ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Звіт до індивідуального завдання № 4
з предмету
«Моделі статистичного навчання»

Виконав:

студент групи ПМіМ-13с

Кіс Юрій

Викладач:

проф. Тарас Заболоцький

2023

Хід роботи

Роботу починаю так, як і вказано у завданні – встановлюю значення змінної variant, моя група ПМіМ-13с (якій відповідає числове значення 3), мій порядковий номер у списку студентів 7й. Обчислюю variant, задаю set.seed(variant) та генерую випадкове число redundant:

```
> group_number <- 3
> student_position <- 7
> variant <- group_number * 25 + student_position
> set.seed(variant)
> redundant <- round(runif(1, group_number + 5, 25 - group_number))
```

Оскільки я використовую RStudio одразу можу побачити отримані значення variant та redundant:

Values	
group_number	3
redundant	10
student_position	7
variant	82

Окрім того, добавлю ті бібліотеки, які використовуватиму протягом всього завдання:

```
> library(MASS)
> library(ISLR)
> library(boot)
```

Завдання 1.

Розглянемо набір даних Boston з бібліотеки MASS. Модифікуйте ці дані наступним чином: встановивши seed, що дорівнює значенню змінної variant, видаліть redundant % спостережень з допомогою функції sample.

В першу чергу, як вказано у завданні, відсію зайві значення з набору даних Boston:

```
> set.seed(variant)
> boston_redundant_amount <- floor(redundant / 100 * nrow(Boston))
> boston_data <- Boston[-sample(1:nrow(Boston), boston_redundant_amount),]
```

Data	
boston_data	456 obs. of 14 variables
Values	
boston_redundant_amount	50

Використовуючи модифіковані дані, пристосуйте модель логістичної регресії для передбачення у вибраному районі рівня злочинності більшого чи меншого за середній на основі змінних nox та rad. Оцініть тестову помилку цієї моделі логістичної регресії, використовуючи метод валідаційного набору (використати розбиття 50 на 50, попередньо скинувши seed). Повторіть попередню процедуру три рази, використовуючи нові розбиття вибірки на навчальний та тестовий набори. Прокоментуйте отримані результати.

За рівень злочинності у наборі даних Boston відповідає стовпець crim. Оскільки нам треба за ним пристосувати модель логістичної регресії для передбачення – він має бути бінарним, і як вказано у завданні, більшим чи мешим за середнє. Підготуємо бінарний стовпець crim01 та внесемо його у датафрейм boston_data:

```
> crim_mean <- mean(boston_data$crim)
> crim01 <- rep(0, nrow(boston_data))
> crim01[boston_data$crim > crim_mean] <- 1
> boston_data <- data.frame(boston_data, crim01)
```

Встановлю set.seed(variant). Після цього, використовуючи метод валідаційного набору, розіб'ю наші дані на дві рівні частини, пристосую модель логістичної регресії, проведу передбачення та оціню тестову помилку чотири рази:

```
> set.seed(variant)
> boston_nrow <- nrow(boston_data)
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad, data=boston_data[boston_half,], family=binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  2.631579
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad, data=boston_data[boston_half,], family=binomial)
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  1.754386
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad, data=boston_data[boston_half,], family=binomial)
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  2.192982
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad, data=boston_data[boston_half,], family=binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  2.192982
```

Із отриманих результатів – при зміні розбиття, змінювалась тестова похибка. При отриманні warning повідомлення вперше, тестова похибка сягнула 2.63%. Значення третього та четвертого тесту виявились ідентичними, що дуже цікаво, оскільки четвертий тест вивів warning повідомлення, як і перший тест, а похибка була аналогічна третьому тесту – 2.19%, де warning повідомлення відсутнє. Значення другого тесту без вищезгаданого повідомлення про помилку пристосування моделі логістичної регресії найменше – 1.75%. Середня похибка чотирьох тестів – 2.19%. Класні цифри у нас получились :) Із цих даних можу припустити, що пристосування моделей, які виводитимуть warning повідомлення будуть мати вищу похибку.

Розгляньте модель логістичної регресії для передбачення у вибраному районі рівня злочинності більшого чи меншого за середній на основі змінних nox, rad та medv. Оцініть тестову помилку для цієї моделі, використовуючи метод валідаційного набору описаний вище. Прокоментуйте, чи призводить включення нової змінної до зменшення тестової помилки.

Проведу чотири аналогічні тести, але для передбачення crim01 використовуватиму змінні nox, rad та medv:

```
> set.seed(variant)
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad+medv, data=boston_data[boston_half,], family=binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  2.631579
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad+medv, data=boston_data[boston_half,], family=binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  2.631579
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad+medv, data=boston_data[boston_half,], family=binomial)
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  2.192982
>
> boston_half <- sample(1:boston_nrow, boston_nrow / 2)
> log_boston <- glm(crim01~nox+rad+medv, data=boston_data[boston_half,], family=binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> est_boston <- predict(log_boston, boston_data[-boston_half,], type="response")
> cat("Error: ", mean(boston_data$crim01[-boston_half] != (est_boston > 0.5)) * 100)
Error:  2.192982
```

Цього разу результати тестів 1, 3 та 4 залишились незмінними, проте тест номер 2 отримав warning повідомлення, а його результат став аналогічним першому тесту. Повторивши попередній та цей крок ще один раз я отримав аналогічні результати, проте спробувавши перевірити інші вибірки (із іншими значенням redundant), я не отримав схожих наслідків, тож вважатиму, що мені повезло з варіантом. Із наведених рішень для warning повідомлення в інтернеті я дізнався, що можливими рішеннями є видалення проблемних значень, збільшення розміру вибірки чи просто не зважати на цю помилку, оскільки регресія все одно має працювати коректно навіть попри неї. Моє припущення – medv не має значного впливу на результати логістичної регресії, проте його додавання може вплинути на процес пристосування, внаслідок чого одну чи більше змінних не можна буде відрізнити від 0 чи 1 (про що й говорить помилка).

