МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

ЗВІТ

до індивідуального завдання №5

з дисципліни «Моделі статистичного навчання»

Виконав

студент групи ПМіМ-12:

Бордун Михайло

Перевірив:

Проф. Заболоцький Т. М.

Львів – 2021

**Хід виконання**

**1.** На основі згенерованих даних застосуємо вибір найкращої підмножини.

**1.1** Використовуючи функцію rnorm() згенеруйте предиктор *X* довжиною *n* = 100, та вектор залишків *ε* такої ж довжини *n* = 100.

set.seed(1)

x = rnorm(100)

eps = rnorm(100)

**1.2** Згенеруйте вектор залежних змінних *Y* довжини *n* = 100 відповідно до моделі

*Y* = *β*0 + *β*1*X* + *β*2*X*2 + *β*3*X*3 + *ε*,

де *β*0, *β*1, *β*2 і *β*3 константи на ваш вибір.

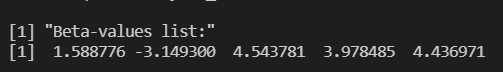
betas = runif(5, min=-5, max=5)

print("Beta-values list:")

print(betas)

y = betas[1] + betas[2] \* x +

 betas[3] \* x^2 + betas[4] \* x^3 + eps

****

Визначення значень *β*i відбувається випадково (в минулому пункті визначено seed як 1), в межах від -5 до 5 будь-які дробові числа. Останнє значення – це значення *β*7, яке нам буде необхідне в наступних пунктах.

**1.3** Використовуючи функцію regsubsets() виберіть найкращу модель методом вибору найкращої підмножини з множини предикторів *X*, *X*2, . . ., *X*10. Яка модель найкраща за показниками *Cp*, *BIC* і скорегований *R*2? Наведіть декілька графіків на підтвердження своєї відповіді та вкажіть оцінки коефіцієнтів найкращої моделі.

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

BestModelSelection = function(method\_, mtext\_) {

    cat("\n")

    reg.fit = regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4)

    + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10), data = data.frame,

     nvmax = 10, method = method\_)

    reg.summary = summary(reg.fit)

    print(reg.summary)

    par(mfrow = c(2, 2))

    plot(reg.summary$cp, xlab = "Кiлькість змiнних", ylab = "Cp", type = "l")

    points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)],

    col = "blue", cex = 2, pch = 20)

    plot(reg.summary$bic, xlab = "Кiлькість змiнних", ylab = "BIC", type = "l")

    points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)],

    col = "blue", cex = 2, pch = 20)

    plot(reg.summary$adjr2, xlab = "Кiлькість змiнних", ylab = "Скорегований R^2", type = "l")

    points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)],

    col = "blue", cex = 2, pch = 20)

    mtext(mtext\_, side = 3, line = -2, outer = TRUE)

    cat("\n")

    point.a = which.min(reg.summary$cp)

    point.b = which.min(reg.summary$bic)

    point.c = which.max(reg.summary$adjr2)

    if (point.a == point.b && point.a == point.c) {

      print(coef(reg.fit, point.a))

    } else if (point.a == point.b && point.a != point.c) {

      print(coef(reg.fit, point.b))

      print(coef(reg.fit, point.c))

    } else if (point.a != point.b && point.a == point.c ||

              point.a != point.b && point.b == point.c) {

      print(coef(reg.fit, point.a))

      print(coef(reg.fit, point.b))

    } else if (point.a != point.b && point.b == point.c) {

    } else {

      print(coef(reg.fit, point.a))

      print(coef(reg.fit, point.b))

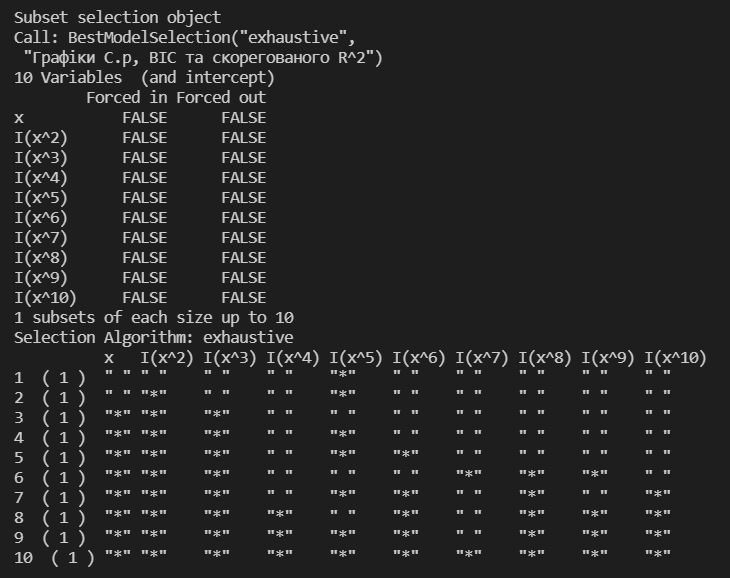
      print(coef(reg.fit, point.c))

    }

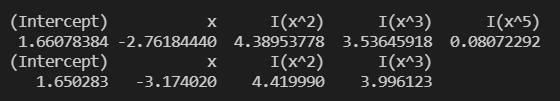
}

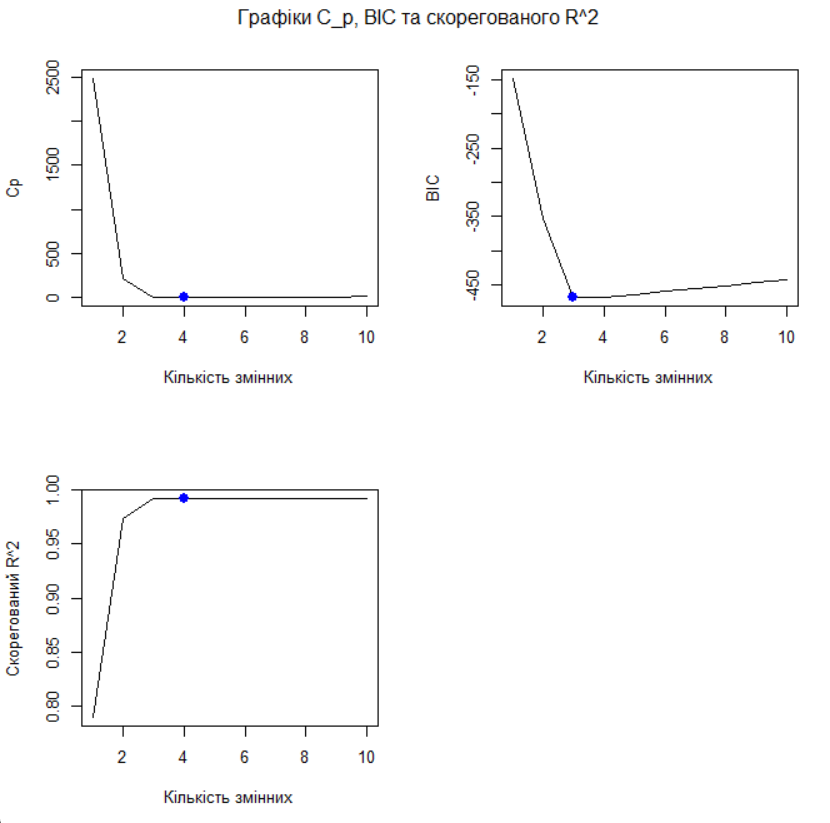
BestModelSelection("exhaustive",

 "Графiки C\_p, BIC та скорегованого R^2")

****

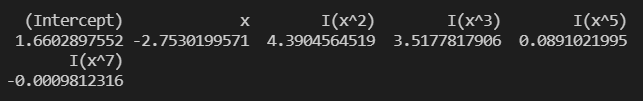
Як бачимо тут подано найкращий набір змінних для кожної розмірності моделі. Зірочки тут означають, що дана змінна включена у відповідну модель.

