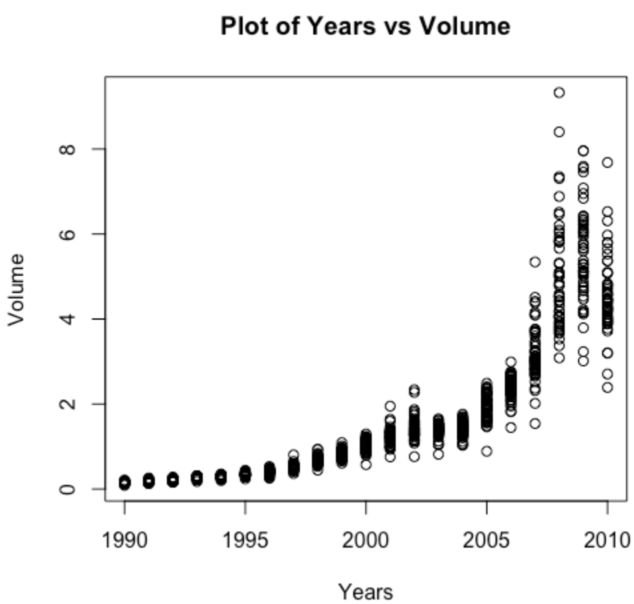
Output:

```
> library(ISLR)
> summary(Weekly)
     Year           Lag1              Lag2              Lag3              Lag4              Lag5
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580   1st Qu.: -1.1580   1st Qu.: -1.1660
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410   Median :  0.2380   Median :  0.2340
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472   Mean   :  0.1458   Mean   :  0.1399
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4050
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
    Volume            Today          Direction
 Min.   :0.08747   Min.   :-18.1950   Down:484
 1st Qu.:0.33202   1st Qu.: -1.1540   Up  :605
 Median :1.00268   Median :  0.2410
 Mean   :1.57462   Mean   :  0.1499
 3rd Qu.:2.05373   3rd Qu.:  1.4050
 Max.   :9.32821   Max.   : 12.0260


> head(Weekly)
  Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume.  Today   Direction
1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

Plots:




From the plot we can conclude that the Year Vs Volume shows a logarithmic pattern.

**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?**

Code:

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

Output:

```
> weeklyglm <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,family =
"binomial", data=Weekly)
> summary(weeklyglm)

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
```

Lag2 seems to be the only predictor which is statistically significant. Even its Estimate coefficient is 0.058 which indicates increase of lag2 value by 1 will affect the result by exponential term e^0.058.

**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.**

Code:

```
 wall <- predict(weeklyglm, type = 'response')
pred <- ifelse(wall > 0.5, "Up","Down")
confusion_matrix <- table(pred, Weekly$Direction)
confusion_matrix
```

Output:

```
pred   Down  Up
Down   54    48
Up     430   557
```

From the Output, we can conclude that the percentage of true positive value is (557/605) *100 = 92.066% which is quite high compared to the percentage of true negative values which is (54/484) *100 = 11.15%. That means, the changes of prediction of market going up is quite higher than the prediction of downfall in the market.

**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).**

Code:

```
early <- Weekly[Weekly$Year<2009,]
later <- Weekly[Weekly$Year>2008,]
earlyglm <- glm(Direction~Lag2, family = "binomial", data= early)
summary(earlyglm)

laterpredict <- predict(earlyglm, newdata = later, type="response")
laterdirs <- Weekly$Direction[Weekly$Year>2008]
summary(laterpred)
laterpred <- rep("Down", 104)
laterpred[laterpredict>0.5] <- "Up"
table(laterpred, laterdirs)
```

Output:

```
> summary(earlyglm)

Call:
glm(formula = Direction ~ Lag2, family = "binomial", data = early)

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
```

Code:

```
laterpredict <- predict(earlyglm, newdata = later, type="response")
laterdirs <- Weekly$Direction[Weekly$Year>2008]
summary(laterpred)
```

Output:

```
             Length     Class      Mode
              104      character character
```

The length of laterpred dataframe is 104, so we assign them as Down.

Code:

```
laterpred <- rep("Down", 104)
laterpred[laterpredict>0.5] <- "Up"
table(laterpred, laterdirs)
```

Output:

