

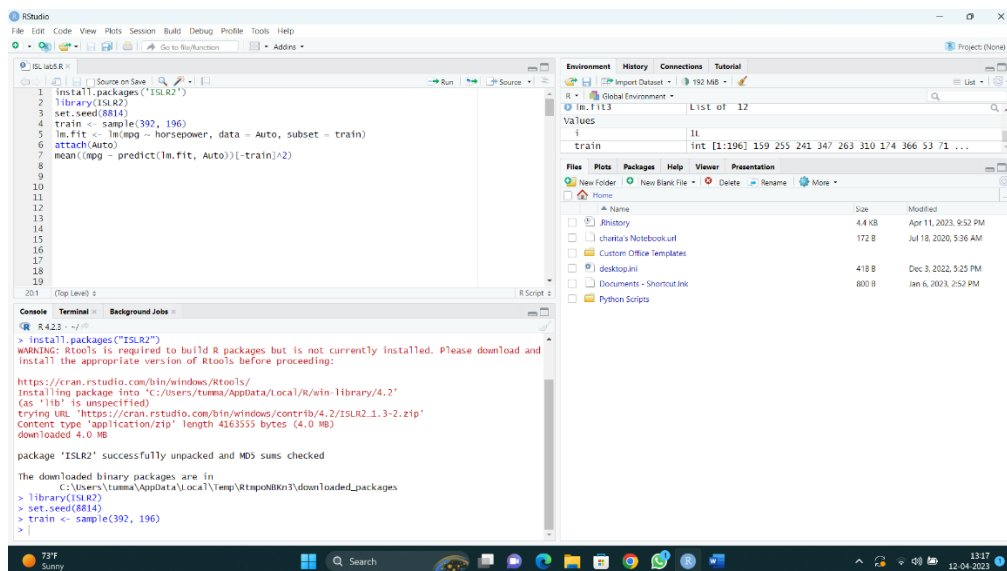
CHARITA TUMMALA – 16338814

ISL ASSIGNMENT

STAT LEARNING LAB: BOOTSTRAP AND CROSS VALIDATION

1) Change the value of the seed for (Validation Set Approach) part 1 to the **last 4 numbers of your student ID** . Report the **test rates** for a Poly fit **quadratic** and **cubic** using the seed.

- last 4 digits of the id number is 8814



The screenshot shows the RStudio interface. The script editor contains the following code:

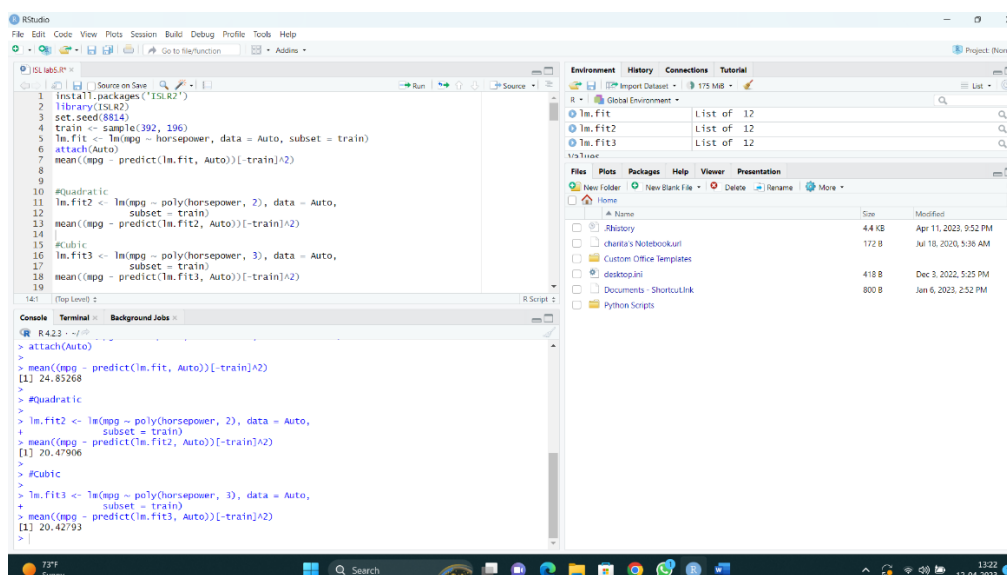
```
1 install.packages("ISLR2")
2 library(ISLR2)
3 set.seed(8814)
4 train <- sample(392, 196)
5 lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
6 attach(Auto)
7 mean(mpg - predict(lm.fit, Auto))[-train]^2
```

