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:
Зелінський Олександр

Перевірив:
Проф. Заболоцький Т. М.

Львів – 2021

Хід виконання

1. Вибір найкращої підмножини

1.1

```
> # 1.1
> set.seed(1)
> x = rnorm(100)
> eps = rnorm(100)
```

1.2

$$Y = \beta_0 + \beta_1 X + \beta_2 X_2 + \beta_3 X_3 + \varepsilon,$$

де $\beta_0 = 6.63, \beta_1 = 3.14, \beta_2 = 2.72, \beta_3 = 6.67$

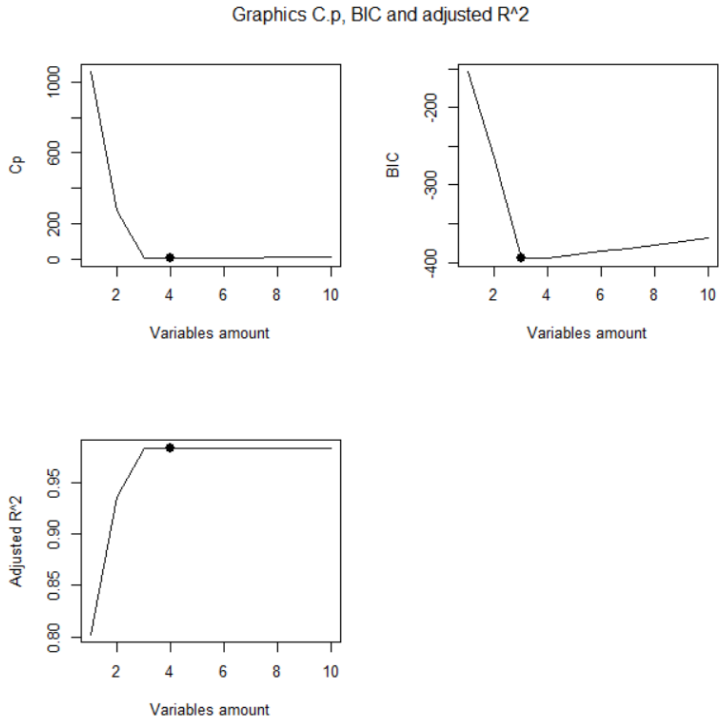
```
> # Plank const, Pi, E, sqrt(2), g const
> betas = c(6.63, 3.14, 2.72, 1.41, 6.67)
>
> y = betas[1] + betas[2] * x +
+   betas[3] * x^2 + betas[4] * x^3 + eps
```

1.3

```
> library(leaps)
> data.frame = data.frame(y = y, x = x)
>
> BestModelSelection = function(methodName, text) {
+     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 = methodName)
+     reg.summary = summary(reg.fit)
+     print(reg.summary)
+
+     par(mfrow = c(2, 2))
+
+     plot(reg.summary$cp, xlab = "Variables amount", ylab = "Cp", type = "l")
+     points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)],
+     col = "blue", cex = 2, pch = 20)
+
+     plot(reg.summary$bic, xlab = "Variables amount", ylab = "BIC", type = "l")
+     points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)],
+     col = "blue", cex = 2, pch = 20)
+
+     plot(reg.summary$adjr2, xlab = "Variables amount", ylab = "Adjusted R^2", type = "l")
+     points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)],
+     col = "blue", cex = 2, pch = 20)
+
+     mtext(text, 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)
+
+     print(coef(reg.fit, point.a))
+     print(coef(reg.fit, point.b))
+     print(coef(reg.fit, point.c))
+ }
> |
```

```
Subset selection object
Call: BestModelSelection("exhaustive", "Graphics C.p, BIC and adjusted R^2")
10 Variables  (and intercept)
       Forced in Forced out
x          FALSE      FALSE
I(x^2)     FALSE      FALSE
I(x^3)     FALSE      FALSE
I(x^4)     FALSE      FALSE
I(x^5)     FALSE      FALSE
I(x^6)     FALSE      FALSE
I(x^7)     FALSE      FALSE
I(x^8)     FALSE      FALSE
I(x^9)     FALSE      FALSE
I(x^10)    FALSE      FALSE
1 subsets of each size up to 10
Selection Algorithm: exhaustive
          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)
1  ( 1 )  " " " "    "*"    " "    " "    " "    " "    " "    " "    " "
2  ( 1 )  " " "*"    "*"    " "    " "    " "    " "    " "    " "    " "
3  ( 1 )  "*" "*"    "*"    " "    " "    " "    " "    " "    " "    " "
4  ( 1 )  "*" "*"    "*"    " "    "*"    " "    " "    " "    " "    " "
5  ( 1 )  "*" "*"    "*"    " "    "*"    "*"    " "    " "    " "    " "
6  ( 1 )  "*" "*"    "*"    " "    " "    " "    "*"    "*"    "*"    " "
7  ( 1 )  "*" "*"    "*"    " "    "*"    "*"    " "    "*"    " "    "*"
8  ( 1 )  "*" "*"    "*"    "*"    " "    "*"    " "    "*"    "*"    "*"
9  ( 1 )  "*" "*"    " "    "*"    "*"    "*"    "*"    "*"    "*"    "*"
10  ( 1 ) "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"

(Intercept)           x      I(x^2)      I(x^3)      I(x^5)
 6.70200775  3.52745596  2.56575641  0.96797426  0.08072292
(Intercept)           x      I(x^2)      I(x^3)
   6.691507    3.115280    2.596209    1.427639
(Intercept)           x      I(x^2)      I(x^3)      I(x^5)
 6.70200775  3.52745596  2.56575641  0.96797426  0.08072292
```

