March 26, 2018

Assignment 3

Sanchit Singhal

INF 397 – Statistical Analysis and Learning w/ Prof. Varun Rai

Spring 2018

The University of Texas at Austin

1. **K-fold Validation**
2. **Implementation**

k-fold cross-validation is implemented by the dividing the full dataset of n observations into k non-overlapping groups – that usually have approximately the same size (n/k). k number of models are then built and trained using all except one of the groups which acts as a validation set for the model one at a time. The test error is estimated by averaging the k resulting MSE estimates.

1. **Comparison to:**
2. **Validation-set approach**

Although the validation-set approach is computationally less intense to execute (because you are only partitioning the data once into two sets and building/testing just 1 model), there are two major disadvantages when compared to the k-fold validation approach. The test error estimate in the validation-set approach is highly variable - it depends on which observations are chosen to be in the training set as opposed to the validation set. K-fold validation on the other hand leads to a less variable test error because it essentially uses every data point in the validation at some point as MSE is calculated for each k group and the test error is taken to be the average of all of them. Further, since the validation set approach only trains the model on a subset of the data, it will tend to perform worse than the k-fold validation approach which uses most of the data to train the model. This means that the validation-set approach can often over-estimate the test error rate on the entire dataset.

1. **LOOCV**

The LOOCV approach can be considered a special case of the k-fold validation approach in which k = n. This approach benefits from unbiased estimates of the test error (since each model is trained on all but 1 observation). However, this approach also has two main drawbacks when compared to the k-fold validation approach. In contrast to the validation-set approach, LOOCV is computationally more expensive than the k-fold because you fit the model n times as opposed to only k times in the k-fold validation approach. This issue is additionally amplified when the model gets more complex – costing even more to build each model. Secondly, even though LOOCV can give a less biased estimate of the test error it usually has higher variance than k-fold validation. This is simply because by taking the average of n fitted models (that are trained on almost identical sets of observations), the outputs are highly correlated and thus lead to higher variance.

Usually, we should consider the bias-variance tradeoff associated when selecting our value of k – it is suggested that using k=5 or 10 leads to test error rate estimates that suffer neither from too much bias nor variance.

library(class)  
  
**### Question 2**  
  
# Read CSV from working directory into R  
  
MyData <- read.csv(file="redwine.csv", header=TRUE, sep=",")  
  
# Create binary variable final\_quality using mean  
  
MyData$final\_quality <- with(ifelse(quality>mean(quality), 1, 0), data=MyData)  
  
# Creating dataset without original quality attribute  
  
myvars <- names(MyData) %in% c("quality")  
fulldataset <- MyData[!myvars]  
  
**## a) Logistic Regression with all the data**  
  
glm.fit <- glm(final\_quality ~ fixed.acidity+volatile.acidity+citric.acid+residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+density+pH+sulphates+alcohol, data=fulldataset, family=binomial)  
summary(glm.fit)

##   
## Call:  
## glm(formula = final\_quality ~ fixed.acidity + volatile.acidity +   
## citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +   
## total.sulfur.dioxide + density + pH + sulphates + alcohol,   
## family = binomial, data = fulldataset)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.4025 -0.8387 0.3105 0.8300 2.3142   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 42.949948 79.473979 0.540 0.58890   
## fixed.acidity 0.135980 0.098483 1.381 0.16736   
## volatile.acidity -3.281694 0.488214 -6.722 1.79e-11 \*\*\*  
## citric.acid -1.274347 0.562730 -2.265 0.02354 \*   
## residual.sugar 0.055326 0.053770 1.029 0.30351   
## chlorides -3.915713 1.569298 -2.495 0.01259 \*   
## free.sulfur.dioxide 0.022220 0.008236 2.698 0.00698 \*\*   
## total.sulfur.dioxide -0.016394 0.002882 -5.688 1.29e-08 \*\*\*  
## density -50.932385 81.148745 -0.628 0.53024   
## pH -0.380608 0.720203 -0.528 0.59717   
## sulphates 2.795107 0.452184 6.181 6.36e-10 \*\*\*  
## alcohol 0.866822 0.104190 8.320 < 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: 2209.0 on 1598 degrees of freedom  
## Residual deviance: 1655.6 on 1587 degrees of freedom  
## AIC: 1679.6  
##   
## Number of Fisher Scoring iterations: 4

**## b) Logistic Regression with the validation set approach  
  
## i) Splitting data into train and val using 80/20 split**  
  
set.seed(1)  
rows <- sample(x=nrow(fulldataset), size=.80\*nrow(fulldataset))  
trainset <- fulldataset[rows, ]  
valset <- fulldataset[-rows, ]  
  
