# Homework #6

Justin Robinette

February 26, 2019

*No collaborators for any problem*

**Question 5.4.3, pg 198:** We now review k-fold cross-validation.

**Part A:** Explain how k-fold cross-validation is implemented.

**Results:** From *page 181* in the text, the k-fold CV approach "involves randomly dividing the set of observations into *k* groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining *k-1* folds. The mean squared error, MSE, is then computed on the observations in the held-out fold. This procedure is repeated *k* times."

The test error is then estimated by averaging the *k* resulting MSE values.

**Part B:** What are the advantages and disadvantages of k-fold cross-validation relative to: - The validation set approach? - LOOCV?

**Results:** The main disadvantages of the validation set approach, relative to k-fold cross-validation, are that the test error rate can be highly variable depending on which observations are included in the training/validation data sets. Also, since only some of the observations are included in the model - the validation set error rate can overestimate the test error for the model fit over the entire dataset.

The valdiation set approach is easier to implement, however.

Regarding LOOCV, in relation to the k-fold CV, LOOCV can be computationally expensive because it has *k* = *n*, meaning it iterates through the entire dataset - as opposed to setting a *k* that is less than the number of observations in the k-fold CV. Also, the LOOCV approach has a higher variance than k-fold CV since we are averaging the results of *n* fitted models - meaning they are highly correlated outputs.

Conversely, the LOOCV method removes the randomness in the splitting of training and test data sets and provides a more unbiased estimate of the test error.

**Question 5.4.5, pg 198:** In chapter 4, we used logistic regression to predict the probability of *default* using *income* and *balance* on the *Default* dataset. 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. *Use set.seed(702) to make results replicable*

**Part A:** Fit a logistic regression model that uses *income* and *balance* to predict *default*.

**Results:** Below I've loaded the Default dataset and fit a logistic regression model using the predictors from the instructions. I've printed the summary for confirmation.

```
## 
## 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
```

**Part 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 and validation set 2) Fit a multiple logistic regression model using only the training observations 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 that individual to the *default* category if the posterior probability is greater than 0.5 4) Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified

**Results:** Here, I followed the instructions and split the data set with 70% of the obs going to the training set and 30% of the obs going to the test set.

Then I fit the glm using the training data set. I then included a confusion matrix, although the problem didn't request one, showing that the model was much more accurate (*99.6%*) at predicting the 'No' default values than it was at predicting the 'Yes' default values (*37.4%*). I believe this is because there are so few 'Yes' default values that it makes it hard for the model to identify what causes them.

Finally, I reported the test error rate as a percentage (*2.5%*), to fulfill the requirements of Part 4 of this exercise.

```
## 
## Call:
## glm(formula = default ~ income + balance, family = binomial, 
##     data = Default.train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.4879  -0.1421  -0.0560  -0.0203   3.7364  
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.157e+01  5.194e-01 -22.284  < 2e-16 ***
## income       1.865e-05  5.889e-06   3.166  0.00154 ** 
## balance      5.723e-03  2.760e-04  20.732  < 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: 2050.5  on 6999  degrees of freedom
## Residual deviance: 1103.5  on 6997  degrees of freedom
## AIC: 1109.5
## 
## Number of Fisher Scoring iterations: 8

##         Predicted
## Observed   No  Yes
##      No  2888   13
##      Yes   62   37
```

***Test Error Rate at 70% Train / 30% Test Split***

| Test Error (%) |
|---|
| 2.5 |

**Part C:** Repeat the process in (b) three times using three different splits of the observations into training and validation sets. Comment on the results obtained.

**Results:** In the previous exercise, I had used a 70/30 train/test split. Here, per the instructions, I used three different split ratios: *75/25*, *60/40*, and *55/45*. I report all 4 error rates as percentages below. As we can see, the original *70/30* split has the best error rate at *2.5%*. This is followed closely by the other three splits with the *55/45* split performing the worst with an error rate of *2.8%*.

***Test Error Rates by Split (%)***

| Test Error 55/45 Split | Test Error 60/40 Split | Test Error 70/30 Split | Test Error 75/25 Split |
|---|---|---|---|
| 2.844444 | 2.575 | 2.5 | 2.56 |

**Part 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 including the dummy variable for *student* lead to a reduction in the test error rate.

