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

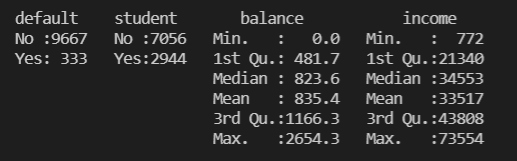
Перевірив:

Проф. Заболоцький Т. М.

Львів – 2021

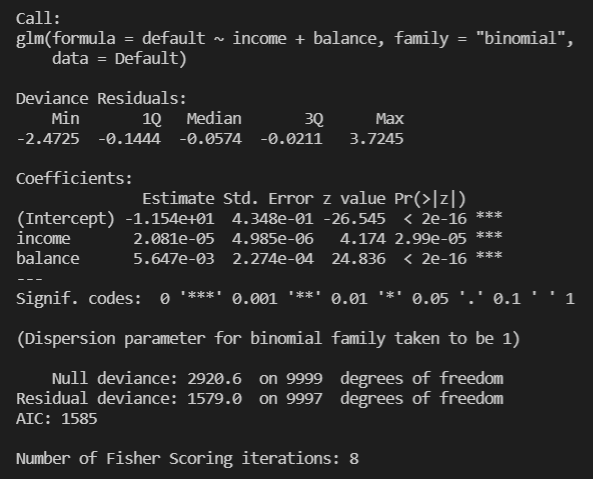
**Хід виконання**

**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.**



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.

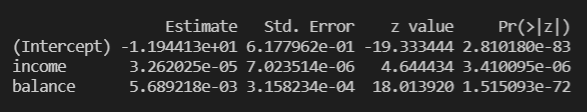
****

**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.

****

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, "%"))

****

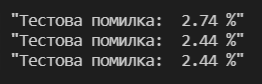
**1.3** Повторив попереднє завдання три рази, використовуючи три різні розбиття вибірки на навчальний та тестовий набори. Це було виконано взявши код до попереднього завдання у функцію, яку було викликано у циклі. Булівський аргумент передано для того, щоб не викликати кожен раз опис регресійної моделі.

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.

****

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

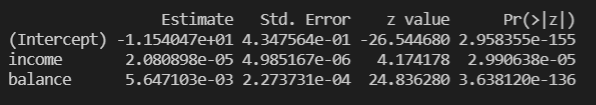
fit.glm = glm(default ~ income + balance + student,

 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.

****

В результаті маємо дані з колонки Std.Error (середньоквадратине відхилення) для коефіцієнтів , та відповідно , та .

**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,

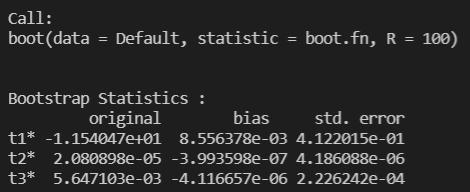
     family = "binomial", subset = index)

    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))

****

Як бачимо було взято R=100, тобто 100 повторень бутстрапу і у підсумку маємо наступні значення середньоквадратичного відхилення для коефіцієнтів , та відповідно , та .

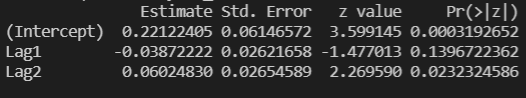
Тобто, оцінки виявилися досить схожими, однак бутстрап оцінив менші середньоквадратичні відхилення для параметрів нашої моделі.

**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.



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

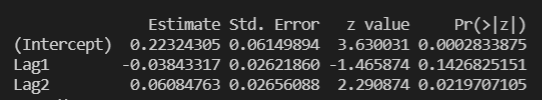
****

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.

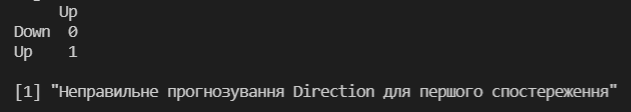
fit.glm2 = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = "binomial")

print(summary(fit.glm2)$coef)

****

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.

****

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") }

}

****

**3.4** В циклібудуємомодель логістичної регресії, яка передбачає Direction за допомогою змінних Lag1 та Lag2, використовуючи всі спостереження, крім і-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 оцiнка для тестової помилки: ",

  round(mean(error\_list) \* 100, 2), "%"))

****

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)

