Introduction to Statistical Learning Lab4

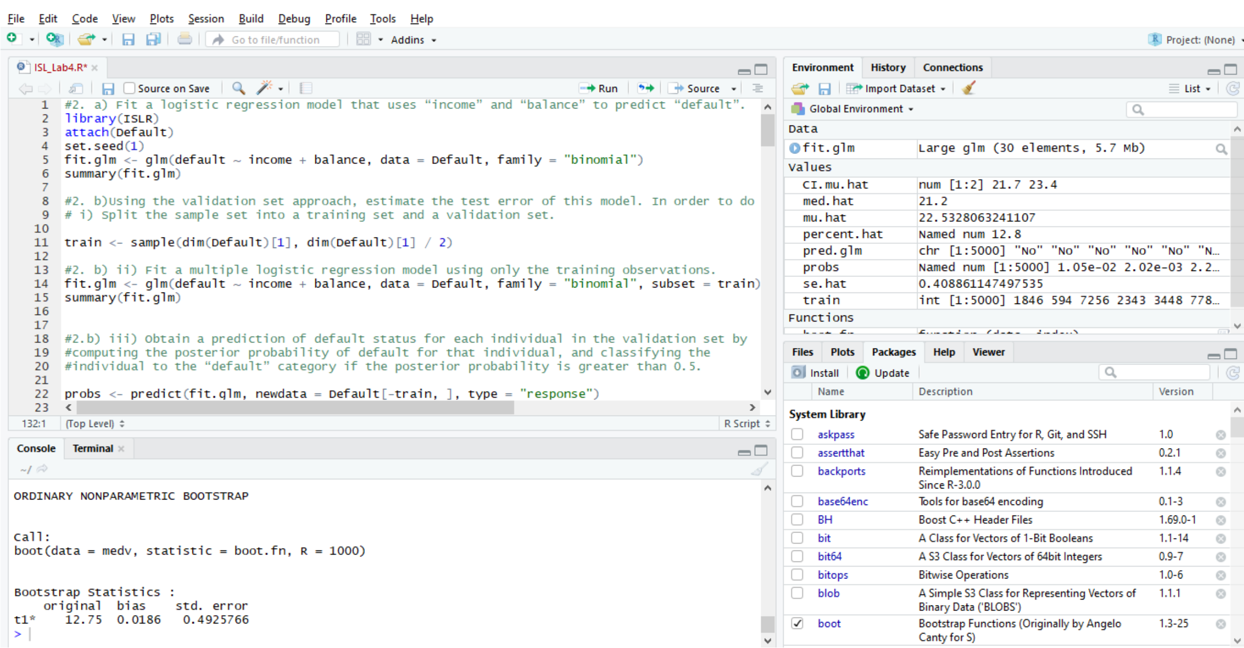
(Cross Validation and Bootstrap)

Name: Sandeep Reddy Salkuti

Id: 16296868

Email: sswf7@umsystem.edu

1) **You may download the R Code for Labs and the Data Sets to use from the textbook website.**



2) **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) (10 points) Fit a logistic regression model that uses income and balance to predict default.

```r
#2. a) Fit a logistic regression model that uses "income" and "balance" to predict "default"
library(ISLR)
attach(Default)
set.seed(1)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(fit.glm)
```

```
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

>
```

(b) (10 points total) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. (2.5 points) Split the sample set into a training set and a validation set.

```r
#2. b)Using the validation set approach, estimate the test error of this model. In order
# i) Split the sample set into a training set and a validation set.

train <- sample(dim(Default)[1], dim(Default)[1] / 2)
```

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
>
```

ii. (2.5 points) Fit a multiple logistic regression model using only the training observations.

```
Console   Terminal ×
~/

> #2. b) ii) Fit a multiple logistic regression model using only the training observations.
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> summary(fit.glm)

Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5830  -0.1428  -0.0573  -0.0213   3.3395

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.194e+01  6.178e-01 -19.333  < 2e-16 ***
income       3.262e-05  7.024e-06   4.644 3.41e-06 ***
balance      5.689e-03  3.158e-04  18.014  < 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: 1523.8  on 4999  degrees of freedom
Residual deviance:  803.3  on 4997  degrees of freedom
AIC: 809.3

Number of Fisher Scoring iterations: 8

>
```

iii. (2.5 points) 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.