**Results:** Because the 70/30 split performed the best in the above comparison, that is the split I went with for this exercise. This time, I fit a model using student as a predictor. As we can see from the summary, *student* is considered statistically significant as predictor at an alpha of 0.05. With that being said, we can see from the error rate comparison table that there is no improvement in the model's error rate with the inclusion of the student variable. The error rate remains at *2.5%*.

```
## 
## Call:
## glm(formula = default ~ income + balance + student, family = binomial, 
##     data = Default.train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.4863  -0.1405  -0.0544  -0.0196   3.7601  
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.094e+01  5.873e-01 -18.626   <2e-16 ***
## income       1.691e-06  9.754e-06   0.173   0.8624    
## balance      5.812e-03  2.817e-04  20.633   <2e-16 ***
## studentYes  -6.164e-01  2.818e-01  -2.187   0.0287 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 2050.5  on 6999  degrees of freedom
## Residual deviance: 1098.8  on 6996  degrees of freedom
## AIC: 1106.8
## 
## Number of Fisher Scoring iterations: 8
```

***Test Error Rate at 70% Train / 30% Test Split***

| Test Error w/out Student (%) | Test Error w/ Student (%) |
|---|---|
| 2.5 | 2.5 |

**Question 5.4.7, pg 200:** In sections 5.3.2 and 5.3.3, we saw that the **cv.glm()** function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quatities just using the **glm()** and **predict.glm()** functions, and a for loop. You will now take this approach in order to compute LOOCV error for a simple logistic regression model on the *Weekly* dataset. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

**Part A:** Fit a logistic regression model that predicts *Direction* using *Lag1* and *Lag2*.

**Results:** Here, I loaded the data and fit a logistic regression model using the predictors as required by the assignment.

```
## 
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly)
## 
## Deviance Residuals: 
##    Min      1Q  Median      3Q     Max  
## -1.623  -1.261   1.001   1.083   1.506  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672    
## Lag2         0.06025    0.02655   2.270 0.023232 *  
## ---
## 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: 1488.2  on 1086  degrees of freedom
## AIC: 1494.2
## 
## Number of Fisher Scoring iterations: 4
```

**Part B:** Fit a logistic regression model that predicts *Direction* using *Lag1* and *Lag2 using all but the first observation*.

**Results:** I fit the glm excluding the 1st observation using the same predictors as requested.

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly[-1,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6258  -1.2617   0.9999   1.0819   1.5071
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22324    0.06150   3.630 0.000283 ***
## Lag1        -0.03843    0.02622  -1.466 0.142683
## Lag2         0.06085    0.02656   2.291 0.021971 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1486.5  on 1085  degrees of freedom
## AIC: 1492.5
##
## Number of Fisher Scoring iterations: 4
```

**Part C:** Use the model from (b) to predict the *Direction* of the first observation. Was this observation correctly classified?

**Results:** As we can see from below, the first observation was incorrectly classified by the model.

```
## [1] "The model's prediction for the first observation is: Up"

## [1] "The true direction of the first observation is: Down"
```

**Part D:** Write a for loop from i=1 to i=n, where n is the number of observations in the dataset, that performs each of the following steps: 1) Fit a logistic regression using all but the ith observation to predict *Direction* using *Lag1* and *Lag2* 2) Compute the posterior probability of the market moving up for the ith observation 3) Use the posterior probability for the ith observation in order to predict whether or not the market moves up 4) Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0

**Results:** Here I wrote a loop that takes into account the number of observations, and iterates through fitting a logistic regression on all but the *ith* observation to predict market Direction. If an error was made, a 1 in indicated. If the correct prediction was made, a 0 is indicated.

