Homework #7

Justin Robinette

March 12, 2019

No collaborators for any problem

Question 5.4.1, pg 197: Using basic statistical properties of the variance, as well as single-variable calculus, derive (5.6). In other words, prove that αgiven by (5.6) does indeed minimize

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

Results: Working through the equation to get alpha:

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

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

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

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

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

$$\frac{\partial}{\partial\alpha}: 2\alpha\sigma_X^2 + 0 + 2\alpha\sigma_Y^2 - 2\sigma_Y^2 + 2\sigma_{XY} - 4\alpha\sigma_{XY} = 0$$

$$(2\sigma_X^2 + 2\sigma_Y^2 - 4\sigma_{XY})\alpha = 2\sigma_Y^2 - 2\sigma_{XY}$$

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

Checking that this is the minimum by proving second derivative is positive:

$$\frac{d^2}{d\alpha^2}Var(\alpha X + (1-\alpha)Y) = 2\sigma_x^2 + 2\sigma_y^2 - 4\sigma_{xy} = 2Var(X-Y) >= 0$$

Question 5.4.6, pg 199: 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** coefficients in two different ways:
(1) using the bootstrap
(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.

Part A: Using the *summary()* and *glm()* functions, determine the estimated standard errors for the coefficients associated with the **income** and **balance** in a multiple logistic regression model that uses both predictors.

Results: Below I've loaded the Default data set and printed the estimated standard errors for the coefficients associated with the predictors from the glm model.

Estimated Standard Error for Coefficients

	Standard Error
income	0.0000050
balance	0.0002274

Part 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 the **income** and **balance** in the multiple logistic regression model.

Results: Here is my function, *boot.fn()* that takes the data set and index of obs as inputs. The output are the coefficient estimates of the predictors.

```
boot.fn <- function(df, trainid) {
  return(coef(glm(default ~ income + balance, data=df, family=binomial, subset=trainid)))
}
boot.fn(Default, 1:nrow(Default))
```

```
##   (Intercept)        income       balance
## -1.154047e+01  2.080898e-05  5.647103e-03
```

Part 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**.

Results: The standard error estimates are pretty close between glm and the bootstrap when R=500. See below for the boostrap summary and glm coefficients.

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 500)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
## t1* -1.154047e+01 -3.219989e-02 4.572611e-01
## t2*  2.080898e-05 -2.132478e-07 4.827695e-06
## t3*  5.647103e-03  2.555284e-05 2.393010e-04

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

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

Results: The estimated standard error, as I said above, is very close. The difference is shown below. **income:** 4.985167e-06 with glm summary vs. 4.827695e-06 using bootstrap **balance:** 2.273731e-04 with glm summary vs. 2.393010e-04 using bootstrap

Question 5.4.9, pg 201: We will now consider the **Boston** housing data set, from the **MASS** library.

Part A: Based on this data set, provide an estimate for the population mean of *medv*. Call this estimate $\hat{\mu}$.

Results: The population mean for the median value of owner-occupied homes (in $1,000s) is approximately 22.53281.

```
## [1] "The population mean for medv is: 22.5328063241107 (in $1,000s)"
```

Part B: Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result. *Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.*

Results: The standard error for the mean of the median value of owner-occupied homes is approximately 0.40886.

```
## [1] "The standard error for medv is: 0.40886 (rounded to 5 decimal places)"
```

Part C: Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare with your answer from (b)?

Results: As we can see below, the mean values are the same rounded to 5 decimals. The standard error, using bootstrap, was 0.4253019 vs. 0.40886 in part B using the formula.

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = mean.fn, R = 500)
##
##
## Bootstrap Statistics :
##     original     bias    std. error
## t1* 22.53281 0.03373004   0.4253019

## [1] 22.53281
```

Part 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)*. *Hint: You can approximate a 95% confidence interval using the formula $[\hat{\mu} - 2SE(\hat{\mu}), \hat{\mu} + 2SE(\hat{\mu})]$.

Results: Using the formula for 95% confidence interval, we get a lower limit of 21.68220 vs. with the t.test function we get 21.72953. Using the formual, we get an upper limit of 23.38341 vs. using the t.test function we get 23.33608. As we can see, these confidence intervals for the mean of *medv* are pretty close but there is a slight difference in both.

