CS 5565, LAB4

(Cross Validation and Bootstrap)

Shubhabrata Mukherjee

Id: 16201097

2) a)

A logistic regression model that uses income and balance to predict default has been fit as below.

```
> defglm= glm(default~income+balance,data = Default, family = binomial)
> defglm

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

Coefficients:
(Intercept)       income      balance
 -1.154e+01    2.081e-05    5.647e-03

Degrees of Freedom: 9999 Total (i.e. Null);  9997 Residual
Null Deviance:      2921
Residual Deviance: 1579        AIC: 1585
>
> defglm.probs=predict(defglm,type ="response")
> defglm.pred = rep ("No" ,nrow(Default))
> defglm.pred[defglm.probs >.5]="Yes"
>
> table(defglm.pred,default)
           default
defglm.pred   No  Yes
        No  9629  225
        Yes   38  108
> mean (defglm.pred == default )
[1] 0.9737
> (1-mean (defglm.pred == default ))*100
[1] 2.63
>
```

So, fitting on training data we get to see that training error is around 2.63%.

b)
Now using validation set approach we get,

i) and ii)

```
> set.seed(1)
> train = sample (10000,2000)
> defglm= glm(default~income+balance,data = Default, family = binomial, subset = train)
> defglm

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

Coefficients:
(Intercept)       income      balance
 -1.237e+01    1.795e-05    6.328e-03

Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
Null Deviance:      586.8
Residual Deviance: 283.7        AIC: 289.7
```

iii) & iv)

```
> defglm.probs=predict(defglm,Default[-train],type ="response")
> defglm.pred = rep ("No",nrow(Default[-train]))
> defglm.pred[defglm.probs >.5]="Yes"
> table(defglm.pred,Default[-train]$default)

defglm.pred   No  Yes
        No  9601  207
        Yes   66  126
> mean (defglm.pred != Default[-train]$default)*100
[1] 2.73
```

So, fitting on training data we get to see that validation set error is around 2.73%.

c)

We now repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Results obtained are below:

2nd iteration :

```
> set.seed(2)
> train = sample (10000,2000)
> defglm= glm(default~income+balance,data = Default, family = binomial, subset = train)
> summary(defglm)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5576  -0.1599  -0.0702  -0.0274   3.5797

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.039e+01  8.922e-01 -11.649   <2e-16 ***
income       1.448e-05  1.138e-05   1.272    0.204
balance      5.087e-03  4.564e-04  11.148   <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: 586.82  on 1999  degrees of freedom
Residual deviance: 337.76  on 1997  degrees of freedom
AIC: 343.76

Number of Fisher Scoring iterations: 8
```

```
> defglm.probs=predict(defglm,Default[-train],type ="response")
> defglm.pred = rep ("No",nrow(Default[-train]))
> defglm.pred[defglm.probs >.5]="Yes"
>
> table(defglm.pred,Default[-train] $default)

defglm.pred   No  Yes
        No  9635  233
        Yes   32  100
> mean (defglm.pred != Default[-train]$default)*100
[1] 2.65
```

So, fitting on training data we get to see that validation set error is around 2.65%.

3rd iteration :

```
> set.seed(3)
> train = sample (10000,2000)
> defglm= glm(default~income+balance,data = Default, family = binomial, subset = train)
> summary(defglm)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0203  -0.1598  -0.0661  -0.0241   3.6527

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.140e+01  9.096e-01 -12.532   <2e-16 ***
income       2.846e-05  1.121e-05   2.539   0.0111 *
balance      5.476e-03  4.625e-04  11.840   <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: 639.66  on 1999  degrees of freedom
Residual deviance: 350.81  on 1997  degrees of freedom
AIC: 356.81

Number of Fisher Scoring iterations: 8
```

```
> defglm.probs=predict(defglm,Default[-train],type ="response")
> defglm.pred = rep ("No",nrow(Default[-train]))
> defglm.pred[defglm.probs >.5]="Yes"
>
> table(defglm.pred,Default[-train] $default)

defglm.pred   No  Yes
        No  9625  226
        Yes   42  107
> mean (defglm.pred != Default[-train]$default)*100
[1] 2.68
```

So, fitting on training data we get to see that validation set error is around 2.68%.

4th iteration :

```
> set.seed(4)
> train = sample (10000,2000)
> defglm= glm(default~income+balance,data = Default, family = binomial, subset = train)
> summary(defglm)

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

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.41407  -0.14270  -0.05776  -0.02249   3.11682

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.169e+01  9.882e-01 -11.829  < 2e-16 ***
income       3.106e-05  1.155e-05   2.688  0.00718 **
balance      5.431e-03  4.973e-04  10.922  < 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: 538.97  on 1999  degrees of freedom
Residual deviance: 296.11  on 1997  degrees of freedom
AIC: 302.11

Number of Fisher Scoring iterations: 8
```

