

## Chapter – 4

### ISLR 4.7

**Q5. 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?**

We assume that QDA will perform better on the training set if the Bayes decision boundary is linear because its greater flexibility might result in a better match. Since QDA might overfit the linearity on the Bayes decision boundary, we anticipate LDA to perform better on the test set than QDA.

- 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?**

We assume that QDA will perform better on both the training and test sets if the Bayes decision boundary is non-linear.

- 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?**

If the training set is very large, QDA is advised so that the classifier's variance is not a major worry (QDA is more flexible than LDA, hence it has more variance).

- 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. A more flexible approach, such as QDA, may produce overfitting when there are fewer sample points, which could result in a worse test error rate.

**Q6.** Suppose we collect data for a group of students in a statistics class with variables  $X_1$  = hours studied,  $X_2$  = 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 hours and has an undergrad GPA of 3.5 gets an A in the class.

It suffices to plug in the beta values in the equation for predicted probability

$$\hat{P}(x) = \frac{e^{-6 + 0.05x_1 + x_2}}{1 + e^{-6 + 0.05x_1 + x_2}}$$

$$\text{where } x_1 = 40 \quad \& \quad x_2 = 3.5$$

$$= \frac{e^{-6 + 0.05(40) + 3.5}}{1 + e^{-6 + 0.05(40) + 3.5}}$$

$$= 0.3775$$

- 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?

the equation for predicted probability

$$\frac{e^{-6 + 0.05x_1 + 3.5}}{1 + e^{-6 + 0.05x_1 + 3.5}} = 0.5,$$

which is equal to

$$e^{-6 + 0.05x_1 + 3.5} = 1$$

taking log on both sides

$$x_1 = \frac{2.5}{0.05} = 50$$

**Q8.** 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?

for KNN with  $K=1$ , the training error rate is 0 as all the training data will be classified correctly. Hence for the KNN with average error rate of 18%, the test error rate is 36% which makes the logistic regression with test error rate 30% is better for classification.

# R Notebook

13. 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.

```
library(ISLR)
```

```
head(Weekly)
```

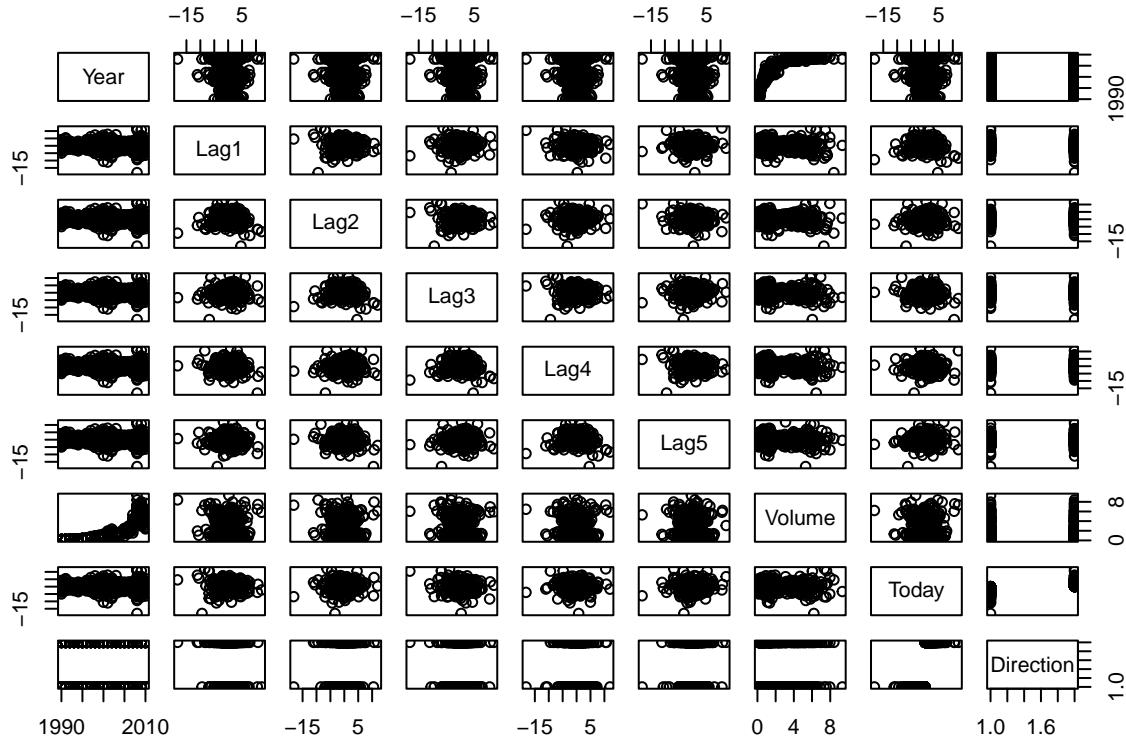
```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5   Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484  0.1549760 -0.270     Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229  0.1485740 -2.576     Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936  0.1598375  3.514      Up
## 4 1990  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
```