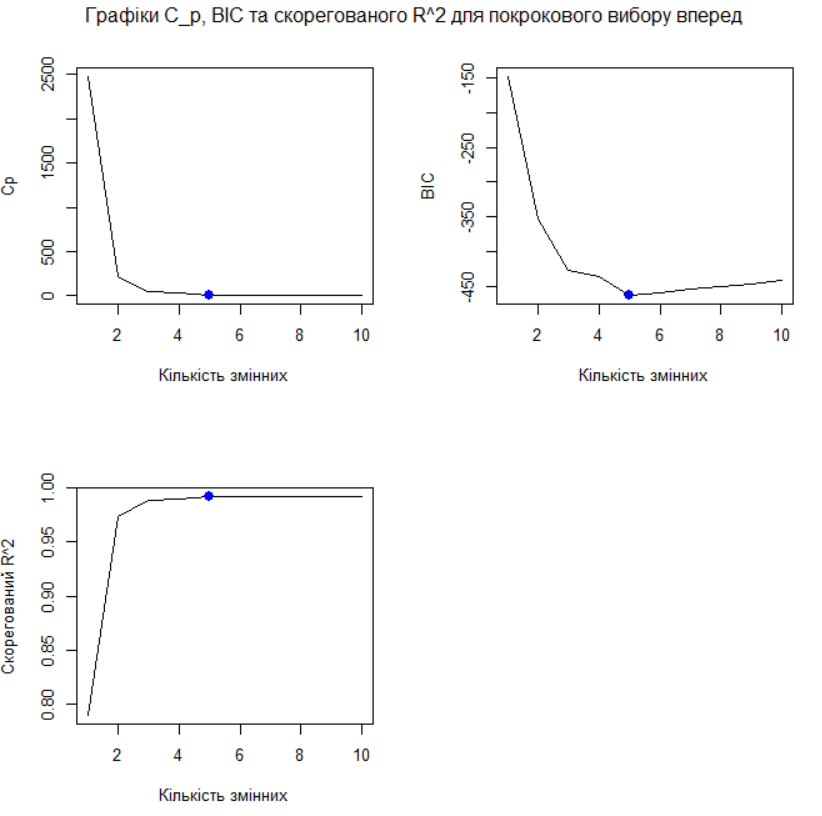
****

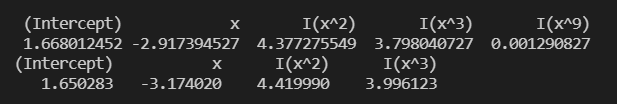
****

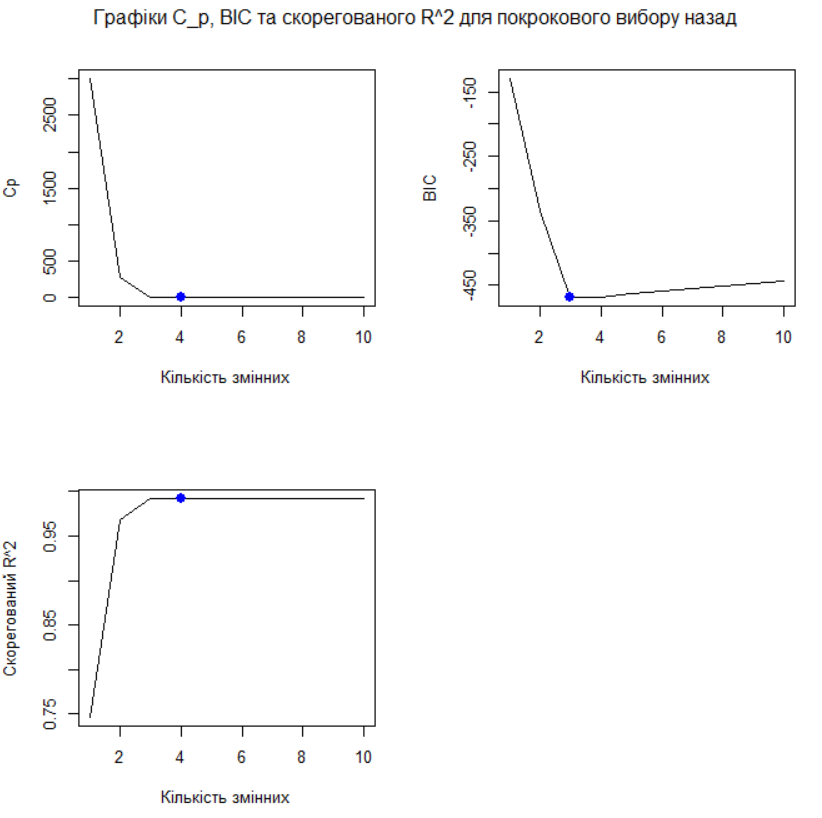
З огляду на наведені вище результати, бачимо, що найкраща модель за показниками Cp та скорегованим R2 – це модель з 4 змінними (x, x2, x3, x5). Для показника BIC бачимо, що вже найкращою буде модель зі змінними x, x2 та x3.

**1.4** Повторіть 1.3, використовуючи методи покрокового вибору вперед та назад. Порівняйте отримані результати з 1.3.

****

****

****

****

Аналізуючи наведені результати, можна сказати, що кожен метод дає інший результат, і тільки для методу покрокового вибору вперед для всіх показників найкращою буде модель з 5-ма змінними (x, x2, x3, x5, x7).

Для методу покрокового вибору назад результат дуже подібних до методу вибору найкращої підмножини, оскільки для кожного показника найкраща модель має таку ж кількість змінних, але вони відрізняються: для покрокового вибору назад найкраща модель за показниками Cp та скорегованим R2 є зі змінними x, x2, x3, x9. Для показника BIC бачимо, що вже найкращою буде модель зі змінними x, x2 та x3.

**1.5** Пристосуйте ласо модель до згенерованих даних, використовуючи *X*, *X*2, . . ., *X*10 як предиктори . Використайте перехресну перевірку для вибору значення λ. Побудуйте графіки помилки перехресної перевірки як функції від λ. Наведіть отримані оцінки коефіцієнтів моделі та обгрунтуйте отримані результати.

Lasso = function() {

    cat("\n")

    xmat = model.matrix(y ~ x + I(x^2) + I(x^3) + I(x^4)

        + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10), data = data.frame)[, -1]

    cv.out = cv.glmnet(xmat, y, alpha = 1)

    par(mfrow = c(1, 1))

    bestlam = cv.out$lambda.min

    print(paste('Min lambda: ', bestlam))

    plot(cv.out)

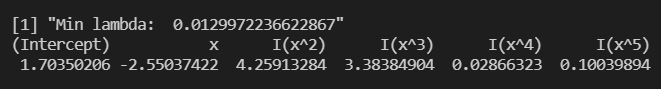
    out = glmnet(xmat, y, alpha = 1)

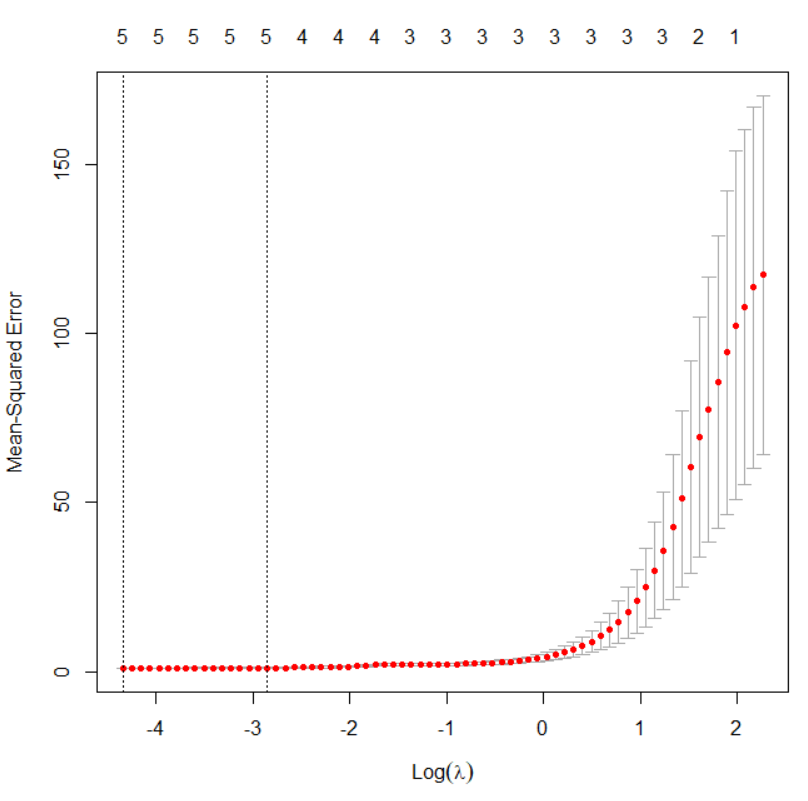
    lasso.coefs = predict(out, s = bestlam, type = "coefficients")[1:11, ]

    print(lasso.coefs[lasso.coefs != 0])

}

Lasso()

****

****

З рисунка видно, що при додатних значеннях логарифма лямбли помилка стрімко зростає. В результаті бачимо, що методом лассо ми знайшли мінімальну в сенсі помилки лямбду, яка дорівнює 0.013 і з допомогою неї знайшли найкращу модель, яка включає в себе 5 змінних (x, x2, x3, x4, x5).