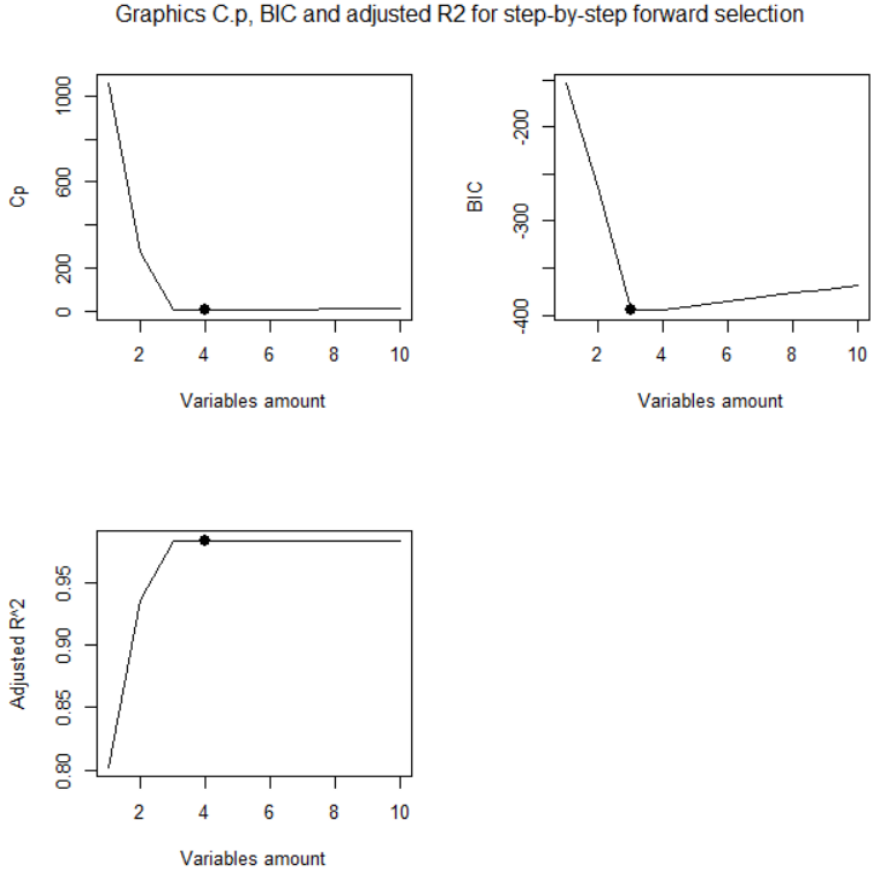
В результаті можемо побачити найкращий набір змінних для кожної розмірності моделі, також видно, що * позначають те, що змінна включена у відповідну модель.



З огляду на результати видно, що найкраща модель за показниками C_p та скорегованим R^2 – це модель з 4 змінними (x, x^2, x^3, x^5), а для показника BIC бачимо, що вже найкращою буде модель зі змінними (x, x^2, x^3).

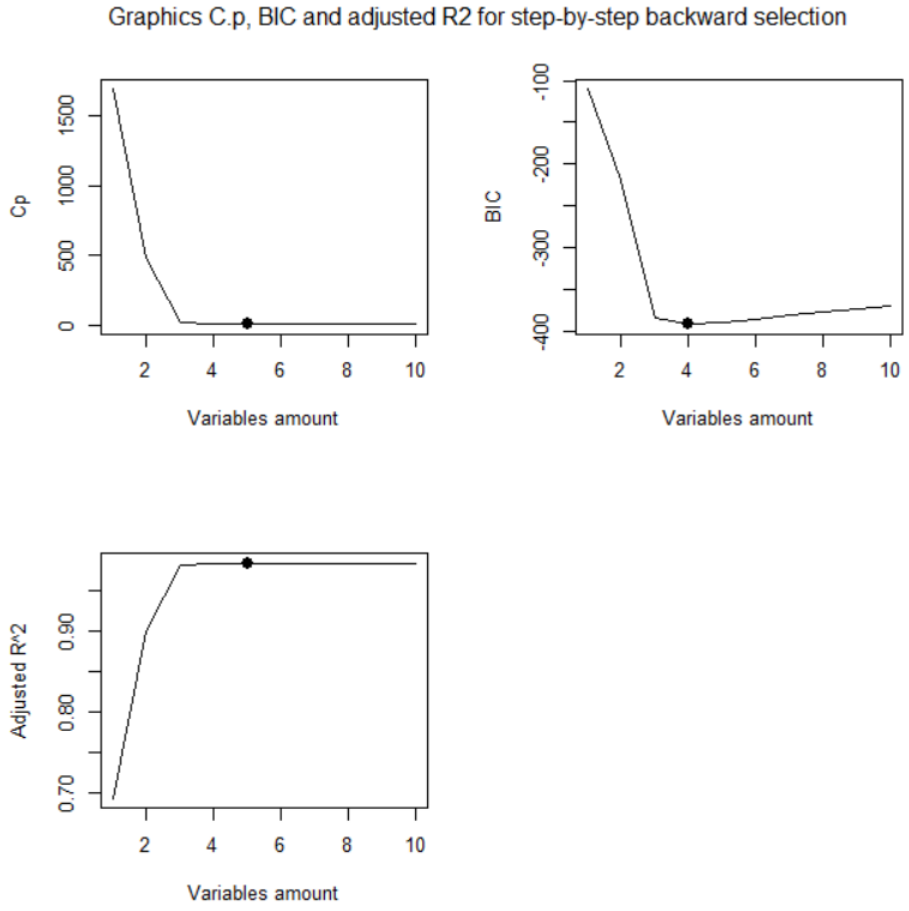
1.4

```
(Intercept)           x      I(x^2)      I(x^3)      I(x^5)
 6.70200775  3.52745596  2.56575641  0.96797426  0.08072292
(Intercept)           x      I(x^2)      I(x^3)
   6.691507    3.115280    2.596209    1.427639
(Intercept)           x      I(x^2)      I(x^3)      I(x^5)
 6.70200775  3.52745596  2.56575641  0.96797426  0.08072292
```



З огляду на результати видно, що для покрокового вибору вперед найкраща модель за показниками C_p та скорегованим R^2 – це модель з 4 змінними (x, x^2, x^3, x^5), а для показника BIC бачимо, що вже найкращою буде модель зі змінними (x, x^2, x^3).

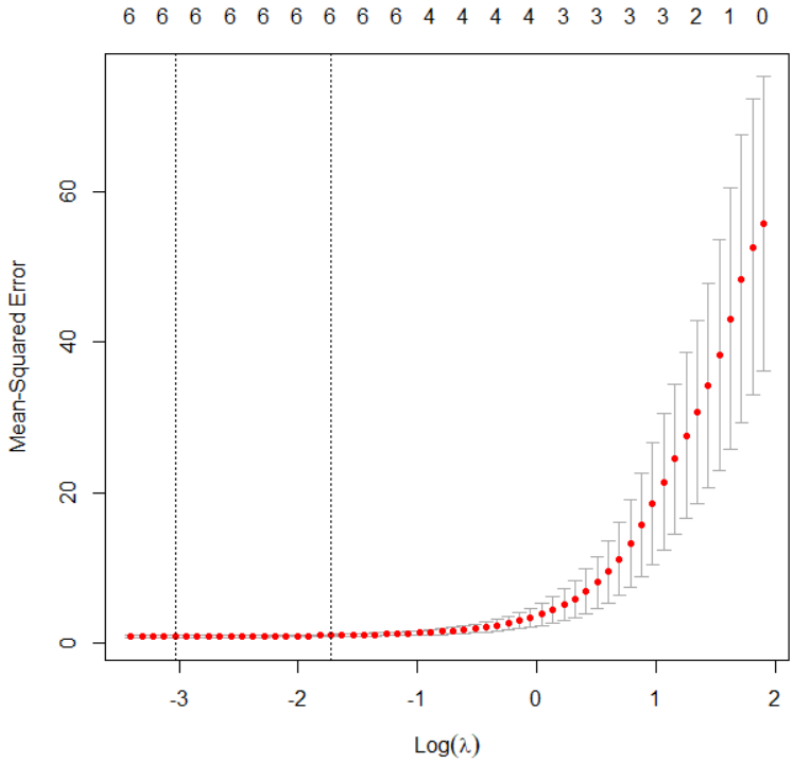
```
(Intercept)           x      I(x^2)      I(x^5)      I(x^7)      I(x^9)
 6.70306390  3.87626224  2.56344561  0.74770838 -0.16284655  0.01294634
(Intercept)           x      I(x^2)      I(x^5)      I(x^7)
 6.68149256  4.07444717  2.59747728  0.46614707 -0.04184129
(Intercept)           x      I(x^2)      I(x^5)      I(x^7)      I(x^9)
 6.70306390  3.87626224  2.56344561  0.74770838 -0.16284655  0.01294634
```



