МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

ЗВІТ

до індивідуального завдання №4

з дисципліни «Моделі статистичного навчання»

Виконав
студент групи ПМіМ-12:
Бордун Михайло

Перевірив:
Проф. Заболоцький Т. М.

Львів – 2021

Хід виконання

1. Використання логістичної регресії для прогнозування ймовірності дефолту на основі income та balance з даних Default та оцінка тестової помилки цієї моделі, використовуючи метод валідаційного набору.

```
default    student       balance          income
No :9667   No :7056   Min.   :   0.0   Min.   :  772
Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340
                      Median : 823.6   Median :34553
                      Mean   : 835.4   Mean   :33517
                      3rd Qu.:1166.3   3rd Qu.:43808
                      Max.   :2654.3   Max.   :73554
```

На рисунку вище наведені загальні властивості змінних з датасету Default: як бачимо ми маємо дві якісні та відповідно дві кількісні змінні, а саме balance та income.

1.1 Побудовано логістичну регресійну модель, яка використовує income та balance для передбачення default.

```
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
```

1.2 Розділено вибірку на навчальний та тестовий набори використовуючи функцію sample(), генерування відбувалося наступним чином:

```
train = sample(length(balance), length(balance) / 2)
Default.test = Default[-train, ]
```

При чому, варто зауважити що попередньо було встановлено set.seed(1). Оцінено логістичну регресійну модель, використовуючи навчальну вибірку.

```
                 Estimate   Std. Error    z value      Pr(>|z|)
(Intercept) -1.194413e+01 6.177962e-01 -19.333444 2.810180e-83
income       3.262025e-05 7.023514e-06   4.644434 3.410095e-06
balance      5.689218e-03 3.158234e-04  18.013920 1.515093e-72
```

З огляду на наведені значення p-value бачимо, що обидві змінні мають значущий вплив на залежну змінну.

Виконано прогнозування дефолт-статусу для кожної людини в тестовій вибірці на основі передбачення апостеріорної ймовірності дефолту для цієї людини.

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

Оцінено тестову помилку на валідаційній множині шляхом обчислення частки статусу осіб, які неправильно класифіковані. В результаті тестова помилка склала 2.54%.

```
print(paste("Тестова помилка: ", mean(pred.glm != Default.test$default) * 100, "%"))
```

```
"Тестова помилка:  2.54 %"
```

1.3 Повторив попереднє завдання три рази, використовуючи три різні розбиття вибірки на навчальний та тестовий набори. Це було виконано взявши код до попереднього завдання у функцію, яку було викликано у циклі. Булівський аргумент передано для того, щоб не викликати кожен раз опис регресійної моделі.

```
for (i in (0:2)) {
  TestError(FALSE)
}
```

З огляду на наведені нижче результати можна сказати, що немає кардинальних змін через постійно однаковий розмір тестових та тренувальних вибірок (а саме 5000 значень) та відсутність аномальних відхилень у нашій загальній вибірці, однак точність очевидно відрізняється через те, що кожен раз в нас рандомно беруться різні спостереження для наших наборів.

```
"Тестова помилка:  2.74 %"
"Тестова помилка:  2.44 %"
"Тестова помилка:  2.44 %"
```

1.4 Розглянемо модель логістичної регресії, яка передбачає ймовірність дефолту за допомогою змінних income, balance та фіктивної змінної для student.

```
fit.glm = glm(default ~ income + balance + student,
 data = Default, family = "binomial", subset = train)
```

Було оцінено тестову помилку моделі використовуючи метод валідаційного набору як і в попередніх завданнях. В результаті тестова помилка склала 2.78%, що є свідченням недоцільності додавання фіктивної змінної в нашу модель, оскільки без неї (використовуючи 4 різних розбиття на навчальний та тестовий набори) точність була вищою.

2. Аналіз стандартного відхилення оцінок параметрів моделі логістичної регресії, використовуючи: бутстрап; стандартну формулу функції glm().

2.1 Використовуючи функції summary() та glm(), визначено оцінку середньоквадратичного відхилення параметрів логістичної регресії, яка використовує income та balance для оцінки ймовірності дефолту.

```
                 Estimate   Std. Error    z value      Pr(>|z|)
(Intercept) -1.154047e+01 4.347564e-01 -26.544680 2.958355e-155
income       2.080898e-05 4.985167e-06   4.174178  2.990638e-05
balance      5.647103e-03 2.273731e-04  24.836280 3.638120e-136
```

В результаті маємо дані з колонки Std.Error (середньоквадратине відхилення) для коефіцієнтів β_0, β_1 та β_2 відповідно 4.35^{-1}, 4.99^{-6} та 2.27^{-4}.