**## ii) Logistic Regression fit with only the training set**  
  
glm2.fit <- glm(final\_quality ~ fixed.acidity+volatile.acidity+citric.acid+residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+density+pH+sulphates+alcohol, data=trainset, family=binomial)  
summary(glm2.fit)

##   
## Call:  
## glm(formula = final\_quality ~ fixed.acidity + volatile.acidity +   
## citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +   
## total.sulfur.dioxide + density + pH + sulphates + alcohol,   
## family = binomial, data = trainset)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.3285 -0.8640 0.3094 0.8477 2.2338   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 19.150545 85.590617 0.224 0.82296   
## fixed.acidity 0.111649 0.105894 1.054 0.29173   
## volatile.acidity -2.844256 0.534364 -5.323 1.02e-07 \*\*\*  
## citric.acid -0.988874 0.630212 -1.569 0.11662   
## residual.sugar 0.006617 0.061182 0.108 0.91388   
## chlorides -4.933251 1.738512 -2.838 0.00454 \*\*   
## free.sulfur.dioxide 0.016142 0.009139 1.766 0.07734 .   
## total.sulfur.dioxide -0.014461 0.003119 -4.636 3.55e-06 \*\*\*  
## density -25.095517 87.419908 -0.287 0.77406   
## pH -0.921976 0.793994 -1.161 0.24557   
## sulphates 2.567119 0.499801 5.136 2.80e-07 \*\*\*  
## alcohol 0.872373 0.113699 7.673 1.68e-14 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1770.7 on 1278 degrees of freedom  
## Residual deviance: 1352.0 on 1267 degrees of freedom  
## AIC: 1376  
##   
## Number of Fisher Scoring iterations: 4

**## iii) Predictions for validation set: high\_quality = 1, low\_quality = 0**  
  
glm2.probs <- predict(glm2.fit, valset, type="response")  
glm2.preds <- ifelse(glm2.probs>0.5, 1, 0)  
confusion\_matrix\_glm2 <- table(valset$final\_quality,glm2.preds)  
print(confusion\_matrix\_glm2)

## glm2.preds  
## 0 1  
## 0 105 27  
## 1 45 143

**## iv) Fraction of misclassified observation in validation set**  
  
Misclassified\_Predictions\_fraction = (confusion\_matrix\_glm2[1,2]+confusion\_matrix\_glm2[2,1])/sum(confusion\_matrix\_glm2)  
sprintf("Overall Fraction of Misclassified Predictions are: %f",Misclassified\_Predictions\_fraction)

## [1] "Overall Fraction of Misclassified Predictions are: 0.225000"

**### Question 3**  
  
**## a. Creating function**  
  
set.seed(1)  
boot.fn = function(data, index) return(coef(glm(final\_quality ~ fixed.acidity+volatile.acidity+citric.acid+residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+density+pH+sulphates+alcohol,data = fulldataset, family = binomial, subset = index)))  
  
**## b. Estimate standard errors**  
  
library(boot)

## Warning: package 'boot' was built under R version 3.4.3

boot(fulldataset, boot.fn, 5000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = fulldataset, statistic = boot.fn, R = 5000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 42.94994813 -4.1180669223 83.957974171  
## t2\* 0.13598034 -0.0016715528 0.105728098  
## t3\* -3.28169367 -0.0458145653 0.521044067  
## t4\* -1.27434734 -0.0439888532 0.599980503  
## t5\* 0.05532602 -0.0035509907 0.062123106  
## t6\* -3.91571291 -0.1327819151 1.777481968  
## t7\* 0.02222037 0.0003045197 0.008274123  
## t8\* -0.01639392 -0.0002640114 0.003078335  
## t9\* -50.93238519 4.1679273843 85.730917155  
## t10\* -0.38060751 -0.0345690167 0.737982583  
## t11\* 2.79510651 0.0763360253 0.522845664  
## t12\* 0.86682223 0.0101357328 0.107123809

**## c. Comparisons of the two sets of standard errors**  
  
# Both sets of the estimated standard errors obtained through glm() function and  
# bootstrap function are very similar. For example, the std. error for  
# free.sulfur.dioxide was 0.008236 with glm and with the bootsrap method, it was  
# 0.008264231 (t7\*). The std. error for residual.sugar was 0.053770 with glm,  
# and 0.062763447 with boostrap (t5\*).