Homework 7

Rylie Fleckenstein

Exercises (ISLR)

1. Question 5.4.3 on page 198

We now review k-fold cross-validation.

(a) Explain how k-fold cross-validation is implemented.

K-fold cross validation is implemented by creating k subsets of the data and then iterating through each subset to be used as the test data while the remaining (k - 1) subsets of the data are used as the training data. Then, at the end, the mean of all k calculated errors is taken for the final k-fold error.

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

i. The validation set approach?

When the data set is relatively small with few observations it can be beneficial to implement k-fold cross validation because you are able to utilize all of the data set in the model. In the validation set approach you are not afforded the luxury of using all the data to train your model because a percentage of it must be set aside as the test data so that there is no data leakage to the model which would skew or bias your predictions. The down side of k-fold cross validation in comparison to the validation set approach is the computational efficiency is much higher for the validation set approach in comparison to the k-fold approach. Since you are fitting k models in the k-fold approach it will take more computing power and more time to implement this strategy.

ii. LOOCV?

The advantage of the k-fold cross validation approach in comparison to the LOOCV approach is that it is more efficient. The LOOCV approach is implemented by iterating through every single data point in the data set and using each individual point as the test data and the remaining data as the training data and fitting a model for every data point in the data set. The LOOCV approach is extremely computationally heavy with a time complexity of O(n^2). The time complexity of the k-fold cross validation is O(n) but approaches the time complexity of O(n^2) as k approaches LOOCV. The disadvantage of k-fold in comparison to LOOCV is again that LOOCV uses all of the data in its fullest capacity. K-fold mirrors that of LOOCV, in the sense that we want to get the most out of the data we have, but is drawn back to save on computational expenditure.

2. Question 5.4.5 on page 198

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.

```r
# multiple logistic regression model
d.fit <- glm(default ~ income + balance, data = train.d, family = binomial())
```

(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.

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

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.

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

```
## Training Validation 70/30 split
```

```
## $`Confusion Matrix`
##        pred
## resData    0    1
##       0 2884   10
##       1   70   36
##
## $`Misclassification Rate`
## [1] 2.666667
```

(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.

Above I implemented a multiple logistic regression model using the validation set approach with a split of 70% training data and 30% test data and was able to obtain a misclassification rate of 2.67%. Below I implemented three different splits and the results are the following: 80/20 misclassification rate 2.8%, 60/40 misclassification rate 2.57%, and 50/50 misclassification rate 2.5%. So, best model was created using a train/test split of 50/50 and the misclassification rate decreased linearly as the test set and training set approached a 1 to 1 ratio.

```
## Training Validation 80/20 split
```

```
## $`Confusion Matrix`
##        pred
## resData    0    1
##       0 1920    6
##       1   50   24
##
## $`Misclassification Rate`
## [1] 2.8
```

```
## Training Validation 60/40 split
## $`Confusion Matrix`
##        pred
## resData    0    1
##       0 3851   14
##       1   89   46
##
## $`Misclassification Rate`
## [1] 2.575
```

```
## Training Validation 50/50 split
```

```
## $`Confusion Matrix`
##        pred
## resData    0    1
##       0 4818    9
##       1  116   57
##
## $`Misclassification Rate`
## [1] 2.5
```

(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.

Below I implemented the validation set approach on a logistic regression model that included a dummy variable for student. I used the 50/50 training/test split since it was the optimal split on the previous model and was able to obtain a misclassification rate of 2.56% which was slightly higher than the previous modeling. Therefore, the addition of the student variable decreases the accuracy of our model and should be left out.

```
## $`Confusion Matrix`
##        pred
## resData    0    1
##       0 4818    9
##       1  119   54
##
## $`Misclassification Rate`
## [1] 2.56
```

3. Question 5.4.7 page 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 quantities using just the glm() and predict.glm() functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

(a) Fit a logistic regression model that predicts Direction using Lag1 and Lag2.

```r
# logistic regression model
glm.fit <- glm(Direction ~ Lag1 + Lag2, data=w.dat, family = binomial())
```

(b) Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.

```r
# log reg model
glm.fit.2 <- glm(Direction ~ Lag1 + Lag2, data=w.train, family = binomial())
```

(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if P(Direction="Up"|Lag1, Lag2) > 0.5. Was this observation correctly classified?

Below I compared the prediction made by our model to the actual classification and we can see the boolean value of False was returned, meaning our model incorrectly classified that observation.

```
## [1] FALSE
```

(d) Write a for loop from i = 1 to i = n, where n is the number of observations in the data set, that performs each of the following steps:

i. Fit a logistic regression model using all but the ith observation to predict Direction using Lag1 and Lag2.

ii. Compute the posterior probability of the market moving up for the ith observation.

iii. Use the posterior probability for the ith observation in order to predict whether or not the market moves up.

iv. 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.

(e) Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

Above I created a for loop to implement LOOCV on a model that used Lag1 and Lag2 to predict if the market with go up or down in the Weekly data set and the model had a test error of 44.99% as seen below. In comparison to the previous homework and the previous models trained on the Weekly data set, this model performed pretty well and proved to be competetive to what we had accomplished before.

```
## [1] 0.4499541
```

4. Write your own code (similar to Exercise #3 above) to estimate test error using k-fold cross validation for fitting a linear regression model of the form

$$mpg = \beta_0 + \beta_1 * X_1 + \beta_2 * X_1^2$$

from the **Auto** data in the **ISLR** library, with X_1 = horsepower. Use `echo = T` to show the code. Test this code with `k = 5` and `k = 30`. Discuss the computational trade-off between the two choices of `k`. Do not use the `cv.glm` function.

The time complexity of k-fold cross validation is O(n) but as k approaches the number of observations in the data set, the time complexity approaches that of LOOCV which is O(n^2). Therefore, in order to reap the benefits of k-fold cross validation versus LOOCV it is in our best interest to limit k to a relatively small number in the range of 5-10. In the example below we can see that setting k=5 gives us a model that performs just as well, if not slightly better, than a model trained using k=30 and it can be trained in a smaller amount of time. This would imply the computational trade-off between the two choices of k (5, 30) leans in favor of k=5. Also, it is important to note that as the data set becomes larger (millions of data points) the trade off becomes even bigger and lower k values really start to show their advantage.

```r
set.seed(16489)

# K-fold CV using a for loop
k.fold.func <- function(k, data){

  # empty vector for MSE values
  MSE <- c()

  # randomly shuffle data
  data <- data[sample(nrow(data)),]

  # creating folds
  folds <- rep_len(1:k, nrow(data))

  for (i in 1:k){

    fold <- which(folds == i)
    d.train <- data[-fold,]
    d.test <- data[fold,]

    lm.fit <- lm(mpg ~ horsepower + I(horsepower**2), data=d.train)
    MSE[i] <- mean((d.test$mpg - predict(lm.fit, d.test,
                                          type = "response"))**2)
  }

  k.fold.MSE <- mean(MSE)

  return(paste(k,"-fold MSE:", round(k.fold.MSE,3)))
}
```

```
## [1] "5 -fold MSE: 19.15"
```

```
## [1] "30 -fold MSE: 19.29"
```

Sources:

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2013). An introduction to statistical learning : with applications in R. New York :Springer