The console shows the output of the installation and the execution of the code:

```
> install.packages("ISLR2")
Warning: Rtools is required to build R packages but is not currently installed. Please download and
install the appropriate version of Rtools before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/tumma/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/ISLR2_1.3-2.zip'
Content type 'application/zip' length 416335 bytes (4.0 MB)
downloaded 4.0 MB
package 'ISLR2' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:/Users/tumma/AppData/Local/Temp/RtmpoNBKn3/downloaded_packages
> library(ISLR2)
> set.seed(8814)
> train <- sample(392, 196)
> |
```

The Environment pane shows the 'train' variable as a vector of 196 integers.

Install ISLR2 packages and set seed as last four digits of UMKC ID



The screenshot shows the RStudio interface with the following code in the script editor:

```
1 install.packages("ISLR2")
2 library(ISLR2)
3 set.seed(8814)
4 train <- sample(392, 196)
5 lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
6 attach(Auto)
7 mean(mpg - predict(lm.fit, Auto))[-train]^2
8
9
10 #Quadratic
11 lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto,
12 subset = train)
13 mean(mpg - predict(lm.fit2, Auto))[-train]^2
14
15 #Cubic
16 lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto,
17 subset = train)
18 mean(mpg - predict(lm.fit3, Auto))[-train]^2
19
```

The console shows the output of the code:

```
> attach(Auto)
> mean(mpg - predict(lm.fit, Auto))[-train]^2
[1] 24.85268
> #Quadratic
> lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto,
+ subset = train)
> mean(mpg - predict(lm.fit2, Auto))[-train]^2
[1] 20.47906
> #Cubic
> lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto,
+ subset = train)
> mean(mpg - predict(lm.fit3, Auto))[-train]^2
[1] 20.42793
> |
```

The Environment pane shows the 'train' variable as a vector of 196 integers.

The poly quadratic and cubic values for the data using the seed(8814) is 20.47906, 20.42793

lm.fit	List of 12	Q
lm.fit2	List of 12	Q
lm.fit3	List of 12	Q
Values		
i	1L	
train	int [1:196] 159 255 241 347 263 310 174 366 53 71 ...	

```

>
>
> boot.fn <- function(data,index)
+   coef(lm(mpg ~ horsepower, data = data, subset = index))
>
> boot.fn(Auto,1:392)
(Intercept)  horsepower
39.9358610   -0.1578447
>

```

2) Change the ratio of train/test to a new percentage. (original was 50/50) Select **3 new ratios** and compute and **report** the **test performance** for each using the **best performing** of the two Poly fit values from task 1.

to create a 60/40 split:

- The ratio of train/test is 392/235

```

> set.seed(8814)
> train <- sample(392, 235)
> lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
> mean((mpg - predict(lm.fit, Auto))[-train]^2)
[1] 27.25061
> lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto,
+               subset = train)
> mean((mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 22.17392
> lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto,
+               subset = train)
> mean((mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 22.12438
> |

```

to create a 70/30 split:

- The ratio of train/test is 392/274.

```

> set.seed(8814)
> train <- sample(392, 274)
> lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
> mean((mpg - predict(lm.fit, Auto))[-train]^2)
[1] 27.66704
> lm.fit2 <- lm(mpg ~ poly(horsepower, 2),
+               subset = train)
> mean((mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 21.98854
> lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto,
+               subset = train)
> mean((mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 21.9517
> |

```

to create a 80/20 split:

- The ratio of train/test is 392/314.

```

> set.seed(8814)
> train <- sample(392, 314)
> lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
> mean((mpg - predict(lm.fit, Auto))[-train]^2)
[1] 27.77654
> lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)
>
> mean((mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 21.75094
> lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)
> mean((mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 21.67995
>

```

Both the linear and cubic values in the data using the seed (8814) are rising. The quadratic values, on the other hand, are falling.

3) The LOOCV used in the lab compares poly order using horsepower for mpg prediction. Compute and **report** the overall performance of **Poly** for orders **1:8** for any **one** of the other features in the Auto dataset (**acceleration**, **cylinders**, **displacement**, **weight**) to **mpg**.

