

*Індивідуальне завдання №4*  
Перехресна перевірка. Бутстрап

Виконала:  
студентка групи ПМОМ-11  
Костецька Уляна

## Варіант 8

Спочатку встановлюю значення змінної variant: для цього  $0(\text{номер групи}) * 25 + 8(\text{порядковий номер в списку групи}) = 8$ .

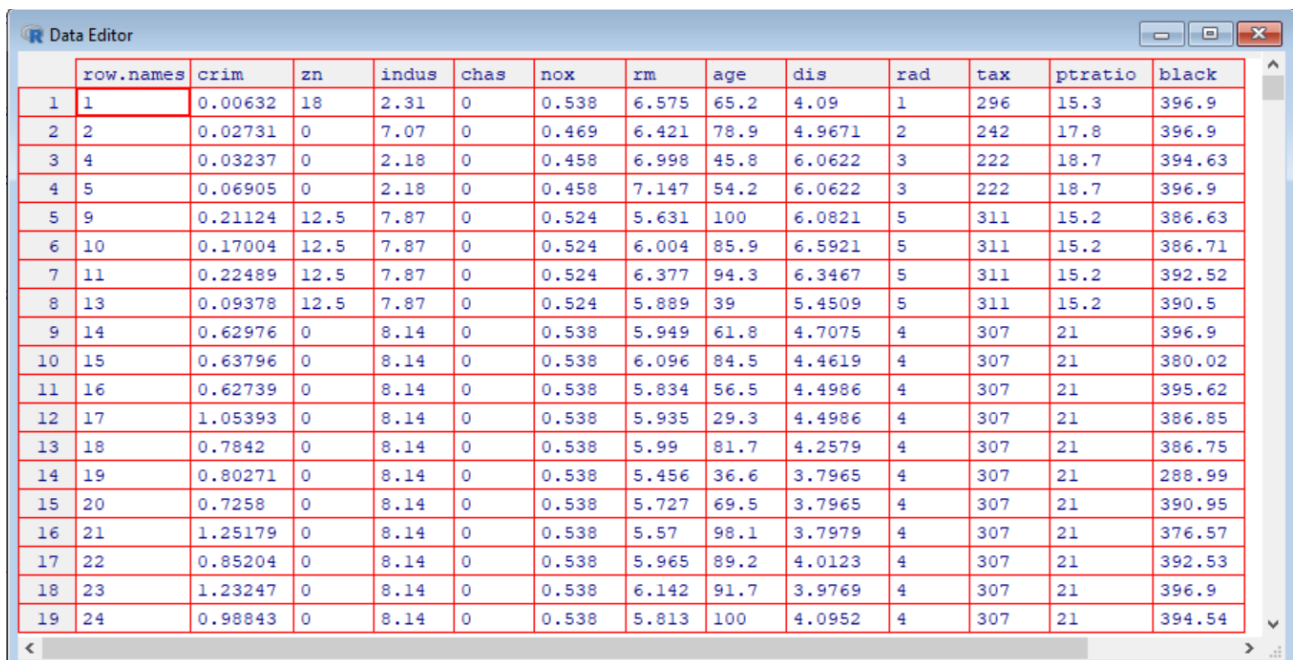
Встановлюю `set.seed(variant)` та генерую значення змінної `redundant` як заокруглене до цілого випадкове число взяте з рівномірного інтервалу (5, 25) розподілу. Для заокруглення використовую функцію `floor`, а для вибору випадкового числа з інтервалу функцію `runif`.

### 1. Завантажую бібліотеку MASS, для подальшої роботи з Boston

```
> variant=8
> variant
[1] 8
> set.seed(variant)
> redundand=floor(runif(1,5,25))
> redundant
[1] 14
> library(MASS)
```

Використовую функцію `sample()` для модифікації завантажених даних Boston - видалення `redundant` (% спостережень).

```
> Boston_new=Boston[-sample(1:length(Boston[,1]),round((redundant/100)*length(Boston[,1]))),]
> fix(Boston_new)
```



	row.names	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
1	1	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9
2	2	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9
3	4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	5	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9
5	9	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63
6	10	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71
7	11	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52
8	13	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5
9	14	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9
10	15	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02
11	16	0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62
12	17	1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85
13	18	0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75
14	19	0.80271	0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21	288.99
15	20	0.7258	0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21	390.95
16	21	1.25179	0	8.14	0	0.538	5.57	98.1	3.7979	4	307	21	376.57
17	22	0.85204	0	8.14	0	0.538	5.965	89.2	4.0123	4	307	21	392.53
18	23	1.23247	0	8.14	0	0.538	6.142	91.7	3.9769	4	307	21	396.9
19	24	0.98843	0	8.14	0	0.538	5.813	100	4.0952	4	307	21	394.54