2.2 Написано функцію boot.fn(), яка приймає на вхід набір даних та індекси спостережень для використання і виводить оцінки коефіцієнтів логістичної регресії, яка використовує income та balance для оцінки ймовірності дефолту.

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

2.3-2.4 Використано функцію boot() з однойменної бібліотеки разом із функцією boot.fn() з попереднього пункту. Це було виконано для оцінки середньоквадратичного відхилення параметрів логістичної регресії, яка використовує income та balance для оцінки ймовірності дефолту.

```
print(boot(Default, boot.fn, 100))
```

```
Call:
boot(data = Default, statistic = boot.fn, R = 100)


Bootstrap Statistics :
        original        bias     std. error
t1* -1.154047e+01  8.556378e-03 4.122015e-01
t2*  2.080898e-05 -3.993598e-07 4.186088e-06
t3*  5.647103e-03 -4.116657e-06 2.226242e-04
```

Як бачимо було взято R=100, тобто 100 повторень бутстрапу і у підсумку маємо наступні значення середньоквадратичного відхилення для коефіцієнтів β_0, β_1 та β_2 відповідно 4.12^{-1}, 4.19^{-6} та 2.23^{-4}.

Тобто, оцінки виявилися досить схожими, однак бутстрап оцінив менші середньоквадратичні відхилення для параметрів нашої моделі.

3. Обчислення оцінки тестової помилки методом LOOCV для логістичної регресійної моделі на наборі даних Weekly, використовуючи лише функції glm(), predict.glm() та цикл for.

Розглянемо датасет Weekly. Для кожної дати наявна дохідність для попередніх 5 днів Lag1,…,Lag5. Змінна Volume містить дані про обсяг торгів попереднього дня у млн., Today – сьогоднішня дохідність, та змінна Direction, яка вказує чи зріс ринок чи впав.

```
     Year          Lag1              Lag2              Lag3
Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
     Lag4              Lag5              Volume            Today
Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
Direction
Down:484
Up  :605
```

3.1 Побудовано модель логістичної регресії, яка передбачає Direction за допомогою змінних Lag1 та Lag2.

```
               Estimate Std. Error   z value     Pr(>|z|)
(Intercept)  0.22122405 0.06146572  3.599145 0.0003192652
Lag1        -0.03872222 0.02621658 -1.477013 0.1396722362
Lag2         0.06024830 0.02654589  2.269590 0.0232324586
```

З огляду на наведені значення p-value бачимо, що тільки змінна Lag2 має значущий вплив на залежну змінну.

3.2 Побудовано модель логістичної регресії, яка передбачає Direction за допомогою змінних Lag1 та Lag2, використовуючи всі спостереження, крім першого.

```
fit.glm2 = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = "binomial")
print(summary(fit.glm2)$coef)
```

```
               Estimate Std. Error   z value     Pr(>|z|)
(Intercept)  0.22324305 0.06149894  3.630031 0.0002833875
Lag1        -0.03843317 0.02621860 -1.465874 0.1426825151
Lag2         0.06084763 0.02656088  2.290874 0.0219707105
```

Очікувано результат практично не змінився, оскільки було відкинуто тільки одне спостереження з вибірки з 1089 спостережень.

3.3 Використано модель з пункту 3.2, щоб передбачити Direction для першого спостереження.

```
     Up
Down  0
Up    1
```

З огляду на результат виконання функції contrasts(), бачимо, що для змінної Up асоційоване значення 1, тобто якщо результат прогнозування для нашої моделі буде більшим за 0.5, то значить спостереження класифіковано як Up для змінної Direction. Логіку перевірки правильності класифікації наведено нижче.

```
if (predict.glm(fit.glm2, Weekly[1, ], type = "response") > 0.5) {
  if (Direction[1] == "Up") {
    print("Правильне прогнозування Direction для першого спостереження")
  }
  else { print("Неправильне прогнозування Direction для першого спостереження") }
}
```

```
[1] "Неправильне прогнозування Direction для першого спостереження"
```