З огляду на результати видно, що для покрокового вибору назад найкраща модель за показниками C_p та скорегованим R^2 – це модель з 5 змінними (x, x^2, x^5, x^7, x^9), а для показника BIC бачимо, що вже найкращою буде модель з 4 змінними (x, x^2, x^5, x^7).

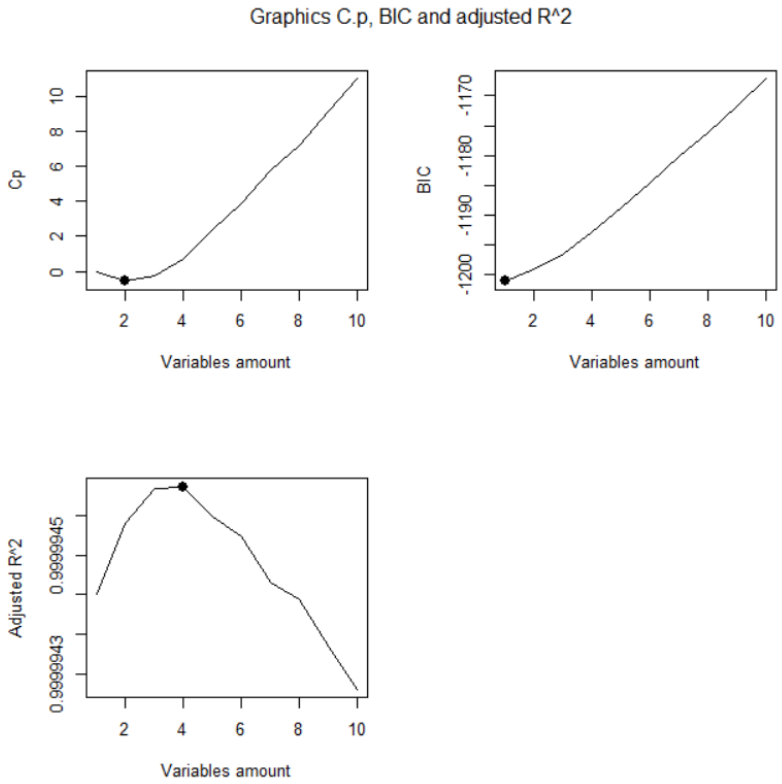
1.5

```
[1] "Min lambda:  0.0482813980063348"
(Intercept)           x      I(x^2)      I(x^3)      I(x^4)      I(x^5)      I(x^7)
6.790778147 3.355231723 2.369236391 1.152028598 0.040760422 0.032082840 0.002544071
```



Згідно з графіком, при значеннях логарифма від лямбди більших за нуль помилка зростає. Лямба для якої помилка буде найменшою дорівнює 0.048, а модель включає в себе змінні $(x, x^2, x^3, x^4, x^5, x^7)$.

1.6

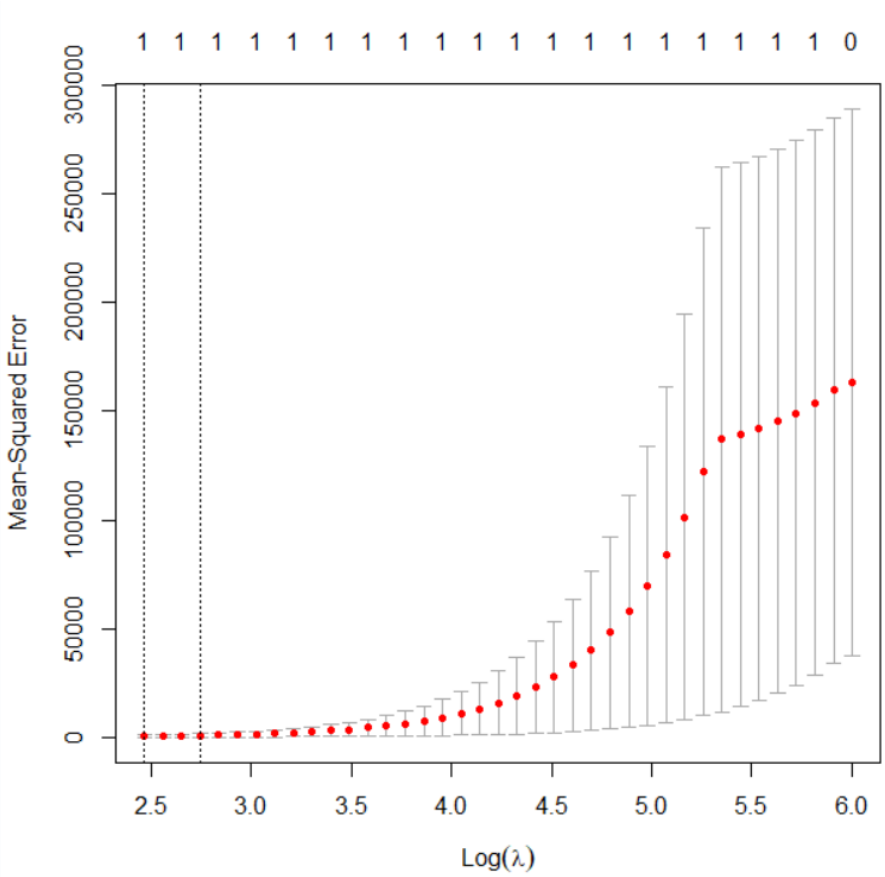


З огляду на результати видно, що найкраща модель за показниками C_p – (x^2, x^7), за скорегованим R^2 – це модель (x^7), а для показника BIC бачимо, що вже найкращою буде модель зі змінними (x, x^2, x^3, x^7).