**1.6**  Згенеруйте вектор залежних змінних *Y* відповідно до моделі

*Y* = *β*0 + *β*7*X*7 + *ε*,

і застосуйте метод найкращого вибору підмножини і ласо. Обгрунтуйте отримані результати.

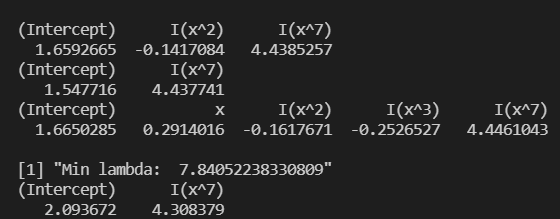
y = betas[1] + betas[5] \* x^7 + eps

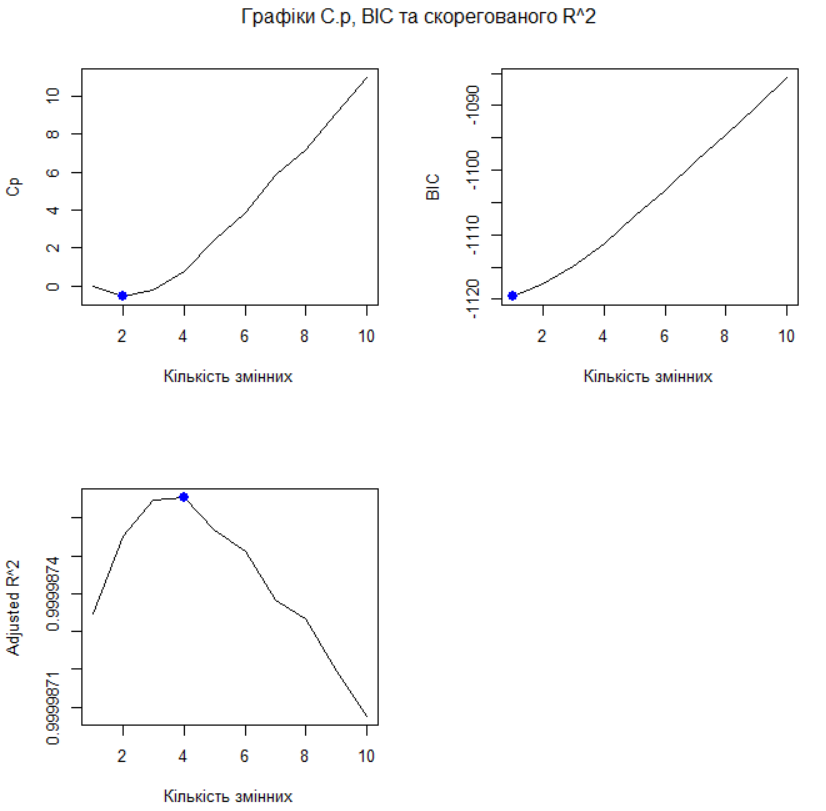
data.frame = data.frame(y = y, x = x)

BestModelSelection("exhaustive",

 "Графiки C.p, BIC та скорегованого R^2")

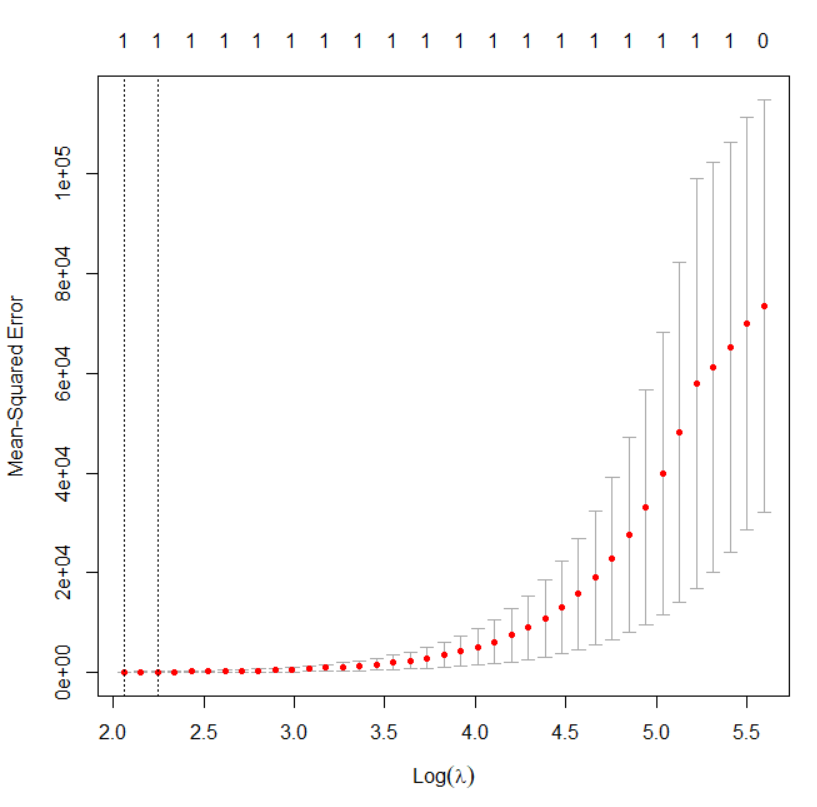
Lasso()

****

****

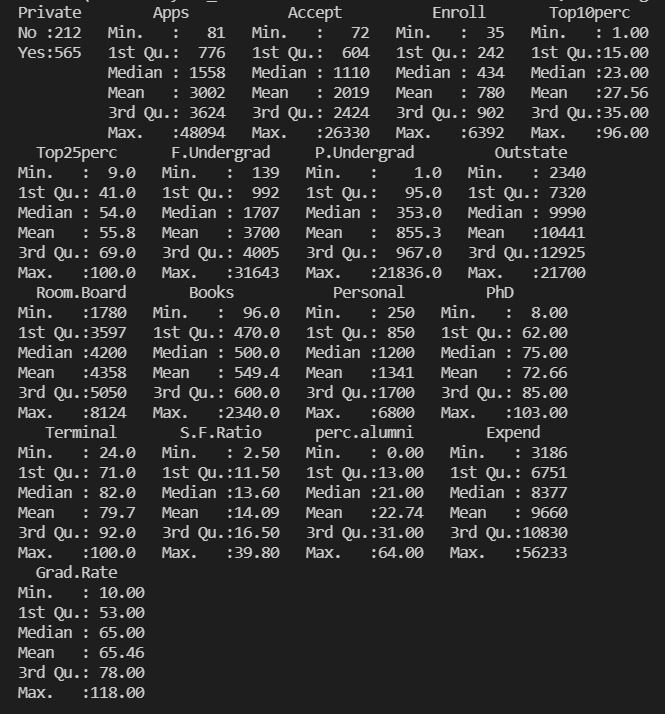
Враховуючи наведені вище результати, бачимо, що найкраща модель для всіх показників має різну кількість змінних, так: за показниками Cp  – це модель з 2 змінними (x2, x7). скорегованим R2 модель з 4 змінними (x, x2, x3, x7). Для показника BIC бачимо, що вже найкращою буде модель зі змінною x7.