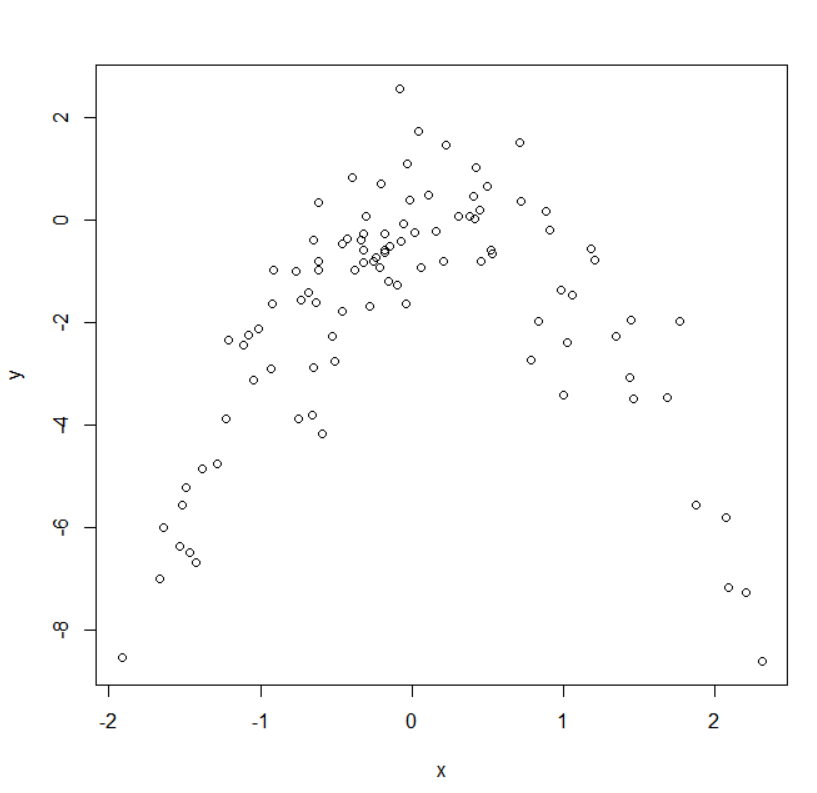
y = rnorm(100)

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 ). Для генерування даних використовується така модель: Y=X−2+ε.

**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 = β0 + β1X + ε

Y = β0 + β1X + β2X2 + ε

Y = β0 + β1X + β2X2 + β3X3 + ε

Y = β0 + β1X + β2X2 + β3X3 + β4X4 + ε

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 оцiнка 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 оцiнка 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 оцiнка для тестової помилки [beta0 - beta3]: ",

    round(cv.glm(Coords, fit.glm3)$delta[1], 2), "%"))

      # 4.3.4

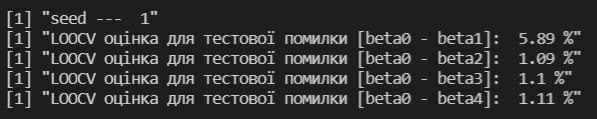
    fit.glm4 = glm(y ~ poly(x, 4))

    print(paste("LOOCV оцiнка для тестової помилки [beta0 - beta4]: ",

    round(cv.glm(Coords, fit.glm4)$delta[1], 2), "%"))

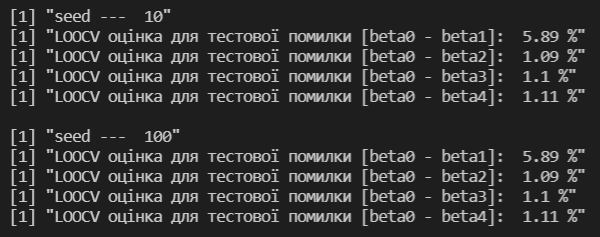
}

LOOCV(1)

****

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

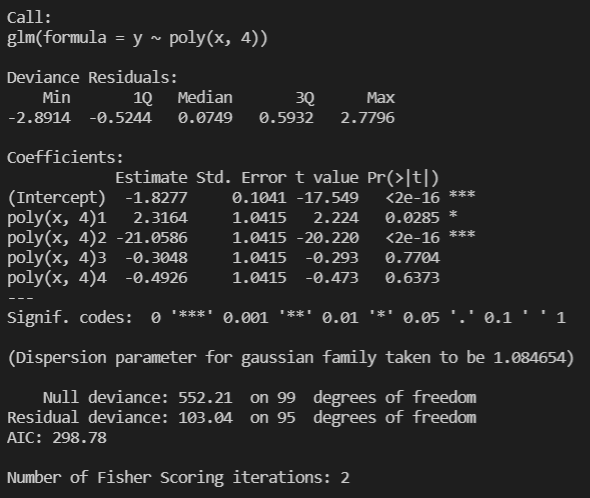
**4.4** Повторено пункт 4.3 з використанням різних seed значень, так, наприклад, я взяв seed, що дорівнює 10 та 100.

****

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.