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

```
summary(Weekly)
```

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

```
pairs(Weekly)
```

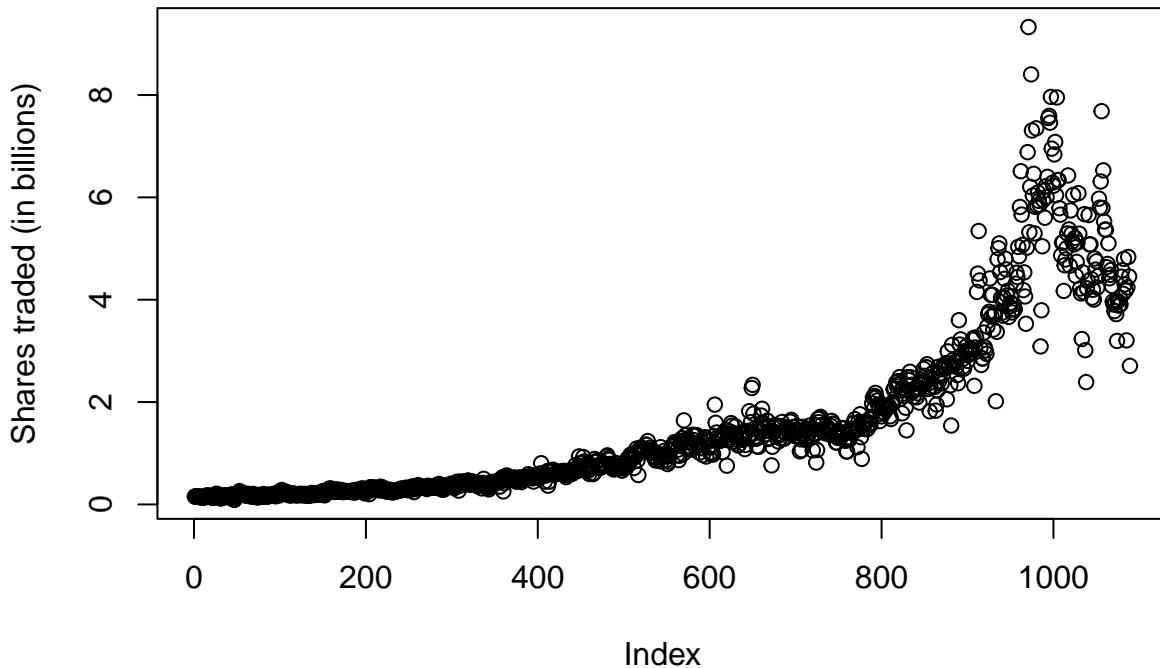


```
cor(Weekly[, -9])
```

```
##          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
```

There seems to be a correlation between year and volume. There are no other noticeable patterns.

```
plot(Weekly$Volume, ylab = "Shares traded (in billions)")
```



- 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?

```
attach(Weekly)
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

```

With a  $\text{Pr}(>|z|) = 3\%$ , Lag 2 seems to have some statistical significance.