Оскільки β7 = 4.437, то можемо з впевненістю сказати, що метод найкращого вибору підмножини з використанням BIC визначає найбільш точну модель з однією змінною.

****

Щодо методу лассо, то ми знайшли мінімальну в сенсі помилки лямбду, яка дорівнює 7.84 і з допомогою неї знайшли найкращу модель, яка включає одну змінну x7. Проте точність моделі є гіршою у порівнянні з методом найкращого вибору підмножини з використанням показника BIC.

**2.** На основі даних College передбачимо кількість отриманих заяв.

****

Загальна характеристика даних College

**2.1** Розбийте набір даних на навчальний та тестовий набори.

set.seed(1)

train = sample(1:length(Apps), 0.5 \* length(Apps))

College.train = College[train, ]

College.test = College[-train, ]

**2.2** Оцініть лінійну модель, використовуючи метод найменших квадратів на навчальному наборі, та обчисліть тестову помилку.

fit.lm = lm(Apps ~ ., data = College.train)

pred.lm = predict(fit.lm, College.test)

cat("\n")

print(paste("Test error: ", round(mean((pred.lm - College.test$Apps)^2), 2)))

****

**2.3** Пристосуйте модель гребеневої регресії до тренувального набору, вибравши λ шляхом перехресної перевірки. Обчисліть тестову помилку.

train.mat = model.matrix(Apps ~ ., data = College.train)

test.mat = model.matrix(Apps ~ ., data = College.test)

grid = 10 ^ seq(10, -2, length = 100)

fit.ridge = glmnet(train.mat, College.train$Apps, alpha = 0,

 lambda = grid)

cv.ridge = cv.glmnet(train.mat, College.train$Apps, alpha = 0,

 lambda = grid)

cat("\n")

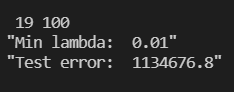
print(dim(coef(fit.ridge)))

bestlam = cv.ridge$lambda.min

print(paste('Min lambda: ', bestlam))

pred.ridge = predict(fit.ridge, s = bestlam, newx = test.mat)

print(paste("Test error: ", round(mean((pred.ridge - College.test$Apps)^2), 2)))

****

З кожним вектором лямбда пов’язаний вектор коефіцієнтів гребеневої регресії, які зберігаються в матриці, до якої можна отримати доступ через coef(). У цьому випадку це 19x100 матриця (лямбди з огляду на grid() є в межах від до ).

Мінімальна в сенсі помилки лямбда дорівнює 0.01 і з допомогою неї знайшли найкращу тестову помилку, що дорівнює 1134676.8, що є кращим результатом в порівнянні з обчисленою методом найменших квадратів.

**2.4** Пристосуйте модель ласо до тренувального набору, вибравши λ шляхом перехресної перевірки. Обчисліть тестову помилку. Яка кількість ненульових оцінок коефіцієнтів.

fit.lasso = glmnet(train.mat, College.train$Apps, alpha = 1,

 lambda = grid)

cv.lasso  = cv.glmnet(train.mat, College.train$Apps, alpha = 1,

 lambda = grid)

cat("\n")

bestlam = cv.lasso$lambda.min

print(paste('Min lambda: ', bestlam))

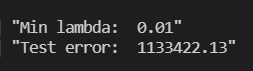
pred.lasso = predict(fit.lasso, s = bestlam, newx = test.mat)

print(paste("Test error: ", round(mean((pred.lasso - College.test$Apps)^2), 2)))

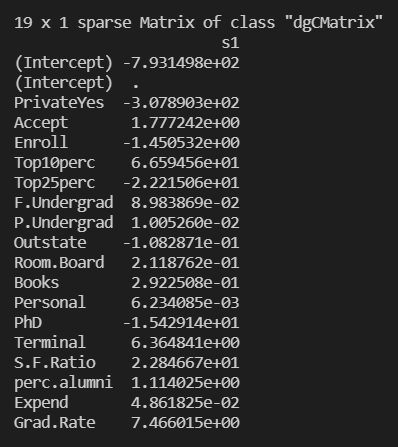
cat("\n")

lasso.coefs = predict(fit.lasso, s = bestlam, type = "coefficients")

print(lasso.coefs)

****

Мінімальна в сенсі помилки лямбда дорівнює 0.01 і з допомогою неї знайшли найкращу тестову помилку, що дорівнює 1133422.1, що є кращим результатом в порівнянні з обчисленою методом найменших квадратів та гребеневої регресії.

****

Також бачимо, що жоден з коефіцієнтів не дорівнює нулю.

**2.5** Пристосуйте модель PCR до тренувального набору, причому М виберіть шляхом перехресної перевірки. Яке отримане значення М? Обчисліть отриману помилку тесту.

fit.pcr = pcr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")

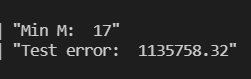
validationplot(fit.pcr, val.type = "MSEP", xlab = "Кількість змінних")

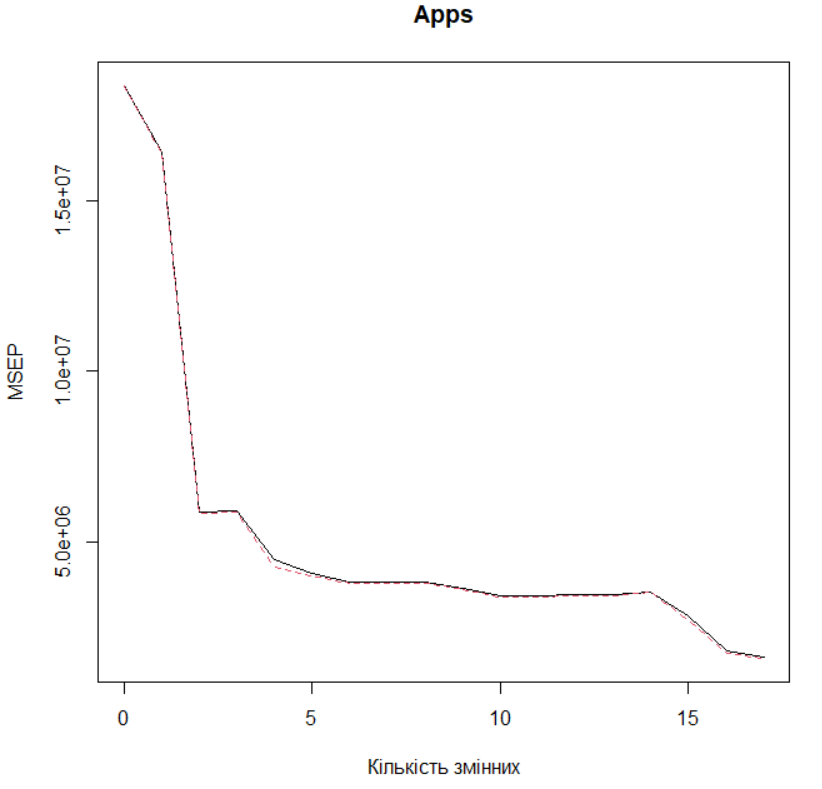
cat("\n")

print(paste('Min M: ', which.min(fit.pcr$validation$adj)))

pred.pcr = predict(fit.pcr, College.test, ncomp = which.min(fit.pcr$validation$adj))

print(paste("Test error: ", round(mean((pred.pcr - College.test$Apps)^2), 2)))

****

****

Мінімальна в сенсі помилки модель має M = 17 з тестовою помилкою, що дорівнює 1135758.3, що є ідентичним результатом в порівнянні з обчисленою методом найменших квадратів.