Оскільки $\beta_7 = 6{,}6707$, то можемо з впевненістю сказати, що метод найкращого вибору підмножини з використанням BIC визначає найбільш точну модель з однією змінною.

```
(Intercept)      I(x^2)      I(x^7)
  6.7004904  -0.1417084   6.6715552
(Intercept)      I(x^7)
    6.58894     6.67077
(Intercept)           x      I(x^2)      I(x^3)      I(x^7)
  6.7062524   0.2914016  -0.1617671  -0.2526527   6.6791338
> Lasso()

[1] "Min lambda:  11.7857993518626"
(Intercept)      I(x^7)
   7.409616    6.476314
```

В методі лассо, ми знайшли лямбду для якої помилка мінімальна, яка дорівнює 7.41 і з допомогою неї знайшли найкращу модель, яка включає одну змінну x^7.



2. Collage

2.1

```
> set.seed(1)
> train = sample(1:dim(College)[1], 0.5 * dim(College)[1])
> College.train = College[train, ]
> College.test = College[-train, ]
```

2.2

```
> fit.lm = lm(Apps ~ ., data = College.train)
> pred.lm = predict(fit.lm, College.test)
> round(mean((pred.lm - College.test$Apps)^2), 2)
[1] 1135758
```

2.3

```
> library(glmnet)
> 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)
>
> dim(coef(fit.ridge))
[1]  19 100
>
> bestlam = cv.ridge$lambda.min
> paste('Best lambda:', bestlam)
[1] "Best lambda: 0.01"
> pred.ridge = predict(fit.ridge, s = bestlam, newx = test.mat)
> round(mean((pred.ridge - College.test$Apps)^2), 2)
[1] 1134677
```

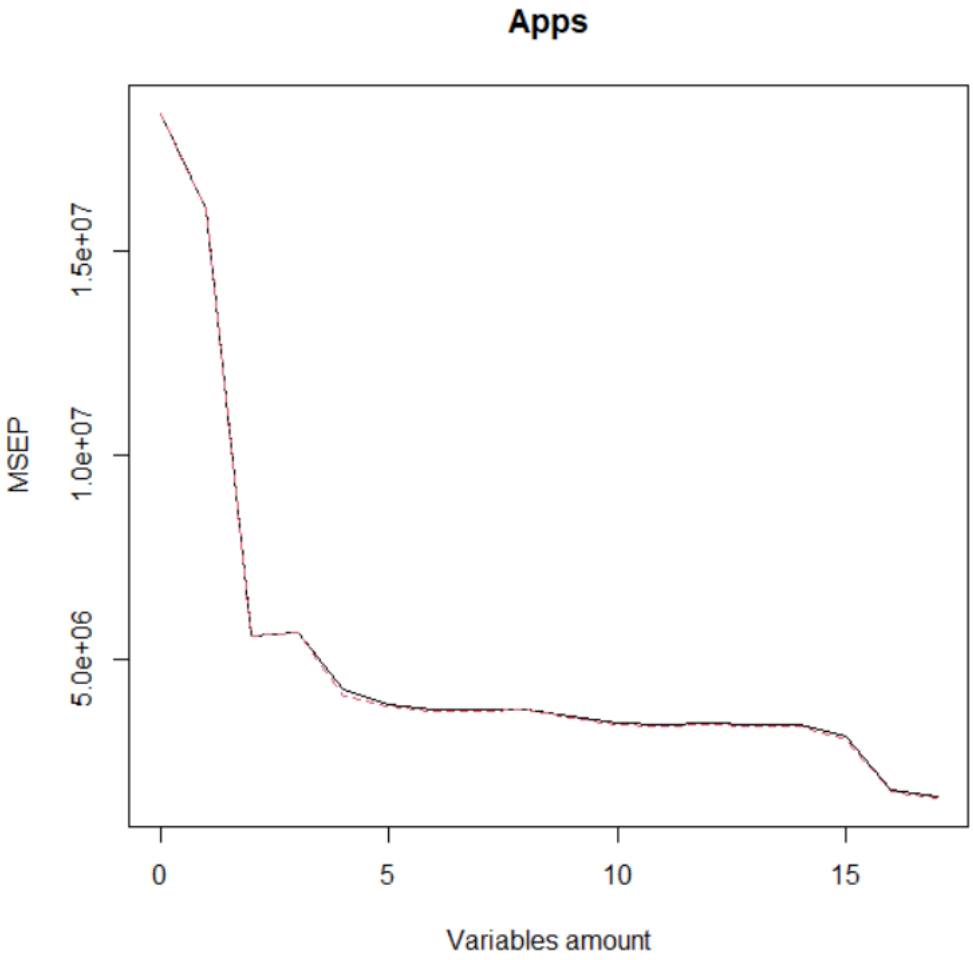
2.4

```
> 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)
>
> bestlam = cv.lasso$lambda.min
> paste('Best lambda:', bestlam)
[1] "Best lambda: 0.01"
> pred.lasso = predict(fit.lasso, s = bestlam, newx = test.mat)
> round(mean((pred.lasso - College.test$Apps)^2), 2)
[1] 1133422
>
> lasso_coefs = predict(fit.lasso, s = bestlam, type = "coefficients")
> lasso_coefs
19 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept) -7.931498e+02
(Intercept)  .
PrivateYes  -3.078903e+02
Accept       1.777242e+00
Enroll      -1.450532e+00
Top10perc    6.659456e+01
Top25perc   -2.221506e+01
F.Undergrad  8.983869e-02
P.Undergrad  1.005260e-02
Outstate    -1.082871e-01
Room.Board   2.118762e-01
Books        2.922508e-01
Personal     6.234085e-03
PhD         -1.542914e+01
Terminal     6.364841e+00
S.F.Ratio    2.284667e+01
perc.alumni  1.114025e+00
Expend       4.861825e-02
Grad.Rate    7.466015e+00
```