- 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")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Direction)

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

mean(glm.pred == Weekly$Direction)

```

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

The percentage of accurate predictions is  $(54 + 557)/(54 + 557 + 48 + 430) = 56.1\%$ .  $557/(557+48) = 92.1$  percent of the time the logistic regression is accurate during market upswings. When the market rises, the logistic regression is frequently off by 11.2 percent, or  $54/(430+54)$  weeks.

- 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.0910 = Weekly[!train, ]
glm.fit = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit, Weekly.0910, type = "response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
Direction.0910 = Direction[!train]
table(glm.pred, Direction.0910)

```

```

##          Direction.0910
## glm.pred Down Up
##      Down    9  5
##      Up     34 56

mean(glm.pred == Direction.0910)

```

## [1] 0.625

e. Repeat (d) using LDA.

```

library(MASS)
lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.pred = predict(lda.fit, Weekly.0910)
table(lda.pred$class, Direction.0910)

```

```

##          Direction.0910
##          Down Up
##      Down    9  5
##      Up     34 56

```

```
mean(lda.pred$class == Direction.0910)
```

## [1] 0.625

f. Repeat (d) using QDA.

```

qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.class = predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)

```

```

##          Direction.0910
## qda.class Down Up
##      Down    0  0
##      Up     43 61

```

```
mean(qda.class == Direction.0910)
```

## [1] 0.5865385

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.0910)

```

```

##          Direction.0910
## knn.pred Down Up
##      Down   21 30
##      Up    22 31

mean(knn.pred == Direction.0910)

```

```

## [1] 0.5

```

h) Repeat (d) using naive Bayes.

- i. Which of these methods appears to provide the best results on this data?

Logistic regression and LDA methods provide similar test error rates.

- j. 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.

```

# Logistic regression with Lag2:Lag1
glm.fit = glm(Direction ~ Lag2:Lag1, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit, Weekly.0910, type = "response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
Direction.0910 = Direction[!train]
table(glm.pred, Direction.0910)

```

```

##          Direction.0910
## glm.pred Down Up
##      Down   1 1
##      Up    42 60

```

```

mean(glm.pred == Direction.0910)

```

```

## [1] 0.5865385

```

```

# LDA with Lag2 interaction with Lag1
lda.fit = lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train)
lda.pred = predict(lda.fit, Weekly.0910)
mean(lda.pred$class == Direction.0910)

```

```

## [1] 0.5769231

```

```

# QDA with sqrt(abs(Lag2))
qda.fit = qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = train)
qda.class = predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)

```

```

##          Direction.0910
## qda.class Down Up
##      Down   12 13
##      Up    31 48

mean(qda.class == Direction.0910)

## [1] 0.5769231

# KNN k =10
knn.pred = knn(train.X, test.X, train.Direction, k = 10)
table(knn.pred, Direction.0910)

##          Direction.0910
## knn.pred Down Up
##      Down   17 18
##      Up    26 43

mean(knn.pred == Direction.0910)

## [1] 0.5769231

# KNN k = 100
knn.pred = knn(train.X, test.X, train.Direction, k = 100)
table(knn.pred, Direction.0910)

##          Direction.0910
## knn.pred Down Up
##      Down    9 12
##      Up     34 49

mean(knn.pred == Direction.0910)

## [1] 0.5576923

```

## **Chapter – 5**

### **ISLR 5.4**

**Q3. We now review k-fold cross-validation.**

**a. Explain how k-fold cross-validation is implemented.**

By randomly dividing the set of  $n$  observations into  $k$  non-overlapping groups,  $k$ -fold cross-validation is carried out. Each of these groups serves as a training set, with the others serving as validation sets. By averaging the  $k$  resulting MSE estimations, the test error is estimated.

**b. What are the advantages and disadvantages of k-fold cross validation relative to:**

**i) The validation set approach?**

The validation set strategy is conceptually straightforward and simple to apply because it only requires splitting the training data into two sets. There are two negatives, though: 1. Depending on which observations are included in the training and validation sets, the estimate of the test error rate can vary greatly. 2. The test error rate for the model fit on the full data set may tend to be underestimated by the validation set error rate.

**ii) LOOCV?**

With  $k = n$ , LOOCV is a specific instance of  $k$ -fold cross-validation. LOOCV is the method that requires the greatest computing because the model needs to be fit  $n$  times. Additionally, compared to  $k$ -fold CV, LOOCV has reduced bias but higher variance.

**Q4. Suppose that we use some statistical learning method to make a prediction for the response  $Y$  for a particular value of the predictor  $X$ . Carefully describe how we might estimate the standard deviation of our prediction.**

We may use the bootstrap method to estimate the standard deviation of our forecast if we were to generate a prediction for the response  $Y$  using some statistical learning method for a certain value of the predictor  $X$ . The bootstrap method involves repeating collecting observations from the original data set (with replacement)  $B$  times for some large value of  $B$ , fitting a new model each time, and then calculating the RMSE of the estimates for all  $B$  models.

## R Notebook

*Q5. 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.*