**2.6** Пристосуйте модель PLS до тренувального набору, причому М виберіть шляхом перехресної перевірки. Яке отримане значення М? Обчисліть отриману помилку тесту.

fit.pls = plsr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")

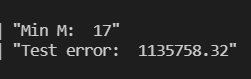
validationplot(fit.pls, val.type = "MSEP", xlab = "Кількість змінних")

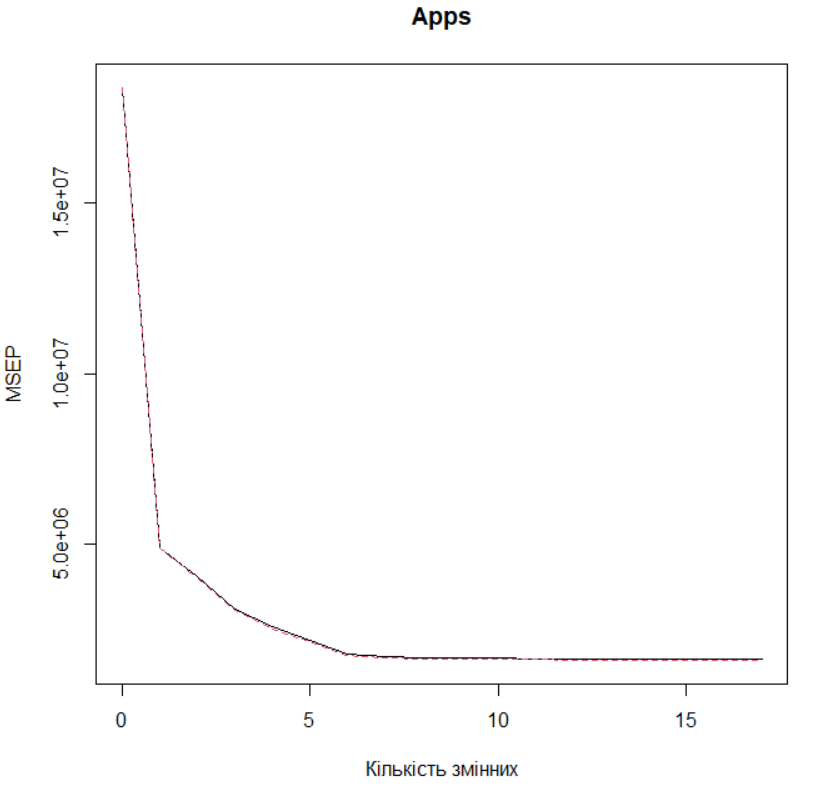
cat("\n")

print(paste('Min M: ', which.min(fit.pls$validation$adj)))

pred.pls = predict(fit.pls, College.test, ncomp = which.min(fit.pls$validation$adj))

print(paste("Test error: ", round(mean((pred.pls - College.test$Apps)^2), 2)))

****

****

Мінімальна в сенсі помилки модель має M = 17 з тестовою помилкою, що дорівнює 1135758.3, що є ідентичним результатом в порівнянні з обчисленою методом найменших квадратів та помилкою у моделі PCR.

**2.7** Прокоментуйте отримані результати. Наскільки точно ми можемо передбачити кількість отриманих заявок на коледж? Чи велика різниця між тестовими помилками, що виникають внаслідок розглянутих п’яти підходів?

test.mean = mean(College.test$Apps)

methods.list = c(pred.lm, pred.ridge, pred.lasso, pred.pcr, pred.pls)

GetRSqr = function(m) {

  r\_result = 1 - mean((m - College.test$Apps)^2) / mean((test.mean - College.test$Apps)^2)

  return(round(r\_result, 7)\*100)

}

cat("\n")

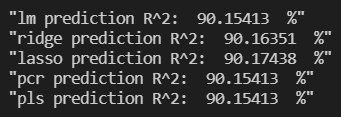
print(paste("lm prediction R^2: ", GetRSqr(pred.lm), " %"))

print(paste("ridge prediction R^2: ", GetRSqr(pred.ridge), " %"))

print(paste("lasso prediction R^2: ", GetRSqr(pred.lasso), " %"))

print(paste("pcr prediction R^2: ", GetRSqr(pred.pcr), " %"))

print(paste("pls prediction R^2: ", GetRSqr(pred.pls), " %"))

****

Для перевірки точності передбаченя кількості отриманих заявок на коледж було виконано обчислення коефіцієнту детермінації, який і показує наскільки отримані спостереження підтверджують модель.

Як бачимо є найближчим до ідеального (90.1744 %) для моделі лассо і найнижчий для моделей PCR, PLS та лінійної (90.1541 %).

Щодо різниця між тестовими помилками, то вона є досить мала у розглянутих 5-ти підходах, а саме різниця між найбільшою помилкою та найменшою складає близько 0.2%.

**3.** Ми бачили, що зі збільшенням кількості предикторів, що використовуються в моделі, навчальна помилка обов'язково зменшиться, але тестова - не обов'язково. Дослідимо це на згенерованих даних.

**3.1** Сформуйте набір даних з *p* = 20 ознаками, *n* = 1000 спостереженнями, і пов'язаний з ним вектор залежних змінних відповідно до моделі

*Y* = *Xβ* + *ε*,

де вектор *β* має деякі елементи, які точно дорівнюють нулю.

set.seed(1)

x = matrix(rnorm(1000 \* 20), 1000, 20)

eps = rnorm(1000)

cat("\n")

betas = runif(20, min=-5, max=5)

print("Beta-values list:")

zero\_vals\_num = sample(3:10, 1)

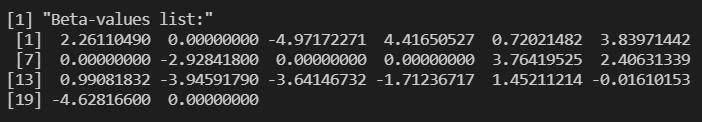
for (i in sample(1:length(betas), zero\_vals\_num)) {

  betas[i] = 0

}

print(betas)

y = x %\*% betas + eps

****

Визначення значень *β*i відбувається випадково (seed дорівнює 1), в межах від -5 до 5 будь-які дробові числа. Далі також випадково береться число від 3 до 10, що позначає кількість значень вектора *β*, які точно дорівнюють нулю, і визначається також випадково які конкретно значення будуть занулюватись. Результат виводу вектора можна бачити вище.

**3.2** Розділіть свій набір даних на навчальний набір, що містить 100 спостережень та тестовий набір, що містить 900 спостережень.

train = sample(1:length(eps), 100)

x.train = x[train, ]

y.train = y[train]

x.test = x[-train, ]

y.test = y[-train]

**3.3**  Використайте метод вибору найкращої підмножини на навчальному наборі та побудуйте графік навчального MSE, який відповідає найкращій моделі кожного розміру.

reg.fit = regsubsets(y ~ .,

                     data = data.frame(y = y.train, x = x.train),

                     nvmax = 20)

BestSubsetSelection = function(x\_, y\_, ylab\_) {

    data\_ = data.frame(y = y\_, x = x\_)

    mat = model.matrix(y ~ ., data = data\_, nvmax = 20)

    val.errors = rep(0, 20)

    for (i in 1:20) {

        coef\_i = coef(reg.fit, id = i)

        pred = mat[, names(coef\_i)] %\*% coef\_i

        val.errors[i] = mean((pred - y\_)^2)

    }

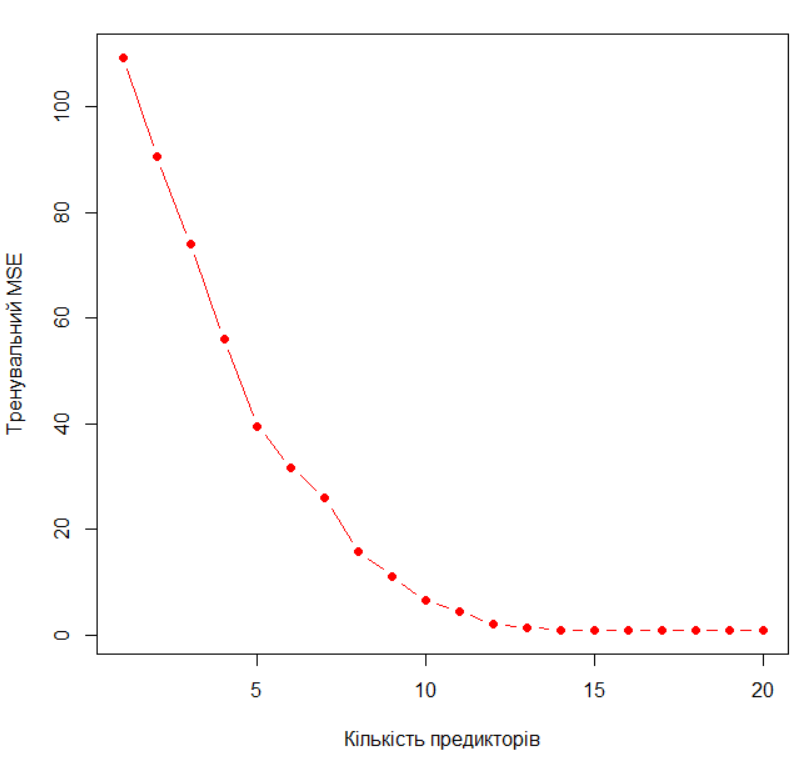
    plot(val.errors, xlab = "Кiлькiсть предикторiв", ylab = ylab\_,

    col = "red", pch = 19, type = "b")

    return(val.errors)