Можемо бачити, що жоден з коефіцієнтів не рівний нулю

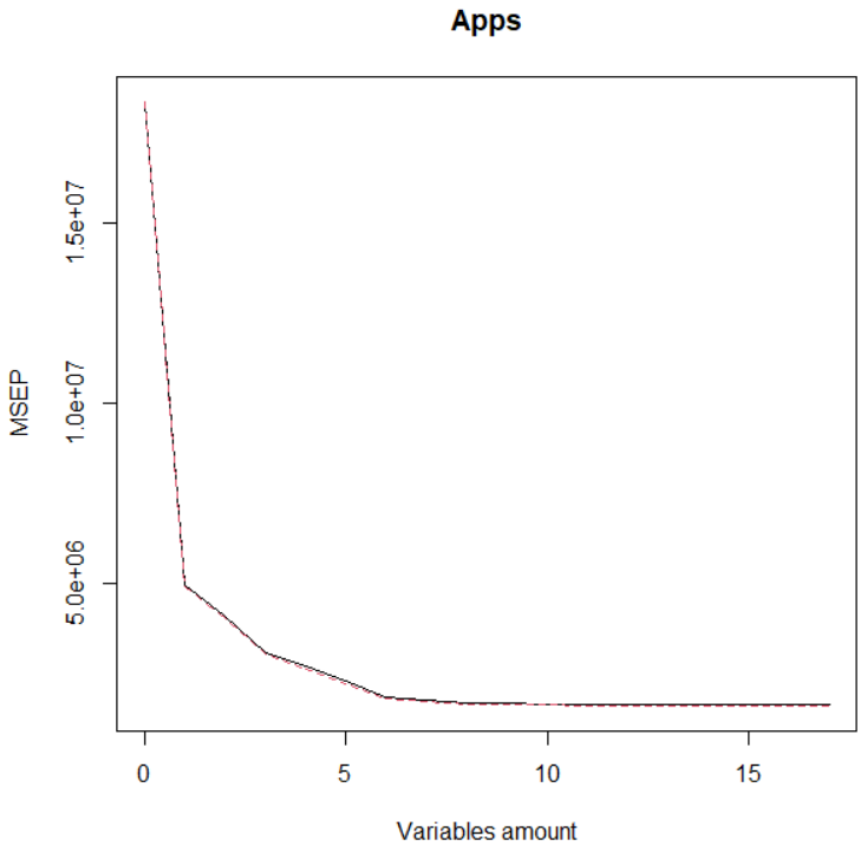
2.5

```
> library(pls)

Attaching package: 'pls'

The following object is masked from 'package:stats':

    loadings

> fit.pcr = pcr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
> validationplot(fit.pcr, val.type = "MSEP")
>
> paste('Min M: ', which.min(fit.pcr$validation$adj))
[1] "Min M:  17"
> pred.pcr = predict(fit.pcr, College.test, ncomp = which.min(fit.pcr$validation$adj))
> round(mean((pred.pcr - College.test$Apps)^2), 2)
[1] 1135758
```



2.6

```
> fit.pls = plsr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
> validationplot(fit.pls, val.type = "MSEP")
>
> paste('Min M: ', which.min(fit.pls$validation$adj))
[1] "Min M:  16"
> pred.pls = predict(fit.pls, College.test, ncomp = which.min(fit.pls$validation$adj))
> round(mean((pred.pls - College.test$Apps)^2), 2)
[1] 1135812
```



2.7

```
> # 2.7
> 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")

> paste("lm prediction R^2: ", GetRSqr(pred.lm), " %")
[1] "lm prediction R^2:  90.15413  %"
> paste("ridge prediction R^2: ", GetRSqr(pred.ridge), " %")
[1] "ridge prediction R^2:  90.16351  %"
> paste("lasso prediction R^2: ", GetRSqr(pred.lasso), " %")
[1] "lasso prediction R^2:  90.17438  %"
> paste("pcr prediction R^2: ", GetRSqr(pred.pcr), " %")
[1] "pcr prediction R^2:  90.15413  %"
> paste("pls prediction R^2: ", GetRSqr(pred.pls), " %")
[1] "pls prediction R^2:  90.15366  %"
```

З обчислених коефіцієнтів детермінації видно, що R^2 є найближчийм (90.1744%) для моделі лассо і найнижчим для моделі PLS (90,1536%). Різниця між тестовими помилками не значна (~2000).

3. Collage

3.1

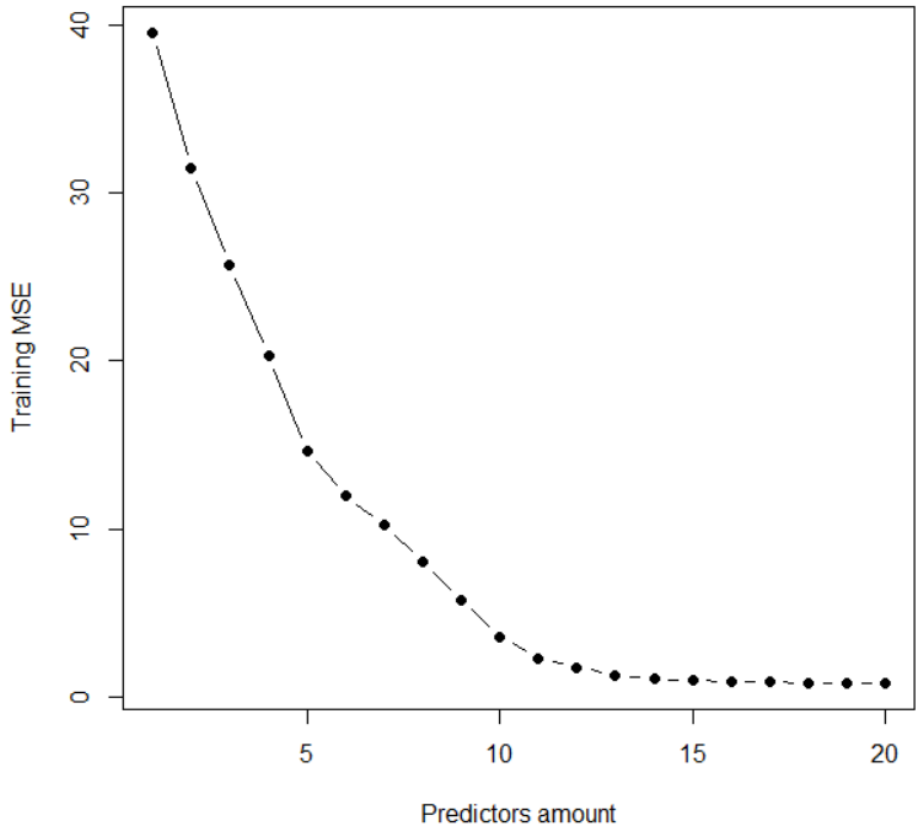
```
> # 3.1
> set.seed(1)
> x = matrix(rnorm(1000 * 20), 1000, 20)
> eps = rnorm(1000)
>
> betas = runif(20, min=-2.72, max=3.14)
>
> zero_vals_num = sample(3:10, 1)
> for (i in sample(1:length(betas), zero_vals_num)) {
+   betas[i] = 0
+ }
> betas
 [1]  1.5350075  0.0000000 -2.7034295  2.7980721  0.6320459  2.4600726
 [7]  0.0000000 -1.5060529  0.0000000  0.0000000  2.4158184  1.6200996
[13]  0.7906195 -2.1023079 -1.9238998 -0.7934472  1.0609377  0.2005645
[19] -2.5021053  0.0000000
>
> y = x %*% betas + eps
```