```
library(ISLR)
summary(Default)

##   default    student      balance       income
##   No :9667    No :7056    Min.   : 0.0   Min.   : 772
##   Yes: 333   Yes:2944   1st Qu.: 481.7  1st Qu.:21340
##                                         Median : 823.6  Median :34553
##                                         Mean   : 835.4  Mean   :33517
##                                         3rd Qu.:1166.3  3rd Qu.:43808
##                                         Max.   :2654.3   Max.   :73554

attach(Default)

a. Fit a logistic regression model that uses “income” and “balance” to predict “default”.
set.seed(1)
glm.fit = glm(default ~ income + balance, data = Default, family = binomial)
summary(glm.fit)

##
## 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:

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

```
train = sample(dim(Default)[1], dim(Default)[1] / 2)
```

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

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

```
##
```

```
## Call:
```

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

```
##
```

```
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max  
## -2.5830 -0.1428 -0.0573 -0.0213  3.3395
```

```
##
```

```
## Coefficients:
```

```
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -1.194e+01 6.178e-01 -19.333 < 2e-16 ***  
## income      3.262e-05 7.024e-06  4.644 3.41e-06 ***  
## balance     5.689e-03 3.158e-04 18.014 < 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: 1523.8 on 4999 degrees of freedom
```

```
## Residual deviance: 803.3 on 4997 degrees of freedom
```

```
## AIC: 809.3
```

```
##
```

```
## 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.

```
glm.pred = rep("No", dim(Default)[1]/2)  
glm.probs = predict(glm.fit, Default[-train, ], type = "response")  
glm.pred[glm.probs > 0.5] = "Yes"
```

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

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

```
## [1] 0.0254
```

2.54% test error rate from 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)
glm.fit = glm(default ~ income + balance, data = Default, family = binomial,
subset = train)
glm.pred = rep("No", dim(Default)[1]/2)
glm.probs = predict(glm.fit, Default[-train, ], type = "response")
glm.pred[glm.probs > 0.5] = "Yes"
mean(glm.pred != Default[-train, ]$default)

## [1] 0.0274

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

## [1] 0.0244

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

## [1] 0.0244
```

It seems to average around 2.7% test error rate.

- 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 dummyvariable for student leads to a reduction in the test error rate.

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

```
## [1] 0.0278
```

Test error rate of 2.64 percent using student dummy variable. Using the validation set method, it doesn't seem like including the student dummy variable causes the test error rate to go down.

***Q6. 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.

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

##
## 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. 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.

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

```
library(boot)
boot(Default, boot.fn, 50)

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 50)
##
##
## Bootstrap Statistics :
##          original      bias    std. error
## t1* -1.154047e+01 -5.661486e-02 4.847786e-01
## t2*  2.080898e-05 -7.436578e-08 4.456965e-06
## t3*  5.647103e-03  1.854126e-05 2.639029e-04
```

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

Similar answers to the second and third significant digits.