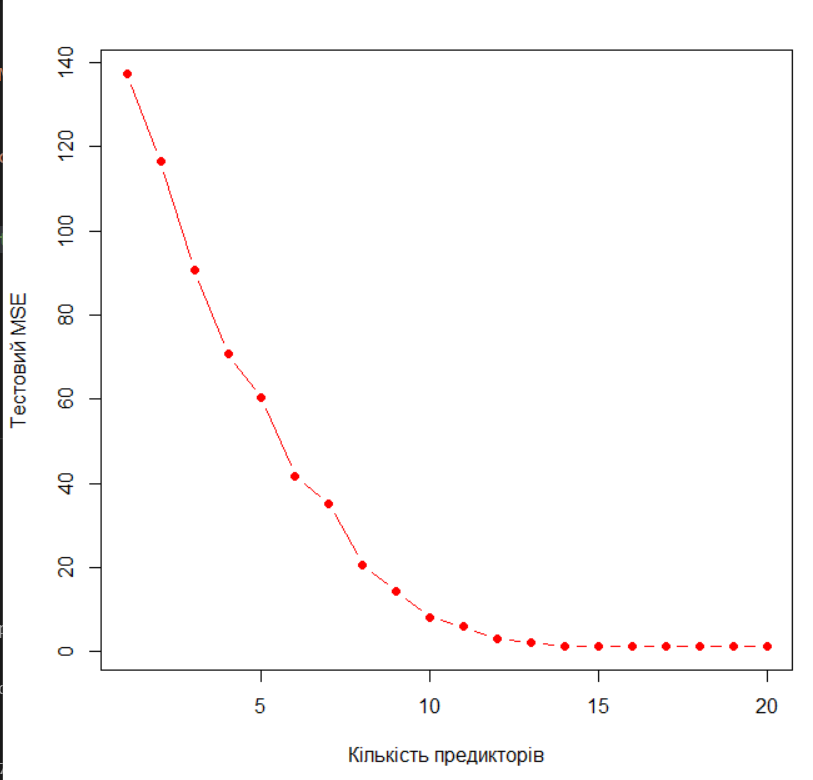
}

BestSubsetSelection(x.train, y.train, "Тренувальний MSE")

****

З графіка навчального MSE бачимо, що справді зі збільшенням кількості предикторів, що використовуються в моделі, навчальна помилка зменшується.

**3.4** Побудуйте графік тестового MSE, який відповідає найкращій моделі кожного розміру.

****

**3.5** Для якого розміру моделі тестовий MSE приймає мінімальне значення? Прокоментуйте отримані результати.

print(paste('Model size for min MSE: ', which.min(test\_mse),

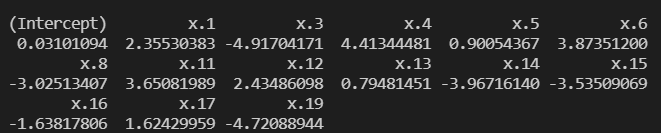
 ', min MSE: ', min(test\_mse)))

****

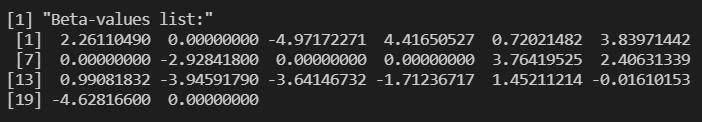
Хоч і з графіка досить складно визначити розмір моделі, де тестовий MSE приймає мінімальне значення, але це модель з 14-ма змінними, тобто модель не містить всі предиктори як, наприклад, це було з навчальною помилкою. Сам MSE в даному випадку дорівнює 1.1617.

**3.6** Як співвідносяться модель, що мінімізує тестовий MSE та справжня модель, яка використовувалася для генерації даних? Прокоментуйте значення оцінок коефіцієнтів.

print(coef(reg.fit, which.min(test\_mse)))

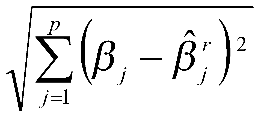
****

Наведу ще раз список коефіцієнтів для справжньої моделі, яка використовувалася для генерації даних.

****

Загалом бачимо, що модель, яка мінімізує тестовий MSE правильно визначила всі 5 нульових коефіцієнтів з списку бет і виключила їх з моделі. Щодо решти коефіцієнтів, то значень оцінок є точними до десятих.

**3.7** Побудуйте графік для відображення величини



для всіх значень *r*, де https://lh3.googleusercontent.com/hyxQ7S6Djzw4PPdiJRfetGiwk55UDeWvjP54c9MAtXeoNQ18ixkDUjgwS_L-zX61VdUl5o-9hrfvwX6LZmt04Xel1rlkuIy-FP27KRgOMhqQ9E78K7FvKQrmRr4vGDEyRfMj9po - оцінка *j*-ого коефіцієнта для найкращої моделі, що містить *r* коефіцієнтів. Прокоментуйте результати. Порівняйте отриманий графік з графіком тестового MSE з 3.4?

val.errors = rep(0, 20)

x\_cols = colnames(x, do.NULL = FALSE, prefix = "x.")

for (i in 1:20) {

    coef\_i = coef(reg.fit, id = i)

    val.errors[i] =

    sqrt(sum((betas[x\_cols %in% names(coef\_i)] -

         coef\_i[names(coef\_i) %in% x\_cols])^2))

}

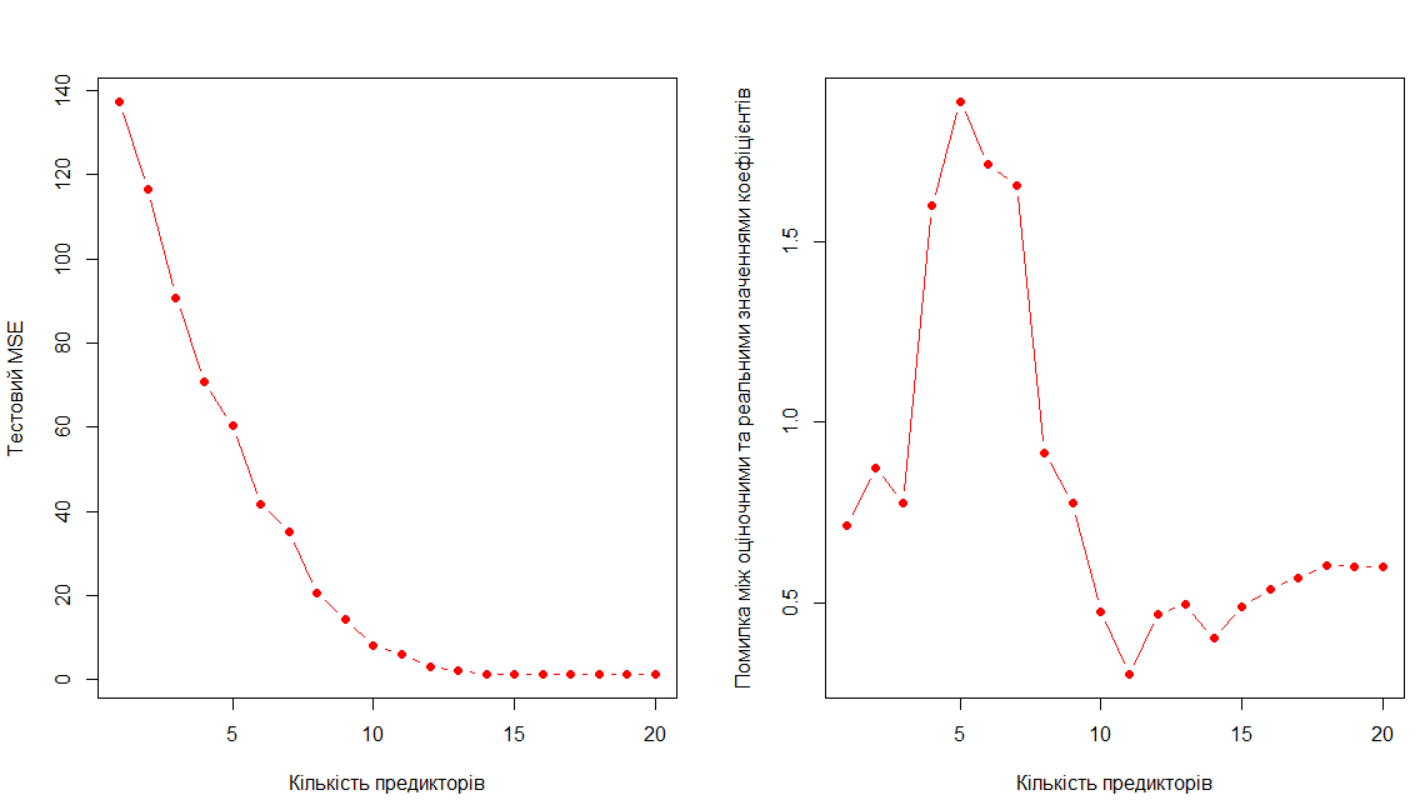
par(mfrow = c(1, 2))