```
                Laterdirs
laterpred.   Down Up
    Down        9     5
    Up           34    56
```

From this output, we can conclude that the percentage of true positive value is (56/61) *100 = 91.80% which is quite high compared to the percentage of true negative values which is (9/43) *100 = 20.93%. That means, the changes of prediction of market going up is quite higher than the prediction of downfall in the market.

Total true correct predictions using logistic regression= 65/104*100 = 62.50%

**e. Repeat d. using LDA**
Code:

```
library(MASS)
earlylda <- lda(Direction~Lag2, family = "binomial", data= early)
laterpredictlda <- predict(earlylda, newdata = later, type="response")
laterdirslda <- Weekly$Direction[Weekly$Year>2008]
table(laterpredictlda$class, laterdirslda)
```

Output:

```
          laterdirslda
           Down  Up
  Down      9.    5
  Up        34    56
```

From this output, we can conclude that the percentage of true positive value is (56/61) *100 = 91.80% which is quite high compared to the percentage of true negative values which is (9/43) *100 = 20.93%. That means, the changes of prediction of market going up is quite higher than the prediction of downfall in the market.
Total true correct predictions using LDA = 65/104*100 = 62.50%

f. **Repeat d. using QDA**
Code:

```
earlyqda <- qda(Direction~Lag2, family = "binomial", data= early)
laterpredictqda <- predict(earlyqda, newdata = later, type="response")
laterdirsqda <- Weekly$Direction[Weekly$Year>2008]
table(laterpredictqda$class, laterdirsqda)
```

Output:

```
          laterdirsqda
            Down Up
  Down       0    0
  Up         43   61
```

QDA implies all the data is going up. Thus we can say QDA has the worst error rate which is 43/104*100 = 41.35%
Total true correct predictions using QDA = 61/104*100 = 58.65%

g. **Repeat d. using KNN and k = 1**
Code:

```
library(class)
```

```
set.seed(1)
earlyset <- (Weekly$Year<2009)
laterset <- (Weekly$Year>2008)
earlylag2 <- Weekly[earlyset,"Lag2",drop=F]
laterlag2 <- Weekly[laterset,"Lag2",drop=F]
predknn <- knn(earlylag2, laterlag2, early$Direction, k = 1)
table(predknn, laterdirs)
```

Output:

```
           laterdirs
predknn   Down Up
   Down     21  30
   Up       22  31
```

We can state that the percentage of correct prediction in this case is 52/104*100 = 50%. That means the error rate too is 50%. Through this model we can also say that the true prediction of market goes up is 31/61*100 = 50.82% right.
Total true correct predictions using KNN = 52/104*100 = 50%

h. **Which of these methods appear to provide best results on this data?**
LDA and Logistic regression gives us the best result rate which is 62.50%, followed by QDA which is 58.65% and at last KNN for k =1 which is 50%.

i. **Experiment with different combinations of predictors, including transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that provides the best results on the held-out data. Note that you should also experiment with values for K in the KNN classifier.**

Logistic Regression using Lag2+Lag3+lag4+Volume.
Code:

```
earlyglm2 <- glm(Direction~Lag2+Lag3+Lag4+Volume, family = "binomial", data=
early)
summary(earlyglm2)
laterpredict2 <- predict(earlyglm2, newdata = later, type="response")
laterdirs2 <- Weekly$Direction[Weekly$Year>2008]
laterpred2 <- rep("Down", 104)
laterpred2[laterpredict2>0.5] <- "Up"
table(laterpred2, laterdirs2)
```

Output:

```
                laterdirs2
laterpred2   Down  Up
     Down       26     34
     Up          17     27
```

Here we can see that the true prediction percentage is 53/104*100 = 50.96% for logistic regression using Lag2+Lag3+Lag4+Volume.

Performing LDA using Lag2+Lag3+Lag4+Volume
Code:

```
library(MASS)
earlylda2 <- lda(Direction~Lag2+Lag3+Lag4+Volume, family = "binomial", data=
early)
laterpredictlda2 <- predict(earlylda2, newdata = later, type="response")
laterdirslda2 <- Weekly$Direction[Weekly$Year>2008]
table(laterpredictlda2$class, laterdirslda2)
```

Output:

```
         laterdirslda2
           Down Up
  Down      26   35
  Up        17   26
```

Here we can see that the true prediction percentage is 52/104*100 = 50 % for LDA using Lag2+Lag3+Lag4+Volume.

Performing QDA by interacting Lag3 by Lag4
Code:

```
earlyqda2 <- qda(Direction~Lag3:Lag4, family = "binomial", data= early)
laterpredictqda2 <- predict(earlyqda2, newdata = later, type="response")
laterdirsqda2 <- Weekly$Direction[Weekly$Year>2008]
table(laterpredictqda2$class, laterdirsqda2)
```

Output:

```
        laterdirsqda2
           Down  Up
  Down        0   0
  Up         43  61
```

Here we can see that the true prediction percentage is 62/104*100 = 59.62 % for QDA using Lag3:Lag4.

Performing KNN for k =10

Code:

```
predknn2 <- knn(earlylag2, laterlag2, early$Direction, k = 10)
table(predknn2, laterdirs)
```

Output:

```
          laterdirs
predknn2 Down Up
   Down    17  19
   Up      26  42
```

Here we performed KNN with k=10 on Lag2, it comes out that the true prediction percentage in this case is 59/104*100=56.73%.

Among, all the operations that we performed it seems that QDA using Lag3:Lag4 gave us best true prediction ratio when the market is going up which is 59.62% true prediction.

**5. 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.**

Code:

```
summary(Auto)
auto = Auto
auto$mpg01 <- ifelse(auto$mpg > median(auto$mpg),1,0)
auto$mpg01
head(auto)
auto
```

Output:

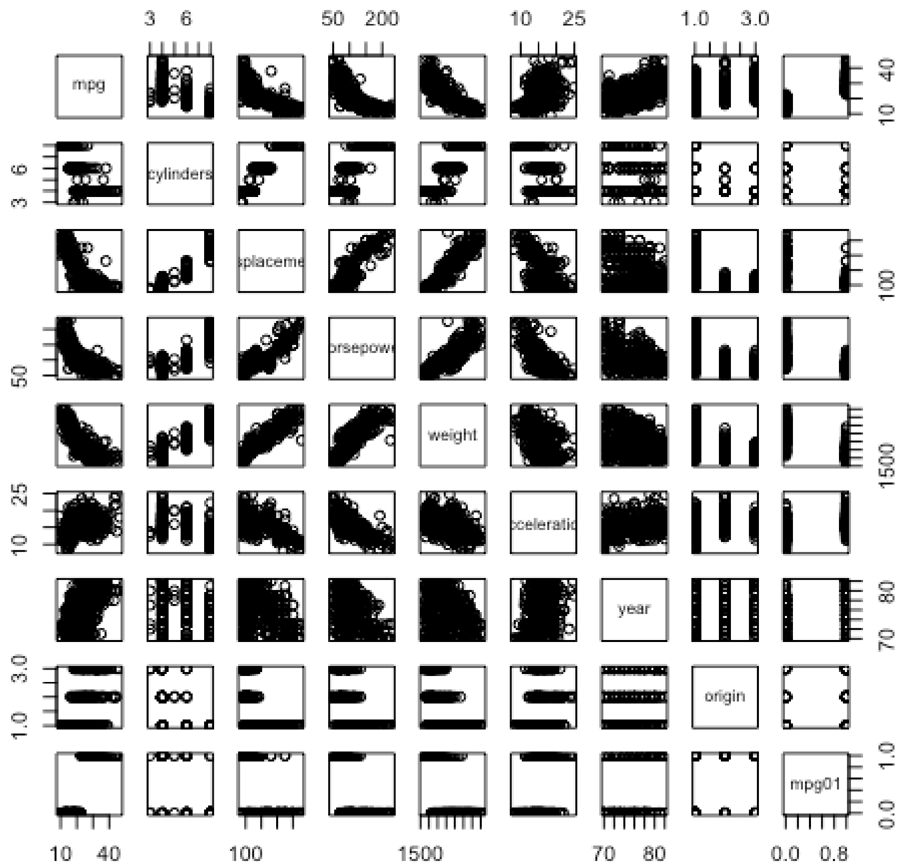
```
> head(auto)
  mpg cylinders displacement horsepower weight acceleration year origin          name             mpg01
1  18     8        307          130       3504      12.0       70     1 chevrolet chevelle malibu    0
2  15     8        350          165       3693      11.5       70     1       buick skylark 320      0
3  18     8        318          150       3436      11.0       70     1      plymouth satellite      0
4  16     8        304          150       3433      12.0       70     1         amc rebel sst        0
5  17     8        302          140       3449      10.5       70     1          ford torino         0
6  15     8        429          198       4341      10.0       70     1       ford galaxie 500       0
```

**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.**

Code:

```
pairs(auto)
```

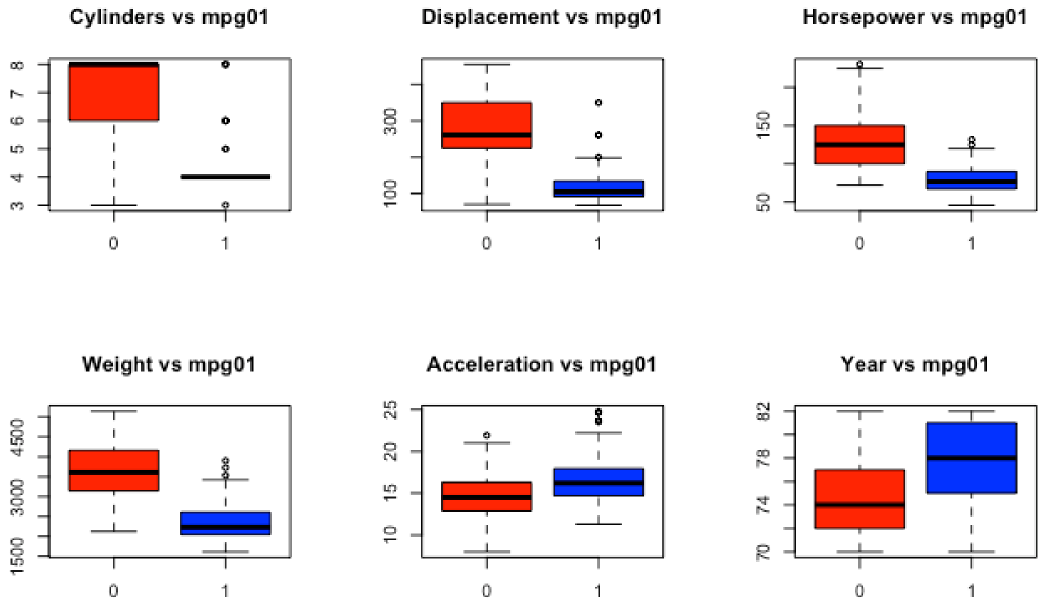
Plot:




Code:

```
par(mfrow=c(2,3))
boxplot(cylinders ~ mpg01, data = auto, main = "Cylinders vs mpg01", col=c("red",
"blue"))
boxplot(displacement ~ mpg01, data = auto, main = "Displacement vs mpg01",
col=c("red", "blue"))
boxplot(horsepower ~ mpg01, data = auto, main = "Horsepower vs mpg01",
col=c("red", "blue"))
boxplot(weight ~ mpg01, data = auto, main = "Weight vs mpg01", col=c("red",
"blue"))
boxplot(acceleration ~ mpg01, data = auto, main = "Acceleration vs mpg01",
col=c("red", "blue"))
boxplot(year ~ mpg01, data = auto, main = "Year vs mpg01", col=c("red", "blue"))
```

Plot:




Displacement, weight, cylinder, Horsepower have a strong correlation in predicting mpg01

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

Code:

```
set.seed(1)
a <- rnorm(nrow(auto))
summary(a)
test <- a > quantile(a,0.75)
train <- !test
trainset <- auto[train,]
testset <- auto[test,]
head(trainset)
head(testset)
```

Output:

```
> head(trainset)
  mpg cylinders displacement horsepower weight acceleration year origin                    name mpg01
1  18         8          307        130   3504         12.0   70      1 chevrolet chevelle malibu     0
2  15         8          350        165   3693         11.5   70      1         buick skylark 320     0
3  18         8          318        150   3436         11.0   70      1        plymouth satellite     0
5  17         8          302        140   3449         10.5   70      1               ford torino     0
6  15         8          429        198   4341         10.0   70      1          ford galaxie 500     0
7  14         8          454        220   4354          9.0   70      1          chevrolet impala     0
> head(testset)
   mpg cylinders displacement horsepower weight acceleration year origin               name mpg01
4   16         8          304        150   3433         12.0   70      1        amc rebel sst     0
8   14         8          440        215   4312          8.5   70      1     plymouth fury iii     0
11  15         8          383        170   3563         10.0   70      1  dodge challenger se     0
```

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 24 | 4 | 113 | 95 | 2372 | 15.0 | 70 | 3 | toyota corona mark ii 1 |
| 18 | 21 | 6 | 200 | 85 | 2587 | 16.0 | 70 | 1 | ford maverick 0 |
| 19 | 27 | 4 | 97 | 88 | 2130 | 14.5 | 70 | 3 | datsun pl510 1 |

**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?**

Code:

```
lda1 <- lda(mpg01 ~ displacement+horsepower+weight+cylinders, data=trainset)
ldapred <- predict(lda1, testset)
table(testset$mpg01, ldapred$class)
mean(testset$mpg01 == ldapred$class)
```

Output:

```
> table(testset$mpg01, ldapred$class)

     0   1
 0  36   8
 1   3   51
> mean(testset$mpg01 == ldapred$class)
[1] 0.8877551
```

The test error of the model is 100 – 88.78 = 11.22%.

**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?**

Code:

```
qda1 <- qda(mpg01 ~ displacement+horsepower+weight+cylinders, data=trainset)
qdapred <- predict(qda1, testset)
table(testset$mpg01, qdapred$class)
mean(testset$mpg01 == qdapred$class)
```

Output:

```
> table(testset$mpg01, qdapred$class)

       0.    1
 0     37    7
 1     7     47
> mean(testset$mpg01 == qdapred$class)
[1] 0.8571429
```

The test error of this model is 100 – 85.71 = 14.29%

**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?**

Code:

```
earlyglm3 <- glm(mpg01~displacement+horsepower+weight+cylinders, family =
"binomial", data= trainset)
summary(earlyglm3)
laterpredict3 <- predict(earlyglm3, newdata = testset, type="response")
length(laterpredict3)
laterpred3 <- rep("0", 98)
laterpred3[laterpredict3>0.5] <- "1"
table(laterpred3, testset$mpg01)
mean(laterpred3 != testset$mpg01)
```

Output:

```
> table(laterpred3, testset$mpg01)

laterpred3  0  1
       0    36  6
       1     8 48
> mean(laterpred3 == testset$mpg01)
[1] 0.8571429
> mean(laterpred3 != testset$mpg01)
[1] 0.1428571
```

The test error rate of this model is 14.28%

**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 performs the best on this data set?**

Code:

```
set.seed(1)
Autotrain = trainset[,c("displacement","horsepower","weight","acceleration")]
Autotest =  testset[,c("displacement","horsepower","weight","acceleration")]
predknn=knn(Autotrain,Autotest,trainset$mpg01,k=1)
table(predknn,testset$mpg01)
mean(predknn!=testset$mpg01)

predknn=knn(Autotrain,Autotest,trainset$mpg01,k=2)
table(predknn,testset$mpg01)
mean(predknn!=testset$mpg01)

predknn=knn(Autotrain,Autotest,trainset$mpg01,k=3)
table(predknn,testset$mpg01)
mean(predknn!=testset$mpg01)

predknn=knn(Autotrain,Autotest,trainset$mpg01,k=4)
```

```
table(predknn,testset$mpg01)
mean(predknn!=testset$mpg01)

predknn=knn(Autotrain,Autotest,trainset$mpg01,k=5)
table(predknn,testset$mpg01)
mean(predknn!=testset$mpg01)

predknn=knn(Autotrain,Autotest,trainset$mpg01,k=6)
table(predknn,testset$mpg01)
mean(predknn!=testset$mpg01)
```

Output:

```
> set.seed(1)
> Autotrain = trainset[,c("displacement","horsepower","weight","acceleration")]
> Autotest =  testset[,c("displacement","horsepower","weight","acceleration")]
> predknn=knn(Autotrain,Autotest,trainset$mpg01,k=1)
> table(predknn,testset$mpg01)

predknn  0  1
      0 35  5
      1  9 49
> mean(predknn!=testset$mpg01)
[1] 0.1428571
> predknn=knn(Autotrain,Autotest,trainset$mpg01,k=2)
> table(predknn,testset$mpg01)

predknn  0  1
      0 34  5
      1 10 49
> mean(predknn!=testset$mpg01)
[1] 0.1530612
> predknn=knn(Autotrain,Autotest,trainset$mpg01,k=3)
> table(predknn,testset$mpg01)

predknn  0  1
      0 36  4
      1  8 50
> mean(predknn!=testset$mpg01)
[1] 0.122449
> predknn=knn(Autotrain,Autotest,trainset$mpg01,k=4)
> table(predknn,testset$mpg01)

predknn  0  1
```

```
     0 36  5
     1  8 49
> mean(predknn!=testset$mpg01)
[1] 0.1326531
> predknn=knn(Autotrain,Autotest,trainset$mpg01,k=5)
> table(predknn,testset$mpg01)