3.2

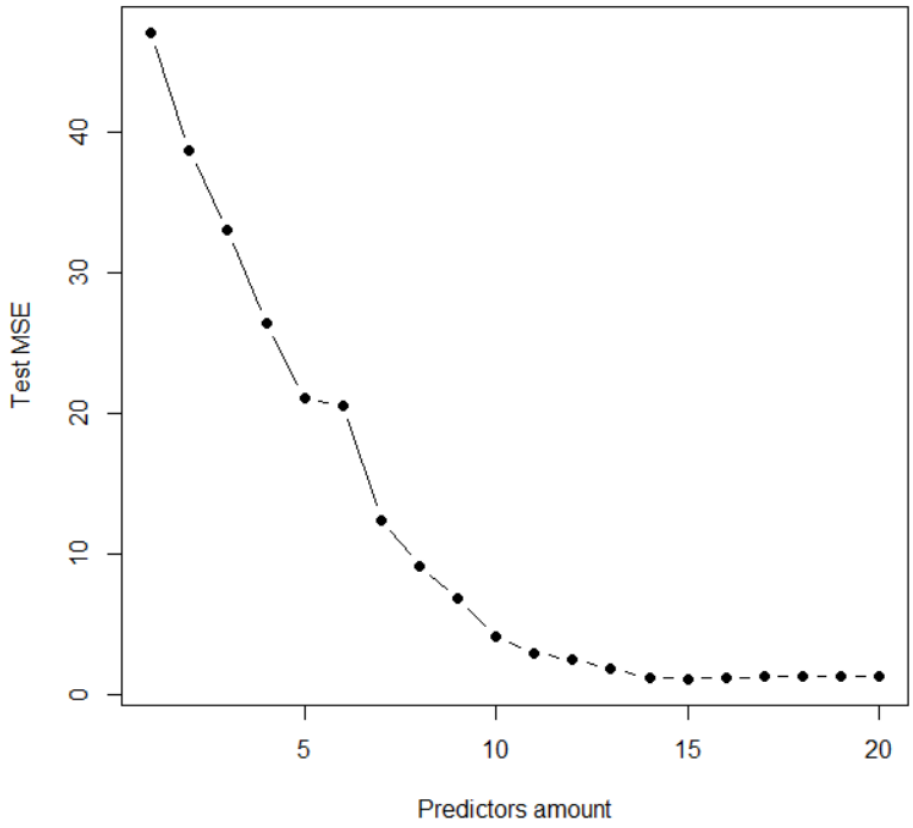
```
> train = sample(1:length(eps), 100)
>
> x.train = x[train, ]
> y.train = y[train]
>
> x.test = x[-train, ]
> y.test = y[-train]
```

3.3

```
> library(leaps)
>
> reg.fit = regsubsets(y ~ .,
+                      data = data.frame(y = y.train, x = x.train),
+                      nvmax = 20)
>
> BestSubsetSelection = function(x, y, ylab) {
+     localData = data.frame(y = y, x = x)
+     mat = model.matrix(y ~ ., data = localData, 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 = "Predictors amount", ylab = ylab, pch = 19, type = "b")
+
+     return(val.errors)
+ }
>
> BestSubsetSelection(x.train, y.train, "Training MSE")
 [1] 39.5019992 31.4815121 25.6894467 20.2829630 14.6205946 11.9391628
 [7] 10.2216451  8.0384360  5.7718576  3.5877601  2.2857950  1.7702461
[13]  1.3107050  1.0733538  0.9809518  0.9116284  0.8680321  0.8530347
[19]  0.8459438  0.8459334
```



3.4



3.5

```
> paste('Model size for min MSE: ', which.min(test_mse), ', min MSE: ', min(test_mse))
[1] "Model size for min MSE:  15 , min MSE:  1.17025329733113"
```

Видно, що модель має 15 змінних, а MSE дорівнює 1.17.

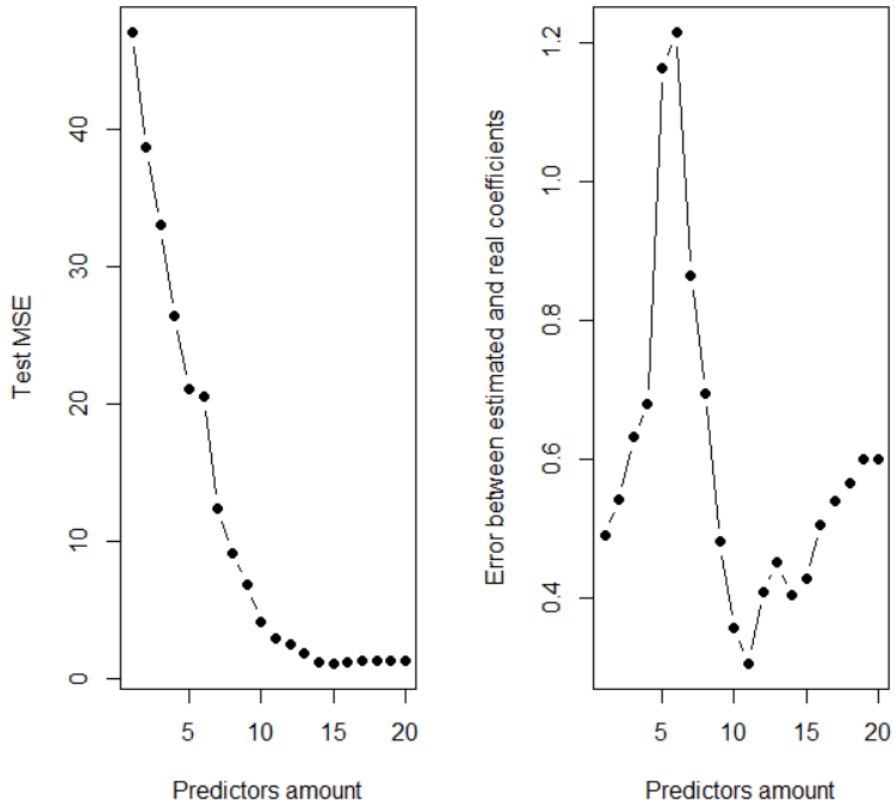
3.6

```
> coef(reg.fit, which.min(test_mse))
(Intercept)         x.1         x.3         x.4         x.5         x.6         x.8        x.11        x.12
  0.0483791   1.6249734  -2.6446191   2.7726741   0.8276008   2.4792278  -1.6001762   2.3008815   1.6403113
       x.13        x.14        x.15        x.16        x.17        x.18        x.19
  0.5978145  -2.1394335  -1.8138665  -0.7183958   1.2395690   0.3297070  -2.5767892
```

З результату видно, що модель правильно визначила нульові коефіцієнти і виключила їх з моделі, а решта оцінок є точною.