Here, In this step I have used displacement feature from the AUTO dataset for the LOOCV,

```

>
> glm.fit <- glm(mpg ~ displacement, data = Auto)
> coef(glm.fit)
(Intercept) displacement
35.12063594 -0.06005143
> lm.fit <- lm(mpg ~ displacement, data = Auto)
> coef(lm.fit)
(Intercept) displacement
35.12063594 -0.06005143
> library(boot)
> glm.fit <- glm(mpg ~ displacement, data = Auto)
>
> cv.err <- cv.glm(Auto, glm.fit)
> cv.err$delta
[1] 21.59246 21.59218
> cv.error <- rep(0,8)
> for(i in 1:8){
+   glm.fit <- glm(mpg ~ poly(displacement, i), data = Auto)
+   cv.error[i] <- cv.glm(Auto, glm.fit)$delta[1]
+ }
> cv.error
[1] 21.59246 19.15356 19.19299 19.29885
[5] 19.36118 19.17039 18.73462 18.35266
>

```

The report for the overall performance of poly for orders 1:8 (Original is 1:10 in the given LAB-5 code)

Standard deviation is rising, and there is no clear pattern in bias.

4) Select a **different** feature from Auto than you did in task 3 and perform a **5** fold and **10** fold **k-fold cross validation**.

- I have selected year feature from the AUTO dataset and performed 5 fold and 10 fold K-fold cross validation.

```

> set.seed(8814)
> cv.error.5 <- rep(0,5)
> for(i in 1:5){
+   glm.fit <- glm(mpg ~ poly(year, i), data = Auto)
+   cv.error.5[i] <- cv.glm(Auto, glm.fit, K = 5)$delta[1]
+ }
> cv.error.5
[1] 40.67819 38.89804 39.61127 38.88682
[5] 39.13951
>

```

```

> set.seed(8814)
> cv.error.10 <- rep(0,10)
> for(i in 1:10){
+ glm.fit <- glm(mpg ~ poly(year, i), data = Auto)
+ cv.error.10[i] <- cv.glm(Auto, glm.fit, K = 10)$delta[1]
+ }
> cv.error.10
[1] 40.49991 38.91699 39.16581 38.99626
[5] 39.19729 38.87759 38.20843 39.10293
[9] 38.18710 36.93303
>

```

Values

cv.error	num [1:8] 21.6 19.2 19.2 19.3 19.4 ...
cv.error.10	num [1:10] 40.5 38.9 39.2 39 39.2 ...
cv.error.5	num [1:5] 40.7 38.9 39.6 38.9 39.1

5) Compute the last bootstrap exercise (the quadratic fit for horsepower) from the lab **3 more times**, using a **new number of samples {250, 500, 2500}**. The goal of this task will be to **compare the error estimates** of the increasing number of samples. **Report your observations** about the **error** as the number of bootstrap sets **increases**.

```

> boot.fn <- function(data, index)
+ coef(
+ lm(mpg ~ horsepower + I(horsepower^2),
+ data = data, subset = index)
+ )
> set.seed(8814)
> boot(Auto, boot.fn, 250)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto, statistic = boot.fn, R = 250)

Bootstrap Statistics :
      original      bias      std. error
t1* 56.900099702  5.894938e-02  2.1378396034
t2* -0.466189630 -2.599251e-04  0.0337456010
t3*  0.001230536 -1.214037e-06  0.0001202822
>

```

```

> boot.fn <- function(data, index)
+ coef(
+ lm(mpg ~ horsepower + I(horsepower^2),
+ data = data, subset = index)
+ )
> set.seed(8814)
> boot(Auto, boot.fn, 500)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto, statistic = boot.fn, R = 500)

Bootstrap Statistics :
      original      bias      std. error
t1* 56.900099702  7.845673e-02  2.0038933601
t2* -0.466189630 -7.230106e-04  0.0323349758
t3*  0.001230536  1.431625e-06  0.0001178305
>

```

```
> boot.fn <- function(data, index)
+   coef(
+     lm(mpg ~ horsepower + I(horsepower^2),
+       data = data, subset = index)
+   )
> set.seed(8814)
> boot(Auto, boot.fn, 2500)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto, statistic = boot.fn, R = 2500)

Bootstrap Statistics :
      original      bias      std. error
t1* 56.900099702  8.404618e-02  2.1140203797
t2* -0.466189630 -1.396053e-03  0.0338202898
t3*  0.001230536  5.258816e-06  0.0001229194
> |
```

From the three sample and from the seed(8814), I have observed that the Standard Deviation and bias of the sample 250, 2500 are higher than the 500.