BestSubsetSelection(x.test, y.test, "Тестовий MSE")

plot(val.errors, xlab = "Кiлькiсть предикторiв",

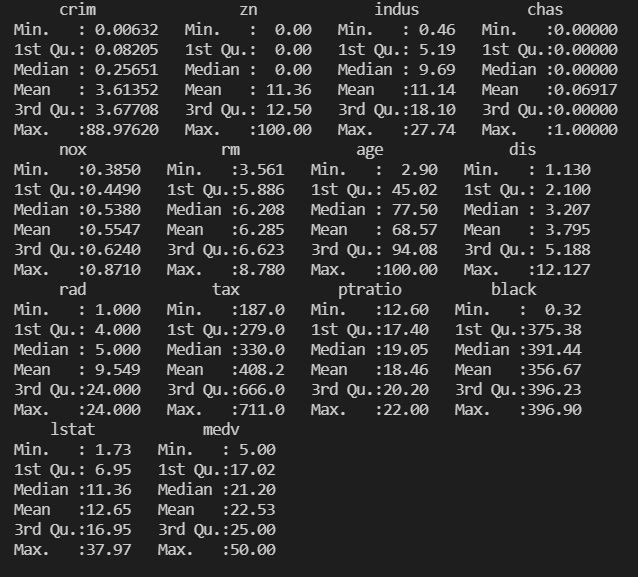
 ylab = "Помилка між оціночними та реальними значеннями коефіцієнтів",

 col = "red", pch = 19, type = "b")

****

В результаті бачимо, що модель з 11-ма предикторами мінімізує помилку між оціночними та справжніми значеннями коефіцієнтів. Як було сказано раніше модель з 14-ма змінними є найкращою для тестового MSE. Проте варто сказати, що модель з 14-ма змінними є найкращою після моделі з 11-ма для помилки між оціночними та справжніми значеннями коефіцієнтів, але якоїсь явної кореляції між двома методами немає.

**4.** Спробуємо передбачити рівень злочинності на основі набору даних Boston.

****

Загальна характеристика даних Boston

**4.1-4.3** Застосуйте методи вибору моделі регресії, розглянуті раніше, такі як вибір найкращої підмножини, ласо, гребенева регресія та PCR. Представте та обговоріть результати щодо підходів, які ви використовуєте.

Запропонуйте модель, яка мала б добре працювати і обґрунтуйте свою відповідь. Для оцінки якості моделі використайте помилки валідаційної множини.

Чи включає обрана модель всі предиктори? Чому?

predict.regsubsets = function(object, newdata, id, ...) {

    form = as.formula(object$call[[2]])

    mat = model.matrix(form, newdata)

    coef\_i = coef(object, id = id)

    xvars = names(coef\_i)

    mat[, xvars] %\*% coef\_i

}

k = 10

folds = sample(1:k, nrow(Boston), replace = TRUE)

cv.errors = matrix(0, k, 13)

for (j in 1:k) {

    best.fit = regsubsets(crim ~ ., data = Boston[folds != j, ], nvmax = 13)

    for (i in 1:13) {

        pred = predict.regsubsets(best.fit, Boston[folds == j, ], id = i)

        cv.errors[j, i] = mean((Boston$crim[folds == j] - pred)^2)

    }

}

mean.cv.errors = rep(0, 13)

for (i in 1:13) {

  mean.cv.errors[i] = mean(cv.errors[, i])

}

cat("\n")

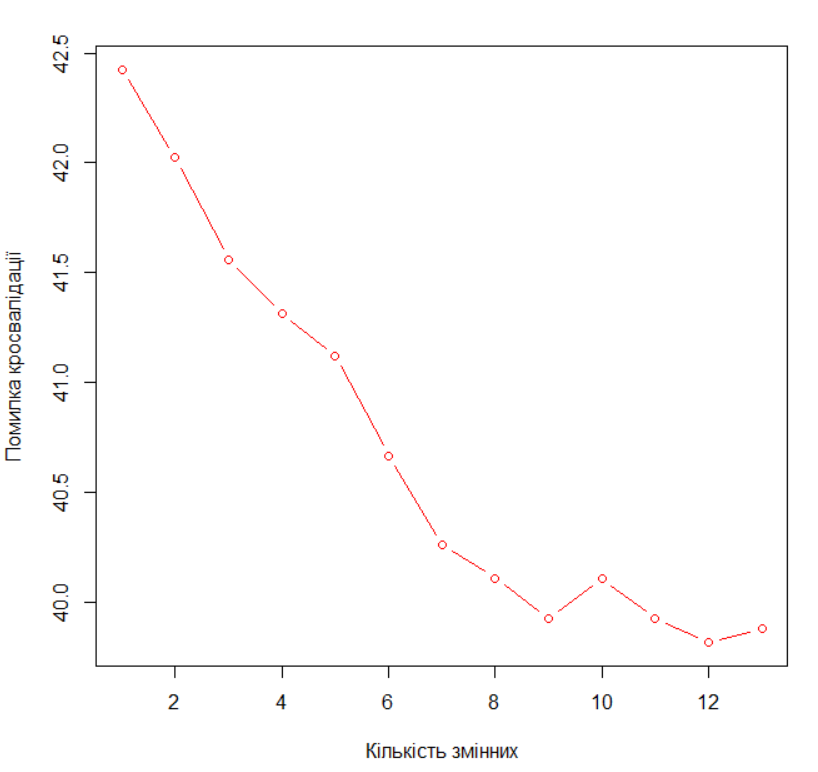
print(paste('Model size for min CV: ', which.min(mean.cv.errors),

 ', min CV error: ', min(mean.cv.errors)))

plot(mean.cv.errors, xlab = "Кількість змінних", ylab = "Помилка кросвалідації",

     col = "red", type = "b")

****

****

# lasso

cat("\n")

library(glmnet)

x = model.matrix(crim ~ ., Boston)[, -1]

y = Boston$crim

cv.lasso = cv.glmnet(x, y, alpha = 1, type.measure = "mse")