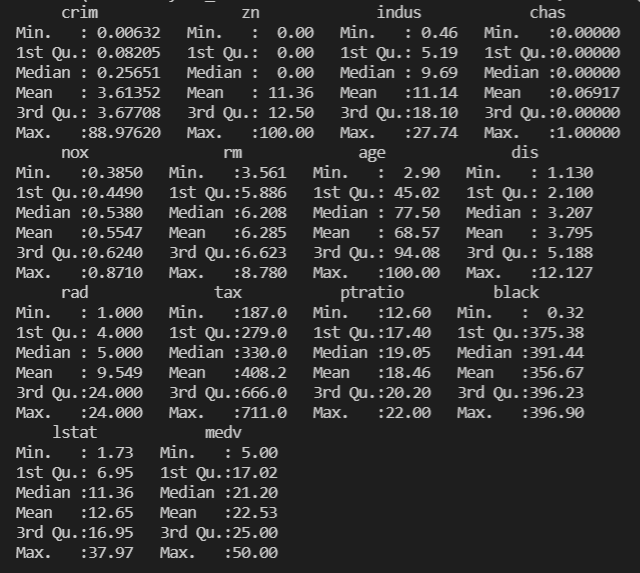
**4.5** Очікувано, що серед моделей з пункту 4.3 мала найменшу тестову помилку LOOCV модель з двома предикторами, а саме помилка склала 1.09%. Найгіршу точність показала лінійна модель. Це все виводиться з діаграми розсіювання, що показує квадратичну залежність між змінними X та Y.

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

****

Після виводу p-value для моделі з 4 предикторами, бачимо, що статистично значущими є тільки перші два x та . 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.**

****

Загальна характеристика даних Boston

**5.1** Обчислено оцінку середнього для змінної medv.

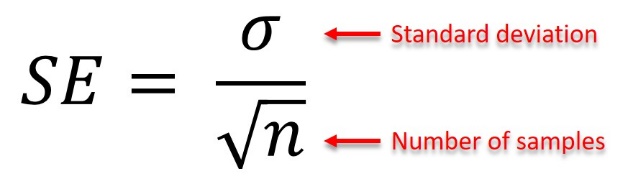
medv\_mean = mean(medv)

print(paste("Середнє для змiнної medv: ", round(medv\_mean, 2)))

****

Як бачимо середнє значення було оцінено як 22.53.

**5.2** Обчислено стандартну похибку цієї оцінки використовуючи наступну формулу:



medv\_se = sd(medv) / sqrt(length(medv))

print(paste("Стандартна похибка для змiнної medv: ", round(medv\_se, 2)))

****

Як бачимо стандартна похибка була оцінена як 0.41.

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

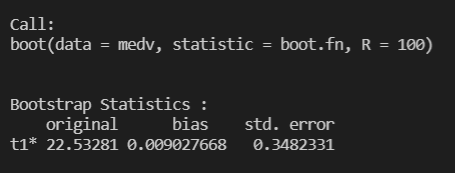
set.seed(1)

boot.fn = function(data, index) {

    return (mean(data[index]))

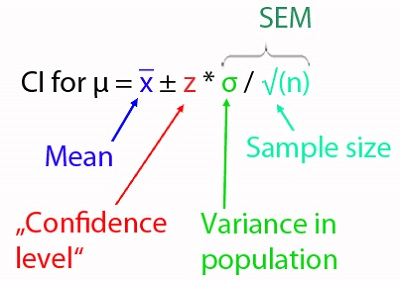
}

print(boot(medv, boot.fn, 100))

****

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.

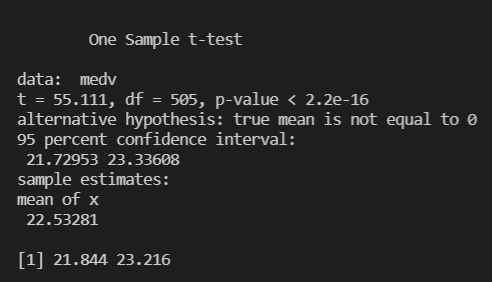
**5.4** На основі бутстрап оцінки побудовано 95% довіри для середнього значення змінної medv. Це було виконано за наступною формулою (в даному випадку для 95% довіри z=1.95):



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

print(t.test(medv))

print(ci)

****

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("Медiана для змiнної medv: ", round(medv\_median, 2)))

****

Як бачимо медіану було оцінено як 21.2.

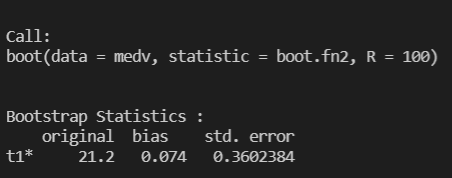
**5.6** Оцінено стандартну помилку оцінки медіани змінної medv за допомогою бутстрапу.

boot.fn2 = function(data, index) {

    return (median(data[index]))

}

print(boot(medv, boot.fn2, 100))

****

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)

****

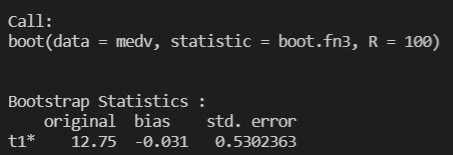
**5.8** Використано бутстрап, щоб оцінити стандартну похибку десятого процентиля змінної medv.

boot.fn3 = function(data, index) {

    return (quantile(data[index], c(0.1)))

}

print(boot(medv, boot.fn3, 100))

****

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).