predknn  0  1
     0 36  7
     1  8 47
> mean(predknn!=testset$mpg01)
[1] 0.1530612
> predknn=knn(Autotrain,Autotest,trainset$mpg01,k=6)
> table(predknn,testset$mpg01)

predknn  0  1
     0 37  6
     1  7 48
> mean(predknn!=testset$mpg01)
[1] 0.1326531
```

In this KNN model, test error rate is :
For k=1, Test Error Rate = 14.29%
For k=2, Test Error Rate = 15.30%
For k=3, Test Error Rate = 12.24%
For k=4, Test Error Rate = 13.26%
For k=5, Test Error Rate = 15.31%
For k=6, Test Error Rate = 13.27%

**6. 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).**

HW 2 Question 6

To prove: $\alpha = \dfrac{\sigma_X^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$

where $\sigma_X^2 = Var(X)$, $\sigma_Y^2 = Var(Y)$ & $\sigma_{XY}^2 = Cov(X,Y)$

Solution:

properties

1. $Var(X+Y) = Var(X) + Var(Y) + 2\,Cov(X,Y)$
2. $Var(aX) = a^2 Var(X)$
3. $Cov(aX, Y) = a\,Cov(X,Y)$

We have $Var(\alpha X + (1-\alpha)Y)$

$= Var(\alpha X) + Var((1-\alpha)Y) + 2\,Cov(\alpha X, (1-\alpha)Y)$ by 1

$= \alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)\,Cov(X,Y)$

$\hookrightarrow$ by 2 & 3

$= \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\alpha(1-\alpha)\,\sigma_{XY}^2$ by definition

To minimize variance.

$\dfrac{d\,Var(\alpha X + (1-\alpha)Y)}{d\alpha} = 0$

$\therefore\ 2\alpha\sigma_X^2 + 2\sigma_Y^2(1-\alpha)(-1) + 2\sigma_{XY}(-2\alpha+1) = 0$

$\alpha\sigma_X^2 + \sigma_Y^2(\alpha-1) + \sigma_{XY}(-2\alpha+1) = 0$

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

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

Hence proved.

**7. 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.**

The probability that an observation is chosen is 1/n as every observation is equally like and the total probability is 1. Thus, the probability that a particular observation is not the jth observation is given by

1 – 1/n

**(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?**

The probability of second bootstrap observation is not the jth observation is still (1-1/n) as bootstrap performs repeated sampling with replacement.

**(c) Argue that the probability that the jth observation is not in the bootstrap sample is (1 − 1/n) ^ n.**

As stated in questions above the probability of not choosing a bootstrap sample is (1 – 1/n). Bootstrap by definition samples with replacement, hence new samples are independent of already chosen values.

So, when we are doing it n times the probability that jth observation is not in bootstrap sample simply gets multiplied. Hence the probability is given by:

(1-1/n) (1-1/n) (1-1/n) ……. n times = (1 -1/n) ^n.

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

Probability that jth observation is in bootstrap sample is given by:
1 – (1 -1/n) ^n = 1 – (1 – 1/5) ^5

= 0.672

**(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?**
Probability that jth observation is in bootstrap sample is given by:
1 – (1 -1/n) ^n = 1 – (1 – 1/100) ^100

= 0.634

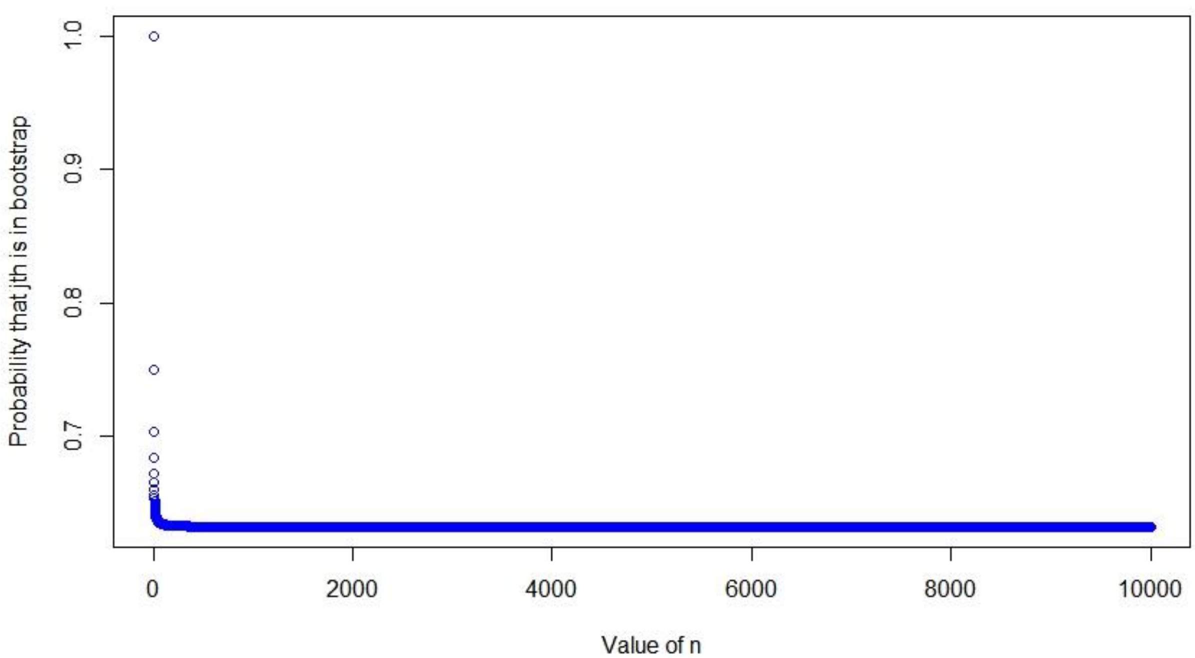
**(f) When n =10, 000, what is the probability that the jth observation is in the bootstrap sample?**
Probability that jth observation is in bootstrap sample is given by:
1 – (1 -1/n) ^n = 1 – (1 – 1/10000) ^10000

= 0.632

**(g) Create a plot that displays, for each integer value of n from 1 to 100, 000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.**