3.4 В циклі будуємо модель логістичної регресії, яка передбачає Direction за допомогою змінних Lag1 та Lag2, використовуючи всі спостереження, крім i-ого. Обчислено апостеріорну ймовірність для i-го спостереження для прогнозування, чи рухатиметься ринок вгору чи ні. А також визначено, чи допущена помилка при прогнозуванні Direction для i-го спостереження.

```
error_list = rep(0, length(Direction))
```

```
for (i in 1:length(Direction)) {
    fit.glm = glm(Direction ~ Lag1 + Lag2, data = Weekly[-
i, ],  family = "binomial")

    if (predict.glm(fit.glm, Weekly[i, ], type = "response") > 0.5) {
      if (Direction[i] == "Down") {
        error_list[i] = 1
      }
    }
}
```

3.5 Обчислено середнє з n чисел, отриманих у пункті 3.4, для того, щоб отримати оцінку LOOCV для тестової помилки.

```
print(paste("LOOCV оцінка для тестової помилки: ",
  round(mean(error_list) * 100, 2), "%"))
```

```
"LOOCV оцінка для тестової помилки:  41.32 %"
```

В результаті бачимо, що оцінка для тестової помилки становить 41.32%, що є досить поганим результатом.

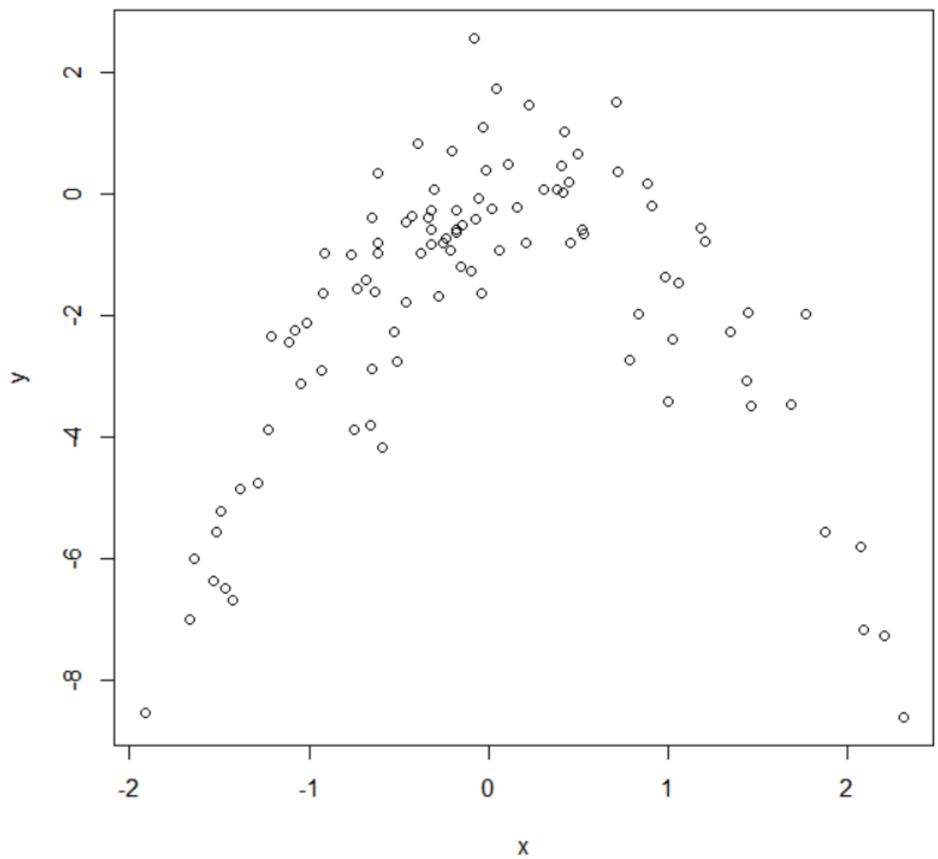
4. Використання перехресної перевірки на змодельованому наборі даних.

4.1 Створив змодельований набір наступним чином:

```
set.seed(1)
y = rnorm(100)
x = rnorm(100)
y = x - 2 * x^2 + rnorm(100)
```

В даному випадку бачимо, що аргументом для функції rnorm слугує число 100, що і є нашим n та свідчить про розмір змодельованої вибірки. Значення p=2 (оскільки в нас є 2 предиктора x та x^2). Для генерування даних використовується така модель: $Y=X-2X^2+\varepsilon$.

4.2 Побудовано діаграму розсіювання X vs Y.



З рисунку видно, що є наявна від'ємна квадратична залежність між змінними X та Y. Загалом графік нагадує обернену параболу.