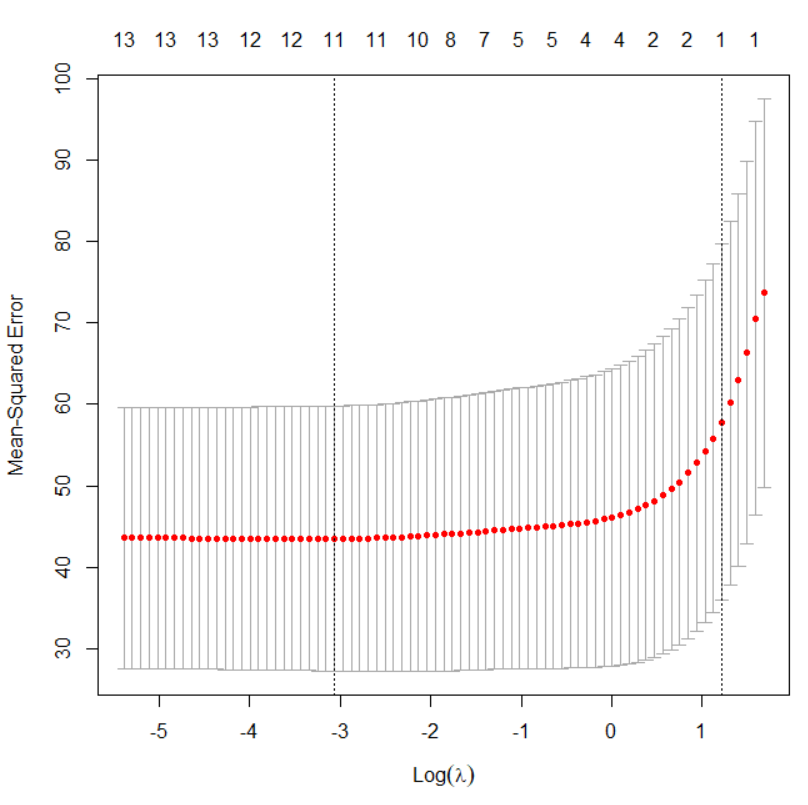
cat("\n")

print(paste('Lasso: Min lambda: ', cv.lasso$lambda.min,

 ', min CV error: ', min(cv.lasso$cvm)))

plot(cv.lasso)

****

****

# ridge

cv.ridge = cv.glmnet(x, y, alpha = 0, type.measure = "mse")

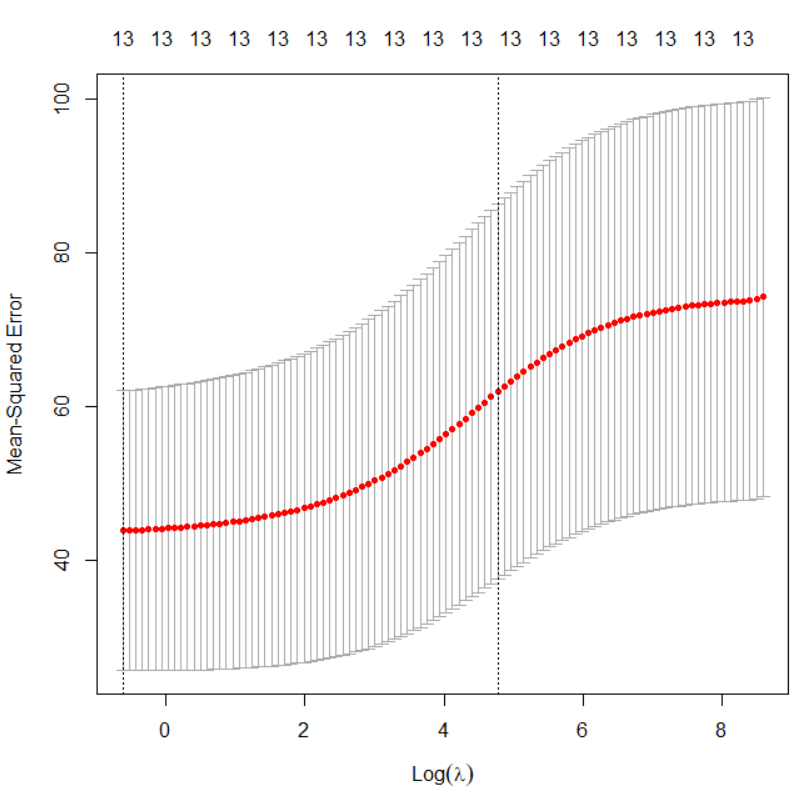
cat("\n")

print(paste('Ridge: Min lambda: ', cv.ridge$lambda.min,

 ', min CV error: ', min(cv.ridge$cvm)))

plot(cv.ridge)

****

****

# PCR

cat("\n")

library(pls)

fit.pcr = pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV")

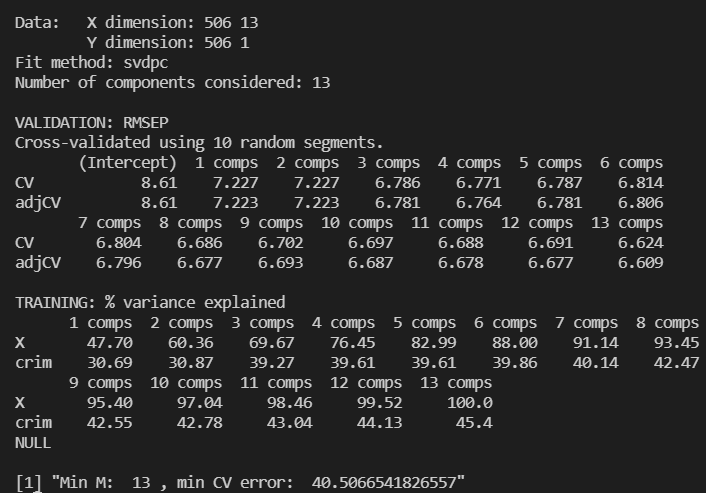
print(summary(fit.pcr))

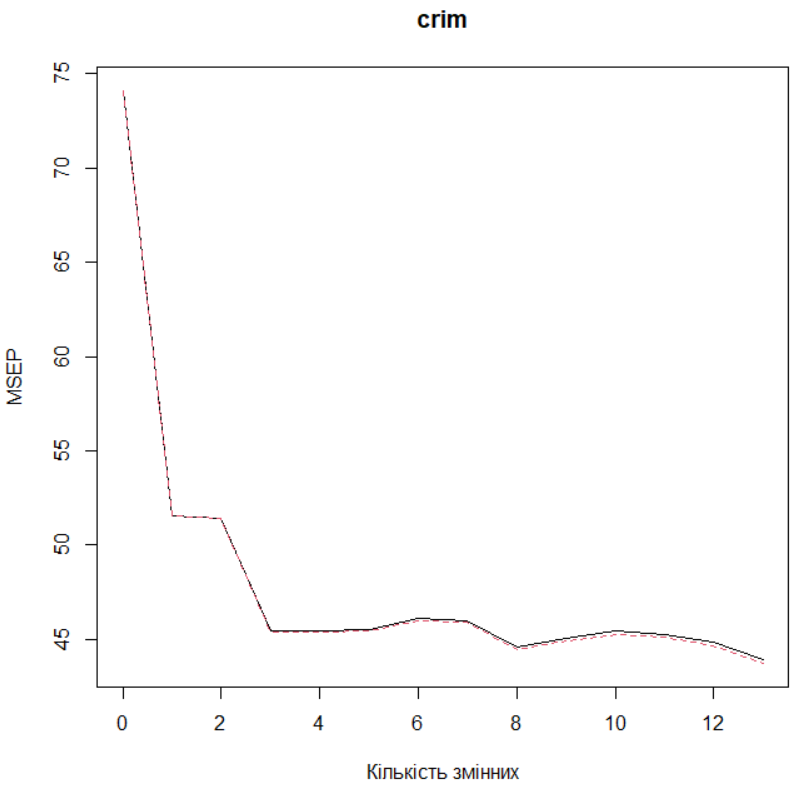
cat("\n")

print(paste('Min M: ', which.min(fit.pcr$validation$adj),

 ', min CV error: ', min(fit.pcr$validation$adj)))

validationplot(fit.pcr, val.type = "MSEP", xlab = "Кількість змінних")

****

****

Оцінюючи результати помилки кросвалідації серед наведених вище методів можна сказати, що найнижчу помилку має метод найкращого вибору підмножини, а саме 39.7152, найвищу помилку має модель гребеневої регресії - 42.8758. При чому, варто наголосити, що модель з найнижчою помилкою для методу найкращого вибору підмножини має 12 предикторів.