```
Code:
pr = function(n) return(1 - (1 - 1/n)^n)
x = 1:10000
plot(x, pr(x), col = 'blue',xlab = "Value of n", ylab = "Probability that jth is in bootstrap")
```

Output




The plot is exponentially decreasing and reaches value of 0.632 which perhaps acts as an asymptote.

**(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.**

**> store=rep (NA, 10000) > for (i in 1:10000) {store[i]=sum (sample (1:100 , rep =TRUE)==4) >0**

**} > mean(store)**

**Comment on the results obtained.**

```
> set.seed(50)
> store=rep (NA, 10000)
> for (i in 1:10000)
+ store[i] <- sum (sample(1:100, rep=TRUE)==4) > 0
> mean(store)
[1] 0.6347
```

The code above samples 100 sample from 10000 and checks is 4 exists in the sample. As given by the plot in question before it should reach an asymptote of 0.632 as n tends to infinity limit of the function tends to 0.632.

Here as it still at 10000, thus limit has not been reached and the code gives value of 0.6347 for seed = 50.
Note seed has been set for reproducibility of the result

**8. In Chapter 4, we used logistic 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.**
Code

```
summary(Default)
attach(Default)
set.seed(50)
lr_m = glm(default~income+balance,family = binomial,data=Default)
summary(lr_m)
```

Output

```
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:**

**i. Split the sample set into a training set and a validation set.**

Code

```
data <- sample(nrow(Default),nrow(Default)*0.8)
train <- Default[data,]
val <- Default[-data,]
summary(train)
summary(val)
```

Output

```
# train
default    student      balance          income
 No :7724   No :5635   Min.   :   0.0   Min.   :  772
 Yes: 276   Yes:2365   1st Qu.: 477.9   1st Qu.:21270
                       Median : 822.9   Median :34470
                       Mean   : 835.3   Mean   :33478
                       3rd Qu.:1167.3   3rd Qu.:43784
                       Max.   :2502.7   Max.   :73554
# validation
default    student      balance          income
 No :1943   No :1421   Min.   :   0.0   Min.   : 2703
 Yes:  57   Yes: 579   1st Qu.: 501.1   1st Qu.:21573
                       Median : 825.1   Median :35054
                       Mean   : 835.5   Mean   :33673
                       3rd Qu.:1163.8   3rd Qu.:43891
                       Max.   :2654.3   Max.   :70701
```

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

Code

```
lr_train <- glm(default~income+balance,family = binomial,data=train)
# summary of model
summary(lr_train)
```

Output

```
# summary of model
Call:
glm(formula = default ~ income + balance, family = binomial,
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5190  -0.1424  -0.0554  -0.0198   3.7343

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.167e+01  4.870e-01 -23.968  < 2e-16 ***
income       2.045e-05  5.470e-06   3.738 0.000185 ***
balance      5.761e-03  2.571e-04  22.407  < 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: 2400.8  on 7999  degrees of freedom
Residual deviance: 1279.1  on 7997  degrees of freedom
AIC: 1285.1
Number of Fisher Scoring iterations: 8
```