4.3 Встановлено random.seed [set.seed(1)] та обчислено оцінки тестових помилок методом LOOCV для наступних чотирьох моделей:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

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

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

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \varepsilon$$

Взагалі варто зазначити, що всі обчислення були огорнуті у функцію з аргументом seed, для зручності виконання наступного завдання.

```
Coords = data.frame(x, y)

LOOCV = function(seed) {
    set.seed(seed)
    cat("\n")
    print(paste("seed --- ", seed))
      # 4.3.1
    fit.glm = glm(y ~ x)
    print(paste("LOOCV оцінка для тестової помилки [beta0 - beta1]: ",
    round(cv.glm(Coords, fit.glm)$delta[1], 2), "%"))

      # 4.3.2
    fit.glm2 = glm(y ~ x + I(x^2))
    print(paste("LOOCV оцінка для тестової помилки [beta0 - beta2]: ",
    round(cv.glm(Coords, fit.glm2)$delta[1], 2), "%"))

      # 4.3.3
    fit.glm3 = glm(y ~ poly(x, 3))
    print(paste("LOOCV оцінка для тестової помилки [beta0 - beta3]: ",
    round(cv.glm(Coords, fit.glm3)$delta[1], 2), "%"))

      # 4.3.4
    fit.glm4 = glm(y ~ poly(x, 4))
    print(paste("LOOCV оцінка для тестової помилки [beta0 - beta4]: ",
    round(cv.glm(Coords, fit.glm4)$delta[1], 2), "%"))
}

LOOCV(1)
```

```
[1] "seed ---  1"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta1]:  5.89 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta2]:  1.09 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta3]:  1.1 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta4]:  1.11 %"
```

З огляду на наведені тестові помилки видно, що квадратична модель показує найкращі результати (помилка 1.09%).

4.4 Повторено пункт 4.3 з використанням різних seed значень, так, наприклад, я взяв seed, що дорівнює 10 та 100.

```
[1] "seed ---  10"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta1]:  5.89 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta2]:  1.09 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta3]:  1.1 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta4]:  1.11 %"

[1] "seed ---  100"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta1]:  5.89 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta2]:  1.09 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta3]:  1.1 %"
[1] "LOOCV оцінка для тестової помилки [beta0 - beta4]:  1.11 %"
```

Бачимо, що було отримано такі ж результати як і в пункті 4.3. Це досить логічно, оскільки LOOCV оцінка не пов’язана з seed, бо вона бере по черзі абсолютно кожен елемент вибірки як валідаційний та рахує середню помилку серед всіх спроб.

4.5 Очікувано, що серед моделей з пункту 4.3 мала найменшу тестову помилку LOOCV модель з двома предикторами, а саме помилка склала 1.09%. Найгіршу точність показала лінійна модель. Це все виводиться з діаграми розсіювання, що показує квадратичну залежність між змінними X та Y.

4.6 Розглянемо статистичну значимість оцінок коефіцієнтів кожної з моделей розглянутих у пункті 4.3.

```
Call:
glm(formula = y ~ poly(x, 4))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8914  -0.5244   0.0749   0.5932   2.7796

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.8277     0.1041 -17.549   <2e-16 ***
poly(x, 4)1   2.3164     1.0415   2.224   0.0285 *
poly(x, 4)2 -21.0586     1.0415 -20.220   <2e-16 ***
poly(x, 4)3  -0.3048     1.0415  -0.293   0.7704
poly(x, 4)4  -0.4926     1.0415  -0.473   0.6373
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.084654)

    Null deviance: 552.21  on 99  degrees of freedom
Residual deviance: 103.04  on 95  degrees of freedom
AIC: 298.78

Number of Fisher Scoring iterations: 2
```

Після виводу p-value для моделі з 4 предикторами, бачимо, що статистично значущими є тільки перші два x та x^2. Це зрозуміло, оскільки ми використовували тільки перші два предиктора для генерування даних. Щодо узгодженості із результатами перехресної перевірки, то також бачимо, що найбільш значущим є квадратичний параметр (p-value < 0.001) і найменшу тестову помилку мала квадратична LOOCV модель.

5. Аналіз набору даних Boston з бібліотеки MASS.

```
      crim                zn             indus            chas
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
 1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
 Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
      nox               rm             age              dis
 Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
 1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
 Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
 Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
 3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
 Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
      rad              tax           ptratio          black
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
     lstat             medv
 Min.   : 1.73   Min.   : 5.00
 1st Qu.: 6.95   1st Qu.:17.02
 Median :11.36   Median :21.20
 Mean   :12.65   Mean   :22.53
 3rd Qu.:16.95   3rd Qu.:25.00
 Max.   :37.97   Max.   :50.00
```

Загальна характеристика даних Boston

5.1 Обчислено оцінку середнього для змінної medv.

```
medv_mean = mean(medv)
print(paste("Середнє для змінної medv: ", round(medv_mean, 2)))
```

```
"Середнє для змінної medv:  22.53"
```

Як бачимо середнє значення було оцінено як 22.53.