```

#3.b) 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.

probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"

19:1   (Top Level) ÷                                                    R Script
```

```
Console   Terminal ×
~/

> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
>
```

```
Environment   History   Connections
Import Dataset
List
Global Environment
Data
fit.glm      Large glm (30 elements, 3.1 Mb)
Values
pred.glm     chr [1:5000] "No" "No" "No" "No" "No" "No" "No" "No" "No" "No" "...
probs        Named num [1:5000] 0.00175 0.00101 0.00816 0.00199 0.00115 ...
train        int [1:5000] 1017 8004 4775 9725 8462 4050 8789 1301 8522 1799 ...
```

iv. (2.5 points) Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
#2. b) iv) Compute the validation set error, which is the fraction of the observations in the validati
#set that are misclassified.
mean(pred.glm != Default[-train, ]$default)

27:1   (Top Level)   R Script
```

```
Console   Terminal
~/
> #2. b) iv) Compute the validation set error, which is the fraction of the observations in the validation
> #set that are misclassified.
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0254
>
```

From above with validation set approach we have 2.54% test error rate

(c) (10 points) 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.

```
#2. c)Repeat the process in (b) three times, using three different splits of the observations into a t
#and a validation set. Comment on the results obtained.

train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)

41:1   (Top Level)   R Script
```

```
Console   Terminal
~/
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0274
>
```

```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)
```

```
42:1  (Top Level) ÷
Console  Terminal ×
~/
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0244
>
```

```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)
```

```
47:1  (Top Level) ÷
Console  Terminal ×
~/
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0244
>
```

From above it is clear that validation test error rate is varying depending on which observations are included in training set and which observations are included in validation set.

(d) (10 points) 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.

```
#2. 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.

train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train)
pred.glm <- rep("No", length(probs))
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)
```

Console Terminal

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train)
> pred.glm <- rep("No", length(probs))
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0278
>
```

From above it doesn’t seems that by adding “student” dummy variable leads to reduction in validation set estimate of the test error rate.

3. (40 points) 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 logistic regression coefficients in two different ways: (1) using the bootstrap, and (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.

(a) (10 points) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.

```
> set.seed(1)
> attach(Default)
The following objects are masked from Default (pos = 3):

    balance, default, income, student

The following objects are masked from Default (pos = 4):

    balance, default, income, student

> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
> summary(fit.glm)

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
```

The glm() estimates of the standard errors for the coefficients β0, β1 and β2 are respectively 0.4347564, 4.9851672 x 10-6 and 2.2737314 x 10-4.

(b) (10 points)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 income and balance in the multiple logistic regression model.

```
#3. 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 “income” and “balance” in the multiple logistic regression model.

boot.fn <- function(data, index) {
  fit <- glm(default ~ income + balance, data = data, family = "binomial", subset = index)
  return (coef(fit))
}
```

90:1 (Top Level) R Script

Console Terminal

~/

```
> boot.fn <- function(data, index) {
+   fit <- glm(default ~ income + balance, data = data, family = "binomial", subset = index)
+   return (coef(fit))
+ }
>
```

(c) (10 points) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for income and balance.

```
#3. 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”.

library(boot)
boot(Default, boot.fn, 1000)
```

```
package ‘boot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\SandeepReddy\AppData\Local\Temp\Rtmp8CuYvp\downloaded_packages
> library(boot)
Warning message:
package ‘boot’ was built under R version 3.6.3
> boot(Default, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01 -3.945460e-02 4.344722e-01
t2*  2.080898e-05  1.680317e-07 4.866284e-06
t3*  5.647103e-03  1.855765e-05 2.298949e-04
```

The bootstrap estimates of the standard errors for the coefficients β0, β1 and β2 are respectively 0.4344, 4.8662 x 10(-6) and 2.2989 x 10(-4).

(d) (10 points) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

From the above observations we can assume that, the estimated standard errors obtained by the two methods are close.

4. (40 points) We will now consider the Boston housing data set, from the MASS library.

(a) (5 points) Based on this data set, provide an estimate for the population mean of medv. Call this estimate ˆμ.