```
##    [1] 1 1 0 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 0
##   [35] 1 0 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0
##   [69] 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1
##  [103] 1 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0
##  [137] 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 1
##  [171] 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0
##  [205] 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0 1
##  [239] 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0
##  [273] 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1
##  [307] 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0
##  [341] 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 1
##  [375] 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 0
##  [409] 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1
##  [443] 1 1 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1
##  [477] 0 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0
##  [511] 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0
##  [545] 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1
##  [579] 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0
##  [613] 0 0 1 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 0 0 1 1 1 1 0 1
##  [647] 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1
##  [681] 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 1 0 1 0 1
##  [715] 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 1
##  [749] 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 0 1 1 1 0 0 0
##  [783] 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1
##  [817] 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0
##  [851] 1 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1
##  [885] 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0 0
##  [919] 0 0 0 1 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 0
##  [953] 1 0 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0
##  [987] 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0 1 0 0 1
## [1021] 0 0 1 1 0 1 1 1 0 1 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1
## [1055] 1 0 0 0 0 0 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0
## [1089] 0
```

**Part E:** Take the average of the n numbers obtained in *Part D, 4* in order to obtain the LOOCV estimate for the test error. Comment on the results.

**Results:** Here, I've taken the mean of the errors to get the error estimate of the LOOCV model.

***LOOCV Test Error Estimate***

| Test Error Estimate |
|---|
| 0.4499541 |

**Exercise 4:** Write your own code (similar to Q #3 above) to estimate test error using 6-fold cross validation for fitting linear regression with $mpg \sim horsepower + horsepower^2$ from the Auto data in the ISLR library. You should show the code in your final PDF.

**Results:**

```r
# attach data set
data("Auto", package = "ISLR")

set.seed (702)

# randomly shuffle the data set
Auto.dat <- Auto[sample(nrow(Auto)),]

# Create 6 equal size folds
folds <- cut(seq(1, nrow(Auto)), breaks = 6, labels = FALSE)

# Perform 6 fold CV
mse <- c()
for (i in 1:6){
  test_indeces <- which(folds == i, arr.ind = TRUE)
  test_set <- Auto[test_indeces, ]
  train_set <- Auto[-test_indeces, ]
  lm <- lm(mpg ~ horsepower + I(horsepower^2), data = train_set)
  prediction <- predict(lm, test_set)
  mse[i] <- mean((test_set$mpg - prediction)^2)
}

# mean(error for 6 fold cv)
paste("MSE for 6-fold cross validation: ",mean(mse))

## [1] "MSE for 6-fold cross validation:  22.5607957551565"
```

**Exercise 5:** Last homework you started analyzing the dataset you chose. Now continue the analysis and perform Logistic Regression, KNN, LDA, QDA, MclustDA, and MclustDA with EDDA.

**Results:** First, I loaded the dataset and repeating the manipulation and imputation steps that I had done in the previous homework. I printed the summary and header to ensure the steps were completed properly.

```
##  A1             A2              A3          A4       A5            A6     
##  ?:  0   Min.   :  2.0   Min.   : 0.000   ?:  0   ? :  0   c      :146  
##  a:210   1st Qu.: 73.0   1st Qu.: 1.000   l:  2   g :525   q      : 78  
##  b:480   Median :135.5   Median : 2.750   u:525   gg:  2   w      : 64  
##          Mean   :149.0   Mean   : 4.759   y:163   p :163   i      : 59  
##          3rd Qu.:219.8   3rd Qu.: 7.207                    aa     : 54  
##          Max.   :350.0   Max.   :28.000                    ff     : 53  
##                                                            (Other):236  
##        A7             A8          A9      A10         A11        A12    
##  v      :408   Min.   : 0.000   f:329   f:395   Min.   : 0.0   f:374  
##  h      :138   1st Qu.: 0.165   t:361   t:295   1st Qu.: 0.0   t:316  
##  bb     : 59   Median : 1.000                   Median : 0.0          
##  ff     : 57   Mean   : 2.223                   Mean   : 2.4          
##  j      :  8   3rd Qu.: 2.625                   3rd Qu.: 3.0          
##  z      :  8   Max.   :28.500                   Max.   :67.0          
##  (Other): 12                                                          
##  A13          A14              A15            A16    
##  g:625   Min.   :  2.00   Min.   :     0.0   N:383  
##  p:  8   1st Qu.: 25.00   1st Qu.:     0.0   P:307  
##  s: 57   Median : 54.00   Median :     5.0          
##          Mean   : 59.27   Mean   :  1017.4          
##          3rd Qu.: 95.00   3rd Qu.:   395.5          
##          Max.   :171.00   Max.   :100000.0          
##