**iii. 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.**

Code

```
#posterior probability
prob <- predict(lr_train,val,type = "response")
result <- ifelse(prob > 0.5,"Yes","No")
# confusion matrix
table(val$default,result,dnn=c("Actual","Predicted"))
```

Output

```
    Predicted
Actual   No   Yes
  No   1935    8
  Yes   36     21
```

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

Code

```
# error
round(mean(val$default!=result),digits = 3)
```

Output

```
[1] 0.022
```

**(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.**

Code

```
set.seed(50)

##### First iteration 20-80
data <- sample(nrow(Default),nrow(Default)*0.2)
train <- Default[data,]
val <- Default[-data,]
lr_train <- glm(default~income+balance,family = binomial,data=train)
prob <- predict(lr_train,val,type = "response")
result <- ifelse(prob > 0.5,"Yes","No")
round(mean(val$default!=result),digits = 3)

##### Second Iteration 50-50
data <- sample(nrow(Default),nrow(Default)*0.5)
train <- Default[data,]
val <- Default[-data,]
lr_train <- glm(default~income+balance,family = binomial,data=train)
prob <- predict(lr_train,val,type = "response")
result <- ifelse(prob > 0.5,"Yes","No")
round(mean(val$default!=result),digits = 3)
```

```
##### Third Iteration 80-20
data <- sample(nrow(Default),nrow(Default)*0.8)
train <- Default[data,]
val <- Default[-data,]
lr_train <- glm(default~income+balance,family = binomial,data=train)
prob <- predict(lr_train,val,type = "response")
result <- ifelse(prob > 0.5,"Yes","No")
round(mean(val$default!=result),digits = 3)
```

Output

```
#### First Iteration error
[1] 0.026

#### Second Iteration error
[1] 0.024

#### Third Iteration error
[1] 0.022
```

As we observe from the iterations the more data is given to train the model the error rate decreases, this is not always true as there is always a chance that a random validation data will not show this characteristic. It so turns out in our iterations that difference between error is constant, but it is not the case always. This is probably due to random seed that has been 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.**

Code

```
set.seed(50)
data <- sample(nrow(Default),nrow(Default)*0.8)
train <- Default[data,]
val <- Default[-data,]
lr_train <- glm(default~income+balance+student,family = binomial,data=train)
prob <- predict(lr_train,val,type = "response")
result <- ifelse(prob > 0.5,"Yes","No")
round(mean(val$default!=result),digits = 3)

table(val$default,result,dnn=c("Actual","Predicted"))
```

Output

```
### error
[1] 0.021

#### Confusion matrix

     Predicted
Actual   No  Yes
  No  1936    7
  Yes   35   22
```

As we see from the result the error rate slightly decreases with the addition of student variable. As we can see in the confusion matrix this is due to increase of 1 in both true positive and true negative. The error decrease is very small so perhaps adding the student variable doesn't give significantly more information to the model.

**9. 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 compute 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.**

Code

```
set.seed(50)
###
lr_train <- glm(default~income+balance,family = binomial, data=Default)
# summary of model and standard error
summary(lr_train)$coef[,2]
```

Output

```
#### error
(Intercept)      income      balance
4.347564e-01 4.985167e-06 2.273731e-04
```

Thus, standard error coefficients associated with income and balance are 4.985167e-06 and 2.273731e-04 respectively.

**(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.**

Code

```
boot.fn = function(data, index) return(coef(glm(default~income+balance,family = binomial,data=
data, subset = index)))
```

**(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.**

Code

```
library(boot)
boot(Default, boot.fn, 100)
```

Output

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Default, statistic = boot.fn, R = 100)


Bootstrap Statistics :
       original       bias     std. error
t1* -1.154047e+01 -1.431166e-02 4.963704e-01
t2*  2.080898e-05 -3.856758e-07 5.193719e-06
t3*  5.647103e-03  1.287655e-05 2.698000e-04
```

Thus, standard errors of the logistic regression coefficients for income and balance are 5.193719e-06 and 2.698000e-04 respectively.

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

The answer from using bootstrap and glm function are similar but off by a bit around 2e-07 for income and 5e-05 for balance, but this will not be a problem as the variation is very small.