5.2 Обчислено стандартну похибку цієї оцінки використовуючи наступну формулу:

$$SE = \frac{\sigma \;\leftarrow \text{Standard deviation}}{\sqrt{n} \;\leftarrow \text{Number of samples}}$$

```
medv_se = sd(medv) / sqrt(length(medv))
print(paste("Стандартна похибка для змінної medv: ", round(medv_se, 2)))
```

```
[1] "Стандартна похибка для змінної medv:  0.41"
```

Як бачимо стандартна похибка була оцінена як 0.41.

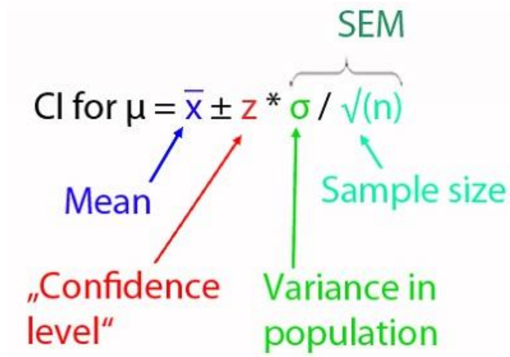
5.3 Оцінено стандартну похибку розглянутої вище оцінки середнього за допомогою бутстрапу.

```
set.seed(1)

boot.fn = function(data, index) {
    return (mean(data[index]))
}

print(boot(medv, boot.fn, 100))
```

```
Call:
boot(data = medv, statistic = boot.fn, R = 100)


Bootstrap Statistics :
    original       bias    std. error
t1* 22.53281 0.009027668   0.3482331
```

Як бачимо було взято R=100, тобто 100 повторень бутстрапу і у підсумку маємо стандартну похибку, що дорівнює 0.34, що таки на 0.06 нижче оціненої похибки у пункті 5.2.

5.4 На основі бутстрап оцінки побудовано 95% довіри для середнього значення змінної medv. Це було виконано за наступною формулою (в даному випадку для 95% довіри z=1.95):




```
ci = c(22.53 - 1.96 * 0.35, 22.53 + 1.96 * 0.35)
```

```
print(t.test(medv))
print(ci)
```

```
        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

[1] 21.844 23.216
```

Порівнюючи це з результатами отриманими за допомогою t.test(medv), бачимо, що результат є досить подібний, проте інтервал довіри на основі бутстрап оцінки є вужчим, тобто оцінка є більш точною.

5.5 На основі цього набору даних обчислено оцінку для медіани змінної medv.

```
medv_median = median(medv)
print(paste("Медіана для змінної medv: ", round(medv_median, 2)))
```

```
[1] "Медіана для змінної medv:  21.2"
```

Як бачимо медіану було оцінено як 21.2.

5.6 Оцінено стандартну помилку оцінки медіани змінної medv за допомогою бутстрапу.

```
boot.fn2 = function(data, index) {
    return (median(data[index]))
}

print(boot(medv, boot.fn2, 100))
```

```
Call:
boot(data = medv, statistic = boot.fn2, R = 100)


Bootstrap Statistics :
    original  bias    std. error
t1*     21.2   0.074   0.3602384
```

Було взято 100 повторень бутстрапу і у підсумку маємо стандартну похибку, що дорівнює 0.36.

5.7 На основі цього набору даних обчислено оцінку десятого процентиля змінної medv.

```
percentile = quantile(medv, c(0.1))
print(percentile)
```

```
  10%
12.75
```

5.8 Використано бутстрап, щоб оцінити стандартну похибку десятого процентиля змінної medv.

```
boot.fn3 = function(data, index) {
    return (quantile(data[index], c(0.1)))
}

print(boot(medv, boot.fn3, 100))
```

```
Call:
boot(data = medv, statistic = boot.fn3, R = 100)


Bootstrap Statistics :
    original  bias    std. error
t1*    12.75 -0.031   0.5302363
```

Результати оцінки показують, що значення десятого процентиля є ідентичні з результатом у пункті 5.7, стандартна похибка є досить високою і дорівнює 0.53. Проте це нормально з огляду на те, що ми працюємо з десятим процентилем (значення 10-ї частини сукупності менше або рівне нашого квантиля).