```
## [1] 21.68220 23.38341

##
##  One Sample t-test
##
## data:  Boston$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
```

Part E: Based on the data set, provide an estimate, $\hat{\mu}_{med}$, for the median value of *medv* in the population.

Results: The median of the *medv* variable is 21.2 (in $1,000s)

```
## [1] "The median for medv is: 21.2 (in thousands)"
```

Part 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.

Results: Here we can see both the standard R function, and the user defined function using bootstrap give the same value of 21.2.

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = median.fn, R = 500)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0073   0.3646061

## [1] 21.2
```

Part G: Based on this data set, provide an estimate for the tenth percentile of *medv* in Boston suburbs. Call this quantity $\hat{\mu}_{0.1}$. (You can use the *quantile()* function.)

Results: The 10th percentile for the *medv* variable, using the quantile function, is 12.75 (in thousands).

	Estimated Tenth Percentile
10%	12.75

Part H: Use the bootstrap to estimate the standard error of $\hat{\mu}_{0.1}$. Comment on your findings.

Results: Here we see that we get the same tenth percent value using both methods. The standard error is 0.5064524.

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = tenth.fn, R = 500)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75   0.013   0.5064524

##   10%
## 12.75
```

Exercise 4: Last homework you have used different classification methods to analyze the dataset you chose. Now use i. Validation Set Approach(VSA) ii. LOOCV and 5-Fold Cross Validation to test the error rate for the following models. Chose the best model based on test error. iii. Logistic Regression iv. KNN (choose the best k) v. LDA vi. QDA vii. MclustDA - best model chosen by BIC viii. MclustDA - with modelType = "EDDA" ix. Find a new method that we haven't covered in class that can do classification

Summarize the results in a table form (See below). **DO NOT** show your summary directly from the code. Report only the important information as figures and tables. If you can't perform any of the analysis mentioned above, write the reason why. Write a description and draw conclusions in the context of the original problem from your analysis. Use the kable() function to make table when knitting.

Note: You may do presentations on the dataset you have been analyzing so be thinking of that while doing the analysis. Make sure to note all the steps you took in analyzing the data.

Results: First, I loaded the dataset and repeated the manipulation and imputation steps that I had done in the previous homework assignments.

```
##   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  1   1   1   f   g  70   0   P
## 2  a 330 4.460  u  g  q  h 3.04  1   1   6   f   g  13 560   P
## 3  a  91 0.500  u  g  q  h 1.50  1   0   0   f   g  98 824   P
## 4  b 127 1.540  u  g  w  v 3.75  1   1   5   t   g  33   3   P
## 5  b  45 5.625  u  g  w  v 1.71  1   0   0   f   s  39   0   P
## 6  b 170 4.000  u  g  m  v 2.50  1   0   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
```

I used the same code from HW6 to get error rates using VSA and requested models.

Here I calculated the error rates for the methods from the previous assignment.

First, I will perform VSA using a new method not covered in class thus far. I chose to use Neural Network.

To do so, I fit the model using the *neuralnet()* function and the same predictor variables as previous models. Then I obtained predictions based on the model and reported the error rate (**12.0192%**) on the test set.

```
## [1] "The test error rate of the Neural Network method is: 12.0192 %"
```

Next, I completed **LOOCV** for the 7 different methods, as requested. I used a couple different methods based on the modeling method being used and saved the error rates for reporting in the final table.

Lastly, I performed **5-fold CV** using a loop that iterates 5 times through each of the modeling methods to obtain the test error rate. I then saved these error rates to add to the final summary table.

No code or outputs required for the LOOCV or 5-Fold CV code chunks. All results reported below.

Below we see a table, similar to the one that was requested. It displays the test error rate for each Method for VSA, LOOCV, and 5-Fold CV.

We can see that the overall worst error rate came from the LOOCV approach with the MclustDA method. We also notice that KNN performed relatively worse than the other methods using k=5 (which was shown to be the best k in the previous homework assignments.)

The best performing methods using VSA were Logistic Reg, LDA, and Neural Network all coming in with a test error rate of *12.0192%.* The best performing methods using the LOOCV approach were Logistic Regression and LDA - each achieving a test error rate of *14.4928% (rounded to 4 decimals)*. This test error rate is also the best test error rate present in the 5-Fold CV column of the table. Here it was achieved by Logistic Regression, LDA, and Neural Network.

Test Error by Validation Approach (%)

Method	VSA	LOOCV	5-Fold CV
Logistic Reg	12.0192	14.4928	14.4928
KNN	16.3462	18.6957	18.5507
LDA	12.0192	14.4928	14.4928
QDA	16.8269	17.3913	17.5362
MclustDA	16.8269	20	17.5362
MclustDA (EDDA)	16.8269	17.3913	17.5362
Neural Network	12.0192	14.6377	14.4928