3.7

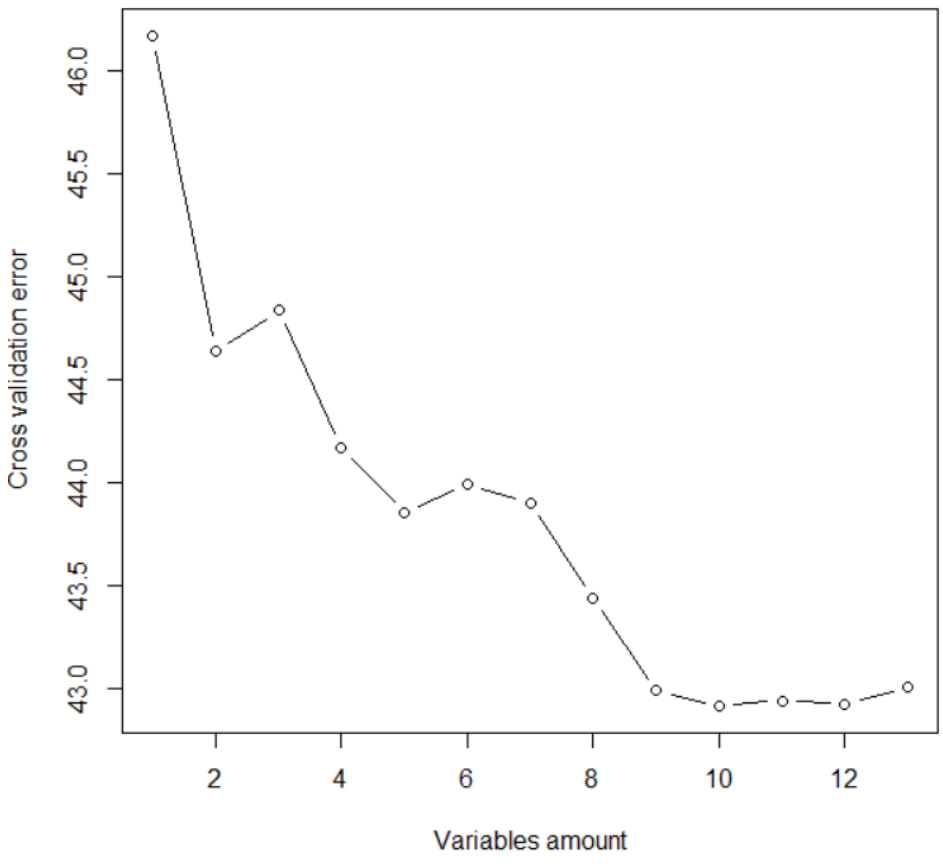
```
> val.errors = rep(0, 20)
> x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
>
> for (i in 1:20) {
+     coef_i = coef(reg.fit, id = i)
+     val.errors[i] =
+     sqrt(sum((betas[x_cols %in% names(coef_i)] -
+          coef_i[names(coef_i) %in% x_cols])^2))
+ }
>
> par(mfrow = c(1, 2))
>
> BestSubsetSelection(x.test, y.test, "Test MSE")
 [1] 47.019131 38.633316 32.987160 26.381954 21.113270 20.547826 12.463378  9.192976  6.869330  4.145006
[11]  3.033259  2.512932  1.887346  1.230968  1.170253  1.261337  1.299058  1.325936  1.371570  1.371136
>
> plot(val.errors, xlab = "Predictors amount",
+  ylab = "Error between estimated and real coefficients", pch = 19, type = "b")
```



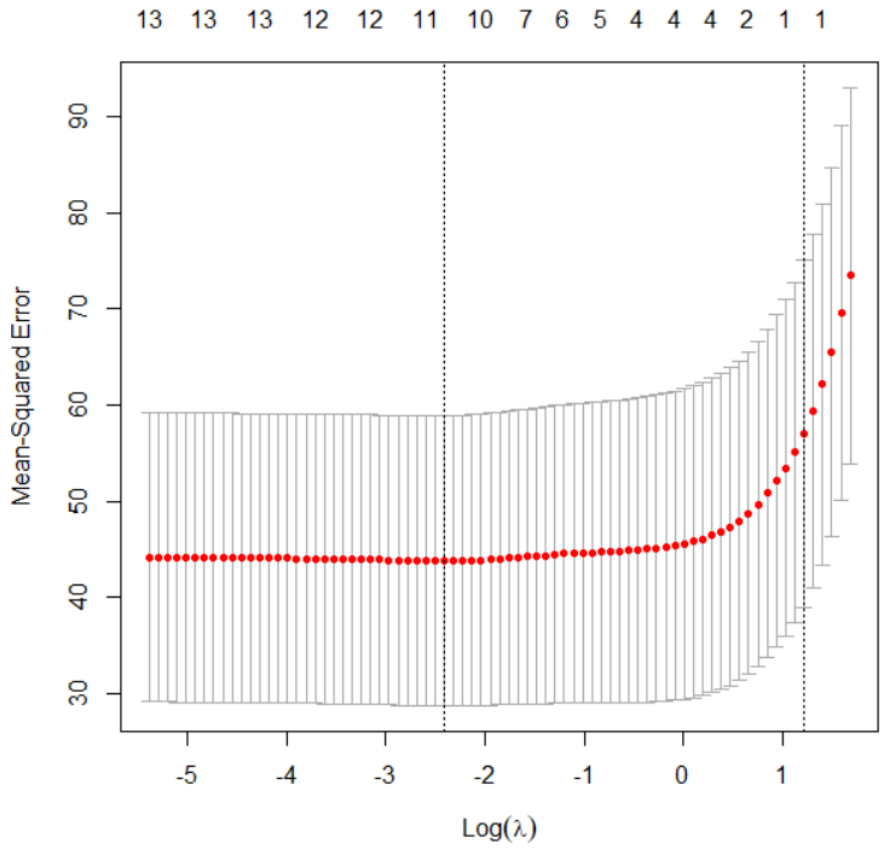
В результаті видно, що модель з одинадцятьма предикторами мінімізує помилку між оціночними та справжніми значеннями коефіцієнтів.