##   A1  A2    A3 A4 A5 A6 A7   A8 A9 A10 A11 A12 A13 A14 A15 A16
## 1  b 158 0.000  u  g  w  v 1.25  t   t   1   f   g  70   0   P
## 2  a 330 4.460  u  g  q  h 3.04  t   t   6   f   g  13 560   P
## 3  a  91 0.500  u  g  q  h 1.50  t   f   0   f   g  98 824   P
## 4  b 127 1.540  u  g  w  v 3.75  t   t   5   t   g  33   3   P
## 5  b  45 5.625  u  g  w  v 1.71  t   f   0   f   s  39   0   P
## 6  b 170 4.000  u  g  m  v 2.50  t   f   0   t   g 117   0   P
```

Then, I used correlation matrix to determine variables that are highly correlated with my response variable, A16. I arbitrarily set my limit at abs(0.3), which as we can see, includes variables A8, A9, A10, and A11.

```
##        A1   A2   A3    A4    A5   A6 A7   A8   A9  A10  A11  A12  A13  A14
## A16 -0.03 0.16 0.21 -0.19 -0.19 0.13  0 0.31 0.72 0.46 0.41 0.03 -0.1 -0.1
##      A15 A16
## A16 0.18   1
```

First, I split the data set into training and test sets. Then I fit a glm, LDA, QDA, KNN, MclustDA and MclustDA with EDDA as instructed by the exercise.

Next, I used the respective models to predict the “Positive” or “Negative” value for A16.

Then, I created confusion matrices for each model based on these predictions. I used KNN k= 1, 5, and 10. As we can see, k=5 is most accurate.

Beyond this insight, I will examine the accuracy rates further in the next section.

```
## [1] "GLM Confusion Matrix"

##         Predicted
## Observed  N  P
##        N 98 26
##        P  8 76

## [1] "LDA Confusion Matrix:"

##         Predicted
## Observed  N  P
##        N 98 26
##        P  8 76

## [1] "QDA Confusion Matrix:"

##         Predicted
## Observed   N   P
##        N 112  12
##        P  22  62

## [1] "MClustDA Confusion Matrix:"

##      Predicted
## Class   N   P
##     N 109  15
##     P  16  68

## [1] "MClustDA Confusion Matrix w/ EDDA:"

##      Predicted
## Class   N   P
##     N 112  12
##     P  22  62

## [1] "KNN Confusion Matrix k=1:"

##         Predicted
## Observed  N  P
##        N 97 27
##        P 13 71

## [1] "KNN Confusion Matrix k=5:"

##         Predicted
## Observed   N   P
##        N 101  23
##        P  15  69

## [1] "KNN Confusion Matrix k=10:"

##         Predicted
## Observed  N  P
##        N 96 28
##        P 13 71
```

Here I calculated the true positive rate, true negative rate, and overall error rate for each model against the test set. Remember, the test set was a randomly selected group of observations equal to 30% of the overall *credit.screening* data set.

Below, I've printed a table that compares the true positive rate, the true negative rate, and the error rate (against the test set) for each model

As we can see, the MclustDA model performed, overall, the best with an error rate of 14.9%. The GLM and LDA models were the best in terms of true positive accuracy at 90.48% accuracy. The QDA and MclustDA w/ EDDA models were the best in terms of true negative accuracy with measures of 90.32% accurate.

Based on the below results, and if overall accuracy is our most important measure, then the MclustDA model is superior.

```
##               True Positive Accuracy (%) True Negative Accuracy (%)
## GLM                                90.48                      79.03
## LDA                                90.48                      79.03
## QDA                                73.81                      90.32
## MclustDA                           80.95                      87.90
## MclustDA.EDDA                      73.81                      90.32
## KNN5                               82.14                      81.45
##               Error Rate (%)
## GLM                    16.35
## LDA                    16.35
## QDA                    16.35
## MclustDA               14.90
## MclustDA.EDDA          16.35
## KNN5                   18.27
```