Пристосовую модель логістичної регресії для передбачення рівня злочинності(`crim_1`) більшого чи меншого за середній на основі змінних `nox` та `rad`. Для оцінки тестової помилки цієї моделі логістичної регресії використовую метод валідаційного набору.

```
> train=sample(435,217)
> LogR_1 = glm(crim_1~nox+rad,data=Boston_new,subset = train, family='binomial')
> prob_1 = predict(LogR_1,Boston_new[-train,1],type='response')
> PredCrim_1 = ifelse(prob_1>0.5,1,0)
> mean(PredCrim_1!=Boston_new$crim_1[-train])
[1] 0.01834862
```

Повторюю попередню процедуру ще два рази, використовуючи нові розбиття вибірки на навчальний та тестовий набори.

```
> train=sample(435,217)
> LogR_1 = glm(crim_1~nox+rad,data=Boston_new,subset = train, family='binomial')
> prob_1 = predict(LogR_1,Boston_new[-train,],type='response')
> PredCrim_1 = ifelse(prob_1>0.5,1,0)
> mean(PredCrim_1!=Boston_new$crim_1[-train])
[1] 0.02293578
```

```
> train=sample(435,217)
> LogR_1 = glm(crim_1~nox+rad,data=Boston_new,subset = train, family='binomial')
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> prob_1 = predict(LogR_1,Boston_new[-train,],type='response')
> PredCrim_1 = ifelse(prob_1>0.5,1,0)
> mean(PredCrim_1!=Boston_new$crim_1[-train])
[1] 0.02752294
```

Усі результати вийшли непоганими але різними, за рахунок того, що `sample()` при кожному запуску бере різні індекси елементів для поділу даних на тренувальні та тестові.

Роблю все те саме, але на основі змінних `nox`, `rad` та `medv`. Отримую майже такі ж результати, а це означає що змінна `medv` не є значущою для моделі.

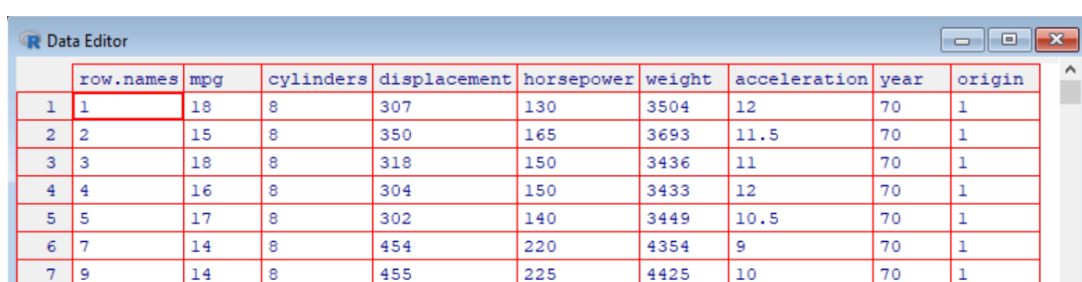
```
> train=sample(435,217)
> LogR_2 = glm(crim_1~nox+rad+medv,data=Boston_new,subset = train, family='binomial')
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> prob_2 = predict(LogR_2,Boston_new[-train,],type='response')
> Pred_2Crim_1 = ifelse(prob_2>0.5,1,0)
> mean(Pred_2Crim_1!=Boston_new$crim_1[-train])
[1] 0.03211009
> |
```

```
> train=sample(435,217)
> LogR_2 = glm(crim_1~nox+rad+medv,data=Boston_new,subset = train, family='binomial')
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> prob_2 = predict(LogR_2,Boston_new[-train,],type='response')
> Pred_2Crim_1 = ifelse(prob_2>0.5,1,0)
> mean(Pred_2Crim_1!=Boston_new$crim_1[-train])
[1] 0.02752294
```

```
> train=sample(435,217)
> LogR_2 = glm(crim_1~nox+rad+medv,data=Boston_new,subset = train, family='binomial')
> prob_2 = predict(LogR_2,Boston_new[-train,],type='response')
> Pred_2Crim_1 = ifelse(prob_2>0.5,1,0)
> mean(Pred_2Crim_1!=Boston_new$crim_1[-train])
[1] 0.01834862
> |
```