```
#4. a) Based on this data set, provide an estimate for the population mean of “medv”. Call this estimate μ^.
library(MASS)
attach(Boston)
mu.hat <- mean(medv)
mu.hat
```

```
[1] 22.53281
>
```

(b) **Provide an estimate of the standard error of ˆµ. 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.**

```
#4. b)Provide an estimate of the standard error of µ^. Interpret this result.
se.hat <- sd(medv) / sqrt(dim(Boston)[1])
se.hat
```

```
108:1  (Top Level)

Console  Terminal
~/
[1] 0.4088611
>
```

(c) **(5 points) Now estimate the standard error of ˆµ using the bootstrap. How does this compare to your answer from (b)?**

```
#4. c)Now estimate the standard error of µ^ using the bootstrap. How does this compare to your answer from (b) ?
set.seed(1)
boot.fn <- function(data, index) {
  mu <- mean(data[index])
  return (mu)
}
boot(medv, boot.fn, 1000)
```

```
97:1  (Top Level)  R Sc

Console  Terminal
~/
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original       bias    std. error
t1* 22.53281 0.007650791   0.4106622
>
```

The bootstrap estimated standard error of µ^ of 0.4106 is very close to the estimate found in (b) of 0.4089.

(d) **(5 points) 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).**

```r
#4. 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).

t.test(medv)

CI.mu.hat <- c(22.53 - 2 * 0.4119, 22.53 + 2 * 0.4119)
CI.mu.hat
```

111:1 (Top Level)

Console Terminal

```
> t.test(medv)

        One Sample t-test

data:  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 

> CI.mu.hat <- c(22.53 - 2 * 0.4119, 22.53 + 2 * 0.4119)
> CI.mu.hat
[1] 21.7062 23.3538
> 
```

The bootstrap confidence interval is very close to the one provided by the t.test() function.

(e) (5 points) Based on this data set, provide an estimate, ˆµmed, for the median value of medv in the population.

```r
#4. e)Based on this data set, provide an estimate, μ^med, for the median value of “medv” in the population.
med.hat <- median(medv)
med.hat
```

127:1 (Top Level)

Console Terminal

```
> med.hat <- median(medv)
> med.hat
[1] 21.2
> 
```

(f) (5 points) We now would like to estimate the standard error of ˆµmed 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.

```
#4. f)We now would like to estimate the standard error of µ^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.

boot.fn <- function(data, index) {
  mu <- median(data[index])
  return (mu)
}
boot(medv, boot.fn, 1000)
```

118:1 (Top Level)

Console Terminal

```
> boot.fn <- function(data, index) {
+   mu <- median(data[index])
+   return (mu)
+ }
> boot(medv, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original  bias    std. error
t1*     21.2 -0.0386   0.3770241
>
```

From above we get estimated median value of 21.2 which is equal to the value obtained in (e), with a standard error of 0.37702 which is relatively small compared to median value.

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity ˆµ0.1 (You can use the quantile() function.)

```

#4. g)Based on this data set, provide an estimate for the tenth percentile of "medv" in Boston suburbs.
#Call this quantity µ^0.1.
percent.hat <- quantile(medv, c(0.1))
percent.hat
```

139:1 (Top Level)

Console Terminal

```
> percent.hat <- quantile(medv, c(0.1))
> percent.hat
  10%
12.75
>
```

(h) (5 points) Use the bootstrap to estimate the standard error of ˆµmed. Comment on your findings.

```r
#4. h)Use the bootstrap to estimate the standard error of µ^0.1. Comment on your findings.

boot.fn <- function(data, index) {
  mu <- quantile(data[index], c(0.1))
  return (mu)
}
boot(medv, boot.fn, 1000)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original  bias    std. error
t1*    12.75 0.0186   0.4925766
>
```

From above we get an estimated 10% value of 12.75 which is equal to the value obtained in (g), with a standard error of 0.4925 which is relatively small compared to percentile value.