Homework 6

Andrew Boschee

3/12/2020

No collaborators. Outside Resources: Rdocumentation.com, The Elements of Statistical Learning

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

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

Results: k-fold involves spliting the observations up into equal sized groups with one of the given sets as the validation set. The remaining sets/folds are used to train the model. This is repeated altering which set is held-out and finding the mean squared error across the the folds.

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

Results: The most noticable difference that stands out is the compuatino power needed with cross-validation. k-fold will be in the middle of LOOCV and validation set approach. LOOCV will be much slower with k equal to the number of observations overall and k-fold is modified to whatever you want k to be. Validation set approach is clearly the most simple approach while LOOCV can also be the most unbiased option taking away the random splitting that occurs from validation set approach.

K-fold is a nice balanced approach in my opinion, but as computing power continues to become more impressive, LOOCV is definitely worth trying.

Default Predictions

Logistic Regression

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: After setting a random seed '702', the Default dataset is imported and model is fit with default as the dependent variable and income and balance as independent variables. Summary is shown with both predictors being statistically significant.

```
##
## 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
```

MLR Validation Set

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: Using 75/25 split, the dataset is indexed and the model is run again identical to part a. Both variables are still seen as statistically significant.

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1551  -0.1339  -0.0515  -0.0186   3.7785
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.178e+01  5.274e-01 -22.326  < 2e-16 ***
## income       1.845e-05  5.962e-06   3.095  0.00197 **
## balance      5.799e-03  2.774e-04  20.904  < 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: 2069.6  on 7499  degrees of freedom
## Residual deviance: 1103.8  on 7497  degrees of freedom
## AIC: 1109.8
##
## Number of Fisher Scoring iterations: 8
```

Table 1: Multiple Logistic Regression - Default

MisclassificationRate	Sensitivity	Specificity
0.0348	0.2871287	0.9937474

Results Continued: Using the classification function created in prior assignment, we can see the misclassification rate, sensitivity, and specificity. With a threshold of .5, the misclassification rate is very low at about .03.

Split Size Comparison

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: Started by declaring new training sets with multiplication of nrow function of dataset and .6, .8, and .9. Created index with sample function on each set, and then split each set based on the index variables

Table 2: 60/40 Split

MisclassificationRate	Sensitivity	Specificity
0.02625	0.3161765	0.9968944

Table 3: 80/20 Split

MisclassificationRate	Sensitivity	Specificity
0.025	0.3548387	0.99484

Table 4: 90/10 Split

MisclassificationRate	Sensitivity	Specificity
0.02	0.3793103	0.9979403

Results Continued: The misclassification rate went down a noticeable amount at the 90/10 split, but there was not much difference at the 60/40 or 80/20. Can see that all splits have very high specificity giving the conclusion there were not many false positives in these classifications

Additional Predictor Variable

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: Using the GLM function with binomial family argument and train set in data argument, predictions were wade with threshold of .5.

Table 5: Logistic Regression w/ Student Variable

MisclassificationRate	Sensitivity	Specificity
0.0996	0.8019802	0.9045436

Results Continued: Adding the dummy variable had a very noticable impact on misclassification, sensitivity and specificity. With prior models returning around .02 error rate, this model ended up with approximately .10 misclassification rate. This result came using the 75/25 split from part a.

Weekly Dataset

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).

Logistic Regression

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

```
##
## 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
```

Results: Model created after loading dataset with Direction as dependent variable and Lag1 and Lag2 as independent variables

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

Results: Fit model with all but the first observation from the dataset. Lag2 is still the only statistically significant independent variable. Slight change in AIC.

```
##
## 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
```

Direction Prediction

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

Results: The model predicted the market to go up while the actual direction was down. Since day one of working with this dataset and my non-optimistic outlook of predicting the market, I am not surprised with the result. Used cbind.data.frame to put prediction and actual output side-by-side.

Table 6: Predicted Output vs Actual Output

Predicted	Actual
up	Down

LOOCV

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: Set variable n as the number of rows from the dataset and looped from 1 to n. Model is built one

row with data parameter in the loop using the ith observation. Prediction is made classifying each iteration and marking 0/1 based on classification.

```
# declare variable to total number of rows
n <- nrow(Weekly)
# framework for final output
weeklyOutput <- rep(0, nrow(Weekly))
for (i in 1:n){
  model <-glm(Direction ~ Lag1 + Lag2, data = Weekly[-i,], family = binomial)
  modelPred <- ifelse(predict(model, Weekly[i, ], type = 'response') > 0.5, 'Up','Down')
  weeklyOutput[i] <- ifelse(Weekly[i,]$Direction == modelPred, 0, 1)

}
```

LOOCV Results

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

Results: Test error was rougly 45% when applying the mean function on the output of the loop.

Table 7: LOOCV Error Rate - Weekly

0.4499541

Auto Cross-Validation

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: In order to try multiple folds, the two options that came to mind were to create a function or a loop that would try multiple number of folds. The easiest way in my opinion is just making a function that can be run multiple times with the number of folds as the argument. The loop goes from 1 to the number of folds given to the function. The variable 'fold' sets up the breaks based on the number of rows n the dataset (could adjust the function to accept dataset for future use). The function splits the dataset up between train/test sets, trains the model, makes a prediction, and iterates for the number of folds calculating the error and returns the average/MSE of the error.

```
# set seed
set.seed(702)
data("Auto", package = 'ISLR')

# function iterating number of folds from parameters
CVnFoldLinMod <- function(numFold){

fold <- cut(seq(1, nrow(Auto)), breaks = numFold, labels = FALSE)
meanSqEr <- c()
```

```
for (i in 1:numFold){
  testIndex <- which(fold == i, arr.ind = TRUE)
  train <- Auto[testIndex,]
  test <- Auto[-testIndex,]
  model <- lm(mpg ~ horsepower + I(horsepower^2), data = train)
  prediction <- predict(model, test)
  meanSqEr[i] <- mean((test$mpg - prediction)^2)

}
return(mean(meanSqEr))
}


sixFoldAuto <- CVnFoldLinMod(6)
```

Table 8: Six Fold CV MSE

76.36089

Wine Type Predictions - Model Comparison

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: After loading the csv files, the wine types are defined as 1/0 for binary output. Then a train/test split is done at a 75/25 ratio.

Each method does a good job predicting the type of wine when all predictors are included in the model. With no tuning of parameters, all modeling methods are above 90% with several methods at nearly 100% accuracy. Looking back, I can see why this dataset is mainly intended for regression analysis with the rating as the dependent variable.

Table 9: Accuracy Comparison

GLM	LDA	QDA
0.9963077	0.9969231	0.9901538

Table 10: Mclust Comparison - Wine Classification

MclustDA	MclustEDDA
0.9956923	0.9901538

Table 11: KNN Comparison

k=1	k=3	k=5	k=10
0.9446154	0.9390769	0.9390769	0.9464615