**Q9. We will now consider the “Boston” housing data set, from the “MASS” library.**

```
library(MASS)
summary(Boston)

##      crim             zn            indus            chas
##  Min. : 0.00632  Min. : 0.00  Min. : 0.46  Min. :0.00000
##  1st Qu.: 0.08205 1st Qu.: 0.00  1st Qu.: 5.19  1st Qu.:0.00000
##  Median : 0.25651 Median : 0.00  Median : 9.69  Median :0.00000
##  Mean   : 3.61352 Mean   : 11.36  Mean   :11.14  Mean   :0.06917
##  3rd Qu.: 3.67708 3rd Qu.: 12.50  3rd Qu.:18.10  3rd Qu.:0.00000
##  Max.   :88.97620 Max.   :100.00  Max.   :27.74  Max.   :1.00000
##      nox              rm            age            dis
##  Min. :0.3850  Min. :3.561  Min. : 2.90  Min. : 1.130
##  1st Qu.:0.4490 1st Qu.:5.886  1st Qu.: 45.02  1st Qu.: 2.100
##  Median :0.5380 Median :6.208  Median : 77.50  Median : 3.207
##  Mean   :0.5547 Mean   :6.285  Mean   : 68.57  Mean   : 3.795
##  3rd Qu.:0.6240 3rd Qu.:6.623  3rd Qu.: 94.08  3rd Qu.: 5.188
##  Max.   :0.8710 Max.   :8.780  Max.   :100.00  Max.   :12.127
##      rad              tax            ptratio          black
##  Min. : 1.000  Min. :187.0  Min. :12.60  Min. : 0.32
##  1st Qu.: 4.000 1st Qu.:279.0  1st Qu.:17.40  1st Qu.:375.38
##  Median : 5.000 Median :330.0  Median :19.05  Median :391.44
##  Mean   : 9.549 Mean   :408.2  Mean   :18.46  Mean   :356.67
```

```

##   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##   Max.    :24.000   Max.    :711.0   Max.    :22.00   Max.    :396.90
##   lstat          medv
##   Min.    : 1.73   Min.    : 5.00
##   1st Qu.: 6.95   1st Qu.:17.02
##   Median  :11.36   Median  :21.20
##   Mean    :12.65   Mean    :22.53
##   3rd Qu.:16.95   3rd Qu.:25.00
##   Max.    :37.97   Max.    :50.00

set.seed(1)
attach(Boston)

```

- a. Based on this data set, provide an estimate for the population mean of "medv". Call this estimate  $\mu^{\wedge}$ .

```

medv.mean = mean(medv)
medv.mean

```

```
## [1] 22.53281
```

- b. Provide an estimate of the standard error of  $\mu^{\wedge}$ . Interpret this result.

```

medv.err = sd(medv)/sqrt(length(medv))
medv.err

```

```
## [1] 0.4088611
```

- c. Now estimate the standard error of  $\mu^{\wedge}$  using the bootstrap. How does this compare to your answer from (b) ?

```

boot.fn = function(data, index) return(mean(data[index]))
library(boot)
bstrap = boot(medv, boot.fn, 1000)
bstrap

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original     bias    std. error
## t1* 22.53281 0.007650791  0.4106622

```

Similar to answer from (b) up to two significant digits. (0.4106 vs 0.4089)

- d. Based on your bootstrap estimate from (c), provide a 95% confidence interval for the mean of "medv". Compare it to the results obtained using t.test(Boston\$medv).

```
t.test(medv)
```

```

##  

##  One Sample t-test  

##  

## data: medv  

## t = 55.111, df = 505, p-value < 2.2e-16  

## alternative hypothesis: true mean is not equal to 0  

## 95 percent confidence interval:  

## 21.72953 23.33608  

## sample estimates:  

## mean of x  

## 22.53281  

c(bstrap$t0 - 2 * 0.4106, bstrap$t0 + 2 * 0.4106)  

## [1] 21.71161 23.35401

```

- e. Based on this data set, provide an estimate,  $\hat{\mu}_{med}$ , for the median value of “medv” in the population.

```

medv.med = median(medv)  

medv.med  

## [1] 21.2

```

- f. We now would like to estimate the standard error of  $\hat{\mu}_{med}$ . Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```

boot.fn = function(data, index) return(median(data[index]))  

boot(medv, boot.fn, 1000)  

##  

## ORDINARY NONPARAMETRIC BOOTSTRAP  

##  

##  

## Call:  

## boot(data = medv, statistic = boot.fn, R = 1000)  

##  

##  

## Bootstrap Statistics :  

##     original    bias    std. error  

## t1*      21.2 -0.0386   0.3770241

```

Median of 21.2 with SE of 0.377. Small standard error relative to median value.

- g. Based on this data set, provide an estimate for the tenth percentile of “medv” in Boston suburbs. Call this quantity  $\hat{\mu}^{0.1}$ .

```

medv.tenth = quantile(medv, c(0.1))  

medv.tenth  

## 10%  

## 12.75

```

- h. Use the bootstrap to estimate the standard error of  $\mu^{0.1}$ . Comment on your findings.

```
boot.fn = function(data, index) return(quantile(data[index], c(0.1)))
boot(medv, boot.fn, 1000)

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original   bias   std. error
## t1*    12.75  0.0186  0.4925766
```

Tenth-percentile of 12.75 with SE of 0.4925. Small standard error relative to tenth-percentile value.