4. Boston

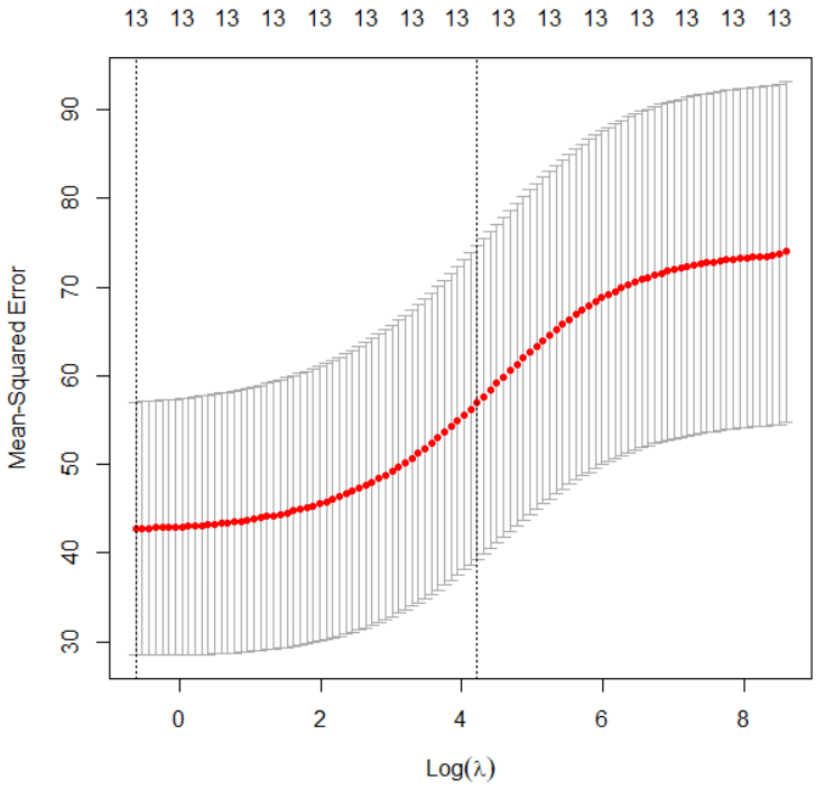
```
> predict.regsubsets = function(object, newdata, id, ...) {
+     form = as.formula(object$call[[2]])
+     mat = model.matrix(form, newdata)
+     coef_i = coef(object, id = id)
+     xvars = names(coef_i)
+     mat[, xvars] %*% coef_i
+ }
>
> k = 10
> folds = sample(1:k, nrow(Boston), replace = TRUE)
> cv.errors = matrix(0, k, 13)
>
> for (j in 1:k) {
+     best.fit = regsubsets(crim ~ ., data = Boston[folds != j, ], nvmax = 13)
+     for (i in 1:13) {
+         pred = predict.regsubsets(best.fit, Boston[folds == j, ], id = i)
+         cv.errors[j, i] = mean((Boston$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")

> paste('Model size for min CV: ', which.min(mean.cv.errors), ', min CV error: ', min(mean.cv.errors))
[1] "Model size for min CV:  10 , min CV error:  42.9215621665693"
> plot(mean.cv.errors, xlab = "Variables amount",
+   ylab = "Cross validation error",type = "b")
```



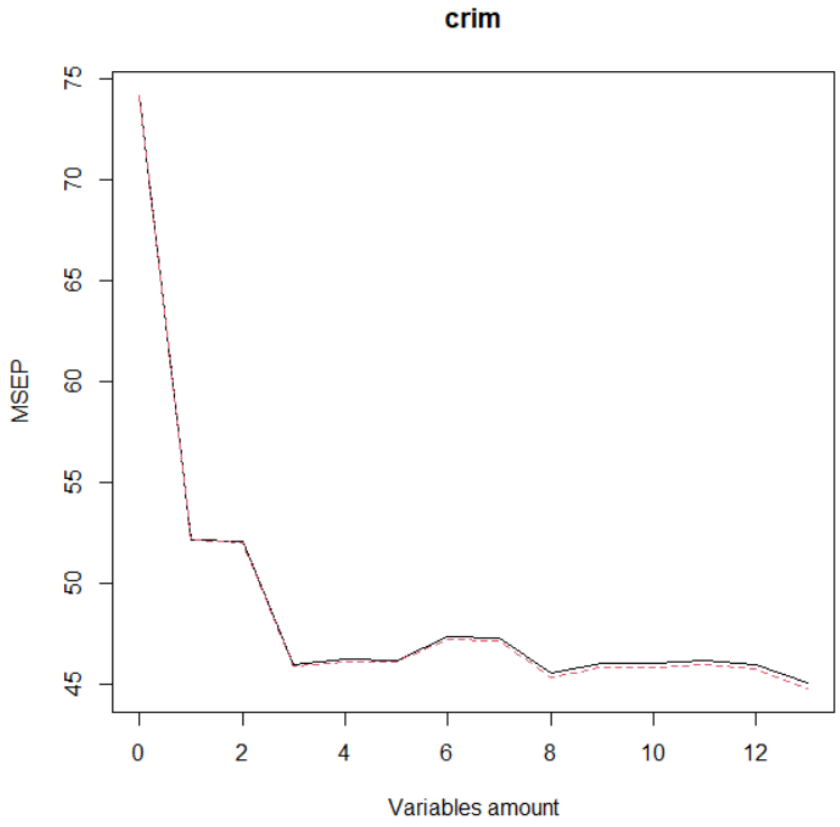
```
> library(glmnet)
> x = model.matrix(crim ~ ., Boston)[, -1]
> y = Boston$crim
>
> cv.lasso = cv.glmnet(x, y, alpha = 1, type.measure = "mse")
> cat("\n")

> paste('Lasso: Min lambda: ', cv.lasso$lambda.min, ', min CV error: ', min(cv.lasso$cvm))
[1] "Lasso: Min lambda:  0.0896602637403995 , min CV error:  43.7964796465473"
> plot(cv.lasso)
```




```
> cv.ridge = cv.glmnet(x, y, alpha = 0, type.measure = "mse")
> cat("\n")

> paste('Ridge: Min lambda: ', cv.ridge$lambda.min, ', min CV error: ', min(cv.ridge$cvm))
[1] "Ridge: Min lambda:  0.537499162479542 , min CV error:  42.8483656762509"
> plot(cv.ridge)
```




```
> fit.pcr = pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV")
> summary(fit.pcr)
Data:   X dimension: 506 13
        Y dimension: 506 1
Fit method: svdpc
Number of components considered: 13

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps
CV            8.61    7.222    7.215    6.780    6.797    6.796    6.883    6.878    6.748    6.784     6.784
adjCV         8.61    7.218    7.212    6.774    6.790    6.789    6.872    6.867    6.735    6.771     6.769
       11 comps  12 comps  13 comps
CV        6.796     6.780     6.713
adjCV     6.780     6.762     6.693

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
X       47.70    60.36    69.67    76.45    82.99    88.00    91.14    93.45    95.40     97.04     98.46
crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14    42.47    42.55     42.78     43.04
      12 comps  13 comps
X        99.52     100.0
crim     44.13      45.4
>
> cat("\n")

> paste('Min M: ', which.min(fit.pcr$validation$adj), ', min CV error: ', min(fit.pcr$validation$adj))
[1] "Min M:  13 , min CV error:  40.5749865125202"
>
> validationplot(fit.pcr, val.type = "MSEP", xlab = "Variables amount")
```



Зважаючи на результати, можна сказати, що найнижчу помилку має метод PCR, а саме 40.575, найвищу помилку має метод ласо – 43.796. При чому, модель з найнижчою помилкою для методу найкращого вибору підмножини має 13 предикторів.