Завдання 2.

Модифікуйте дані Auto наступним чином: встановивши seed, що дорівнює значенню змінної variant, видаліть redundant % спостережень з допомогою функції sample.

```
> set.seed(variant)
> auto_nrow <- nrow(Auto)
> auto_redundant_amount <- floor(redundant / 100 * auto_nrow)
> auto_data <- Auto[-sample(1:auto_nrow, auto_redundant_amount),]
> auto_nrow <- nrow(auto_data)
```

На основі цього набору даних обчисліть оцінку середнього змінної mpg. Оцініть стандартну похибку цієї оцінки. Тепер оцініть стандартну похибку розглянутої вище оцінки середнього за допомогою бутстрапу та порівняйте з попередньо отриманим результатом.

Оскільки в подальшому я працюватиму саме з mpg, винесу його в окрему змінну, також для використання бутстрапу треба підготувати функцію. Її ж можна буде використати для обчислення оцінки середнього статистичного:

```
> auto_mpg <- auto_data$mpg
> func_mean <- function(data, idx) { mean(data[idx]) }
> func_mean(auto_mpg, 1:auto_nrow)
[1] 23.37932
```

Тепер обчислю середнє квадратичне:

```
> sd(auto_mpg) / sqrt(auto_nrow)
[1] 0.4155861
```

І використаю бутстрап:

```
> boot_calls <- 5000
> boot(auto_mpg, func_mean, boot_calls)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = auto_mpg, statistic = func_mean, R = boot_calls)


Bootstrap Statistics :
    original      bias    std. error
t1* 23.37932 0.000231728   0.4133656
```

Середнє статистичне отримане у двох результатах рівне одне одному, а стандартна похибка збігається до двох знаків після коми. При збільшенні boot_calls збіг між стандартними похибками продовжуватиме рости (при R = 10000, std. error = 0.415128, збіг трьох знаків після коми).

Обчисліть оцінку для медіани та десятого процентиля змінної mpg. Оцініть стандартні помилки отриманих оцінок допомогою бутстрапу.

Проведу схожі дії для медіани та десятого процентиля:

```
> func_median <- function(data, idx) { median(data[idx]) }
> func_quantile <- function(data, idx) { quantile(data[idx], c(0.1)) }
>
> func_median(auto_mpg, 1:auto_nrow)
[1] 22.3
> func_quantile(auto_mpg, 1:auto_nrow)
10%
 14
>
> boot(auto_mpg, func_median, boot_calls)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = auto_mpg, statistic = func_median, R = boot_calls)


Bootstrap Statistics :
    original  bias    std. error
t1*     22.3  0.1654   0.7899289
> boot(auto_mpg, func_quantile, boot_calls)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = auto_mpg, statistic = func_quantile, R = boot_calls)


Bootstrap Statistics :
    original   bias    std. error
t1*       14 -0.11112   0.3922539
```

У цьому випадку отримані значення медіани та десятого процентиля також збігаються (медіана – 22.3, десятий процентиль – 14). Стандартна похибка медіани 0.79, стандартна похибка десятого процентиля 0.39.

Завдання 3.

Встановлю x та y як вказано в умові завдання:

```
> set.seed(variant)
> x <- rnorm(100)
> y <- variant*x - ((redundant*40)/variant) * x ^ 2 + rnorm(100)
```

Виконаю оцінку тестових помилок методом LOOCV для чотирьох заданих моделей:

```
> yx_data <- data.frame(y, x)
>
> for (i in 1:4) {
+ set.seed(variant)
+ poly_yx <- glm(y~poly(x, i, raw=TRUE))
+ loocv_result <- cv.glm(yx_data, poly_yx)
+ cat("Error (", i, "):", loocv_result$delta[1], '\n')
+ }
Error ( 1 ): 39.65404
Error ( 2 ): 1.023581
Error ( 3 ): 0.9992304
Error ( 4 ): 1.034324
```

Яка з моделей має найменшу тестову помилку LOOCV? Чи це відповідає очікуванням? Поясніть свою відповідь.

Найменшу тестову помилку має третя модель, що не відповідає очікуванням. Фактично третя модель у завданні описана так:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$$

Проте якщо ми звернемо увагу на те, як ми генеруємо y, то побачимо:

$$> y = variant*x - ((redundant*40)/variant) * x ^ 2 + rnorm(100),$$

тобто фактична формула більше виглядає як b*x + a*x^2 + c, якій відповідає друга модель – «квадратичне» рівняння (із епсилон):

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$$

Приємно вже те, що друга модель має другу найменшу похибку. Окрім того, вирізняється перша модель, яка позначає лінійне рівняння. Оскільки у залежить від x не лінійно – звідси й така велика похибка.