So, fitting on training data we get to see that validation set error is around 2.71%.

```
> defglm.probs=predict(defglm,Default[-train],type ="response")
> defglm.pred = rep ("No",nrow(Default[-train]))
> defglm.pred[defglm.probs >.5]="Yes"
>
> table(defglm.pred,Default[-train] $default)

defglm.pred   No  Yes
        No  9640  244
        Yes   27   89
> mean (defglm.pred != Default[-train]$default)*100
[1] 2.71
```

d)

```
> set.seed(4)
> train = sample (10000,2000)
> defglm= glm(default~income+balance+student,data = Default, family = binomial, subset = train)
> summary(defglm)

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

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.39685  -0.14150  -0.05754  -0.02282   3.09655

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.110e+01  1.176e+00  -9.437   <2e-16 ***
income       1.742e-05  1.935e-05   0.900    0.368
balance      5.452e-03  4.989e-04  10.928   <2e-16 ***
studentYes  -5.124e-01  5.851e-01  -0.876    0.381
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 538.97  on 1999  degrees of freedom
Residual deviance: 295.35  on 1996  degrees of freedom
AIC: 303.35

Number of Fisher Scoring iterations: 8
```

So we saw that, adding dummy variable reduce the error to 2.66%

```
> defglm.probs=predict(defglm,Default[-train],type ="response")
> defglm.pred = rep ("No",nrow(Default[-train]))
> defglm.pred[defglm.probs >.5]="Yes"
>
> table(defglm.pred,Default[-train] $default)

defglm.pred   No  Yes
        No  9641  240
        Yes   26   93
> mean (defglm.pred != Default[-train]$default)*100
[1] 2.66
```

3)a)

Using the summary() and glm() functions, for the logistic regression we get:

```
> set.seed(1)
> train = sample (10000,2000)
> defglm= glm(default~income+balance,data = Default, family = binomial, subset = train)
> summary(defglm)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9106  -0.1188  -0.0433  -0.0149   3.3482

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.237e+01  1.085e+00 -11.393   <2e-16 ***
income       1.795e-05  1.219e-05   1.472    0.141
balance      6.328e-03  5.711e-04  11.079   <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: 586.82  on 1999  degrees of freedom
Residual deviance: 283.73  on 1997  degrees of freedom
AIC: 289.73

Number of Fisher Scoring iterations: 8
```

So, we see the standard errors for the co-efficient is :

```
(1.219e-05)
for income variable

(5.711e-04)
for balance variable
```

b)

Constructing the boot.fn() as per the instruction we see:

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

c)

Using the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for income and balance we get:

```
> library(boot)
> boot(Default, boot.fn, nrow(Default)/5)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = nrow(Default)/5)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01 -1.776094e-02 4.298889e-01
t2*  2.080898e-05 -1.302392e-07 4.728105e-06
t3*  5.647103e-03  1.100047e-05 2.242496e-04
```

So, the estimated the standard errors are:

4.728105e-06 and 2.242496e-04.

d)

for glm()

(1.219e-05) for income variable (5.711e-04) for balance variable.

For boot function:

(4.728105e-06) and (2.242496e-04)

So for bootstrap method, standard errors are less for both the variable.

4)a)

We get population mean of medv as 22.53281

```
> library(MASS)
> names(Boston)
 [1] "crim"    "zn"
[12] "black"   "lstat"
> u = mean(Boston$medv)
> u
[1] 22.53281
```

b)

So we get the standard error as .4088611

```
> se = sd(medv)/sqrt(nrow(Boston))
> se
[1] 0.4088611
```

```
> set.seed(1)
> boot(Boston$medv,boot.fn,1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


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


Bootstrap Statistics :
    original      bias    std. error
t1* 22.53281 0.008517589   0.4119374
```

So we see that, standard error calculated in both error are .4088611 and .4119374, which are pretty close.

d)

```
> con.int = cbind(u -2* 0.4120131,u+2* 0.4120131)
> con.int
         [,1]     [,2]
[1,] 21.70878 23.35683
>
```

```
> t.test(Boston$medv)

	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
```

So, comparing both the value we see both are very close.

e)

```
> mymed = median(medv)
> mymed
[1] 21.2
```

f)

```
> boot.fn = function(data,index) {
+   mymed = median(data[index])
+   return (mymed)
+ }
> boot.fn(Boston$medv,1:nrow(Boston))
[1] 21.2
> set.seed(1)
> boot(Boston$medv,boot.fn,1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


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


Bootstrap Statistics :
    original   bias    std. error
t1*     21.2 -0.01615   0.3801002
```

So, comparing both the value we see both are very close.

g)

h) now using bootstrap we get:

```
> u0.1 = quantile(medv,c(.1))
> u0.1
  10%
12.75
>
```

```
> boot.fn = function(data,index) {
+   u0.1 = quantile(data[index],c(0.1))
+   return (u0.1)
+ }
> boot.fn(Boston$medv,1:nrow(Boston))
  10%
12.75
> set.seed(1)
> boot(Boston$medv,boot.fn,1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


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


Bootstrap Statistics :
    original  bias    std. error
t1*    12.75 0.01005    0.505056
>
```

So, we see that both the value in (g) and (h) are same.