2. Завантажую бібліотеку ISLR, для подальшої роботи з Auto. Використовую функцію `sample()` для модифікації завантажених даних Auto - видалення `redundant` (% спостережень).

```
> set.seed(variant)
> library(ISLR)
> Auto_new=Auto[-sample(1:length(Auto[,1]),round((redundant/100)*length(Auto[,1]))),]
> fix(Auto_new)
> |
```



	row.names	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
1	1	18	8	307	130	3504	12	70	1
2	2	15	8	350	165	3693	11.5	70	1
3	3	18	8	318	150	3436	11	70	1
4	4	16	8	304	150	3433	12	70	1
5	5	17	8	302	140	3449	10.5	70	1
6	7	14	8	454	220	4354	9	70	1
7	9	14	8	455	225	4425	10	70	1

На основі набору даних Auto\_new обчислюю оцінку середнього змінної mpg і стандартну похибку цієї оцінки.

```
> mean(Auto_new$mpg)
[1] 23.49822
> sd(Auto_new$mpg)/sqrt(length(Auto_new$mpg))
[1] 0.4249068
> |
```

Далі оцінюю стандартну похибку розглянутої вище оцінки середнього за допомогою **бутстрапу**.

Для цього спочатку прописую функцію, яка повертає середнє значення змінної для вибраних індексів. Результат такий самий, отже функція рахує правильно.

```
> fun_mean = function(data,index){
+   return(mean(data[index]))}
> fun_mean(Auto_new$mpg,1:length(Auto_new$mpg))
[1] 23.49822
> |
```

Після чого використовую функцію boot для оцінки стандартної похибки. В результаті отримую дуже схожі значення.

```
> library(boot)
> boot(Auto_new$mpg,fun_mean,R=1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto_new$mpg, statistic = fun_mean, R = 1000)

Bootstrap Statistics :
   original      bias   std. error
t1*  23.49822 -0.01321246   0.4374281
```

Обчислюю оцінку для медіани та десятого процентиля змінної mpg.

```
> median(Auto_new$mpg)
[1] 23
> quantile(Auto_new$mpg,0.1)
10%
14
> |
```

Оцінюю стандартні помилки отриманих вище оцінок з допомогою **бутстрапу**.

Стандартна похибка для **медіани** дорівнює **0.8132413**

```
> fun_med = function(data,index){
+   return(median(data[index]))}
> fun_med(Auto_new$mpg,1:length(Auto_new$mpg))
[1] 23
> boot(Auto_new$mpg,fun_med,R=1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto_new$mpg, statistic = fun_med, R = 1000)

Bootstrap Statistics :
   original      bias   std. error
t1*      23     -0.13   0.8132413
> |
```

Стандартна похибка для десятого процентиля дорівнює **0.4274457**

```
> fun_q10 = function(data,index){
+   return(quantile(data[index],0.1))}

> fun_q10(Auto_new$mpg,1:length(Auto_new$mpg))
10%
14
> boot(Auto_new$mpg,fun_q10,R=1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto_new$mpg, statistic = fun_q10, R = 1000)

Bootstrap Statistics :
      original    bias      std. error
t1*          14 -0.1233    0.4274457
> |
```

3. Встановлюю seed, що дорівнює значенню змінної variant та створюю змодельований набір даних так як вказано в завданні.

Далі обчислюю оцінки тестових помилок методом **LOOCV** для вказаних в завданні помилок.

Як і очікувалось, модель **2** має найменшу тестову оцінку, оскільки дані генерувалися з моделі 2-го порядку.

```
> set.seed(variant)
> x = rnorm (100)
> y = variant*x - ((redundant*40)/variant) * x ^ 2 + rnorm (100)
> set.seed(variant)
> cv.error = rep(0,4)

> for(i in 1:4){
+   lr = glm(y~poly(x,i,raw=T))
+   cv.error[i]=cv.glm(data.frame(x,y),lr)$delta[1]}
> which.min(cv.error)
[1] 2
> cv.error
[1] 15406.767882    1.192845    1.209952    1.240460
> |
```