МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

ЗВІТ

до індивідуального завдання №6

з дисципліни «Моделі статистичного навчання»

Виконав
студент групи ПМіМ-12:
Бордун Михайло

Перевірив:
Проф. Заболоцький Т. М.

Львів – 2021

<u>**Хід виконання**</u>

1. Додатково проаналізуєте набір даних Wage.

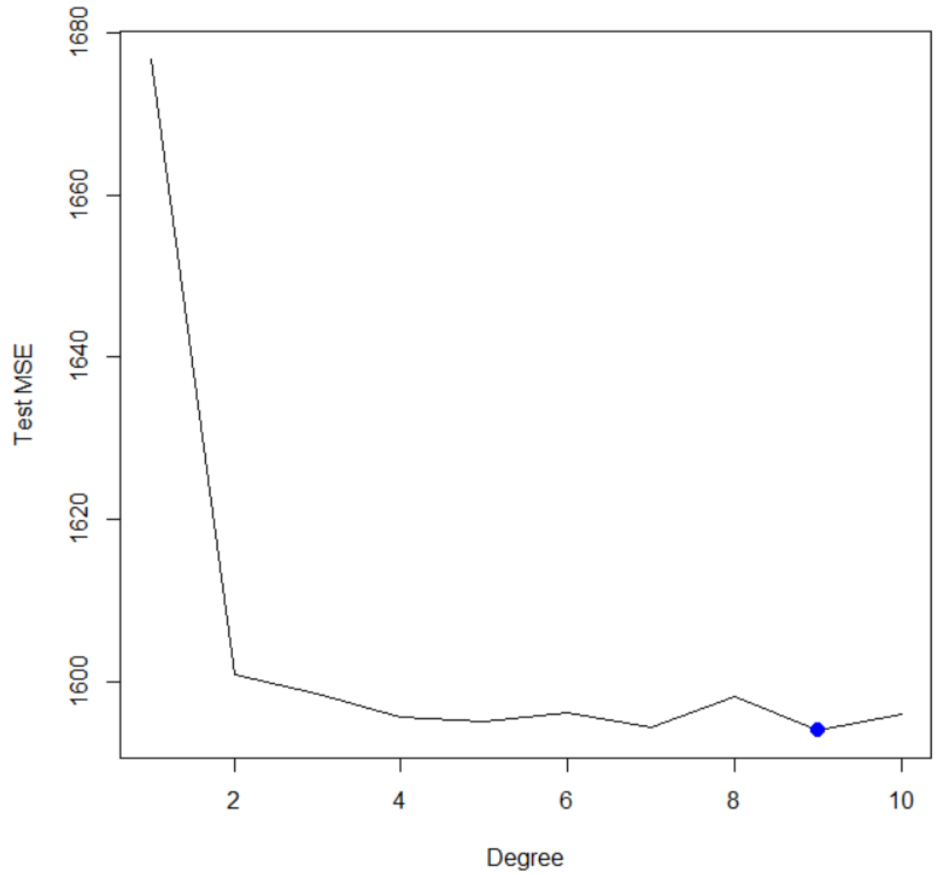
```
     year           age                     maritl           race     
 Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480  
 1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293  
 Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190  
 Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37  
 3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55                  
 Max.   :2009   Max.   :80.00                                          

              education                     region                jobclass   
 1. < HS Grad      :268   2. Middle Atlantic   :3000   1. Industrial :1544  
 2. HS Grad        :971   1. New England       :   0   2. Information:1456  
 3. Some College   :650   3. East North Central:   0                        
 4. College Grad   :685   4. West North Central:   0                        
 5. Advanced Degree:426   5. South Atlantic    :   0                        
                          6. East South Central:   0                        
                          (Other)              :   0                        
            health      health_ins      logwage           wage       
 1. <=Good     : 858   1. Yes:2083   Min.   :3.000   Min.   : 20.09  
 2. >=Very Good:2142   2. No : 917   1st Qu.:4.447   1st Qu.: 85.38  
                                     Median :4.653   Median :104.92  
                                     Mean   :4.654   Mean   :111.70  
                                     3rd Qu.:4.857   3rd Qu.:128.68  
                                     Max.   :5.763   Max.   :318.34  
```

Характеристика даних Wage

1.1. Використайте поліноміальну регресію для прогнозування wage за age. Використайте перехресну перевірку для вибору оптимального степеня d для полінома. Який степінь було обрано, і як це співвідноситься з результатами перевірки гіпотез з використанням ANOVA? Побудуйте графік отриманого поліному пристосованого до даних.

```
deltas = rep(0, 10)

for (i in 1:10) {
    fit = glm(wage ~ poly(age, i), data = Wage)
    deltas[i] = cv.glm(Wage, fit, K = 10)$delta[1]
}

plot(1:10, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")
points(which.min(deltas), deltas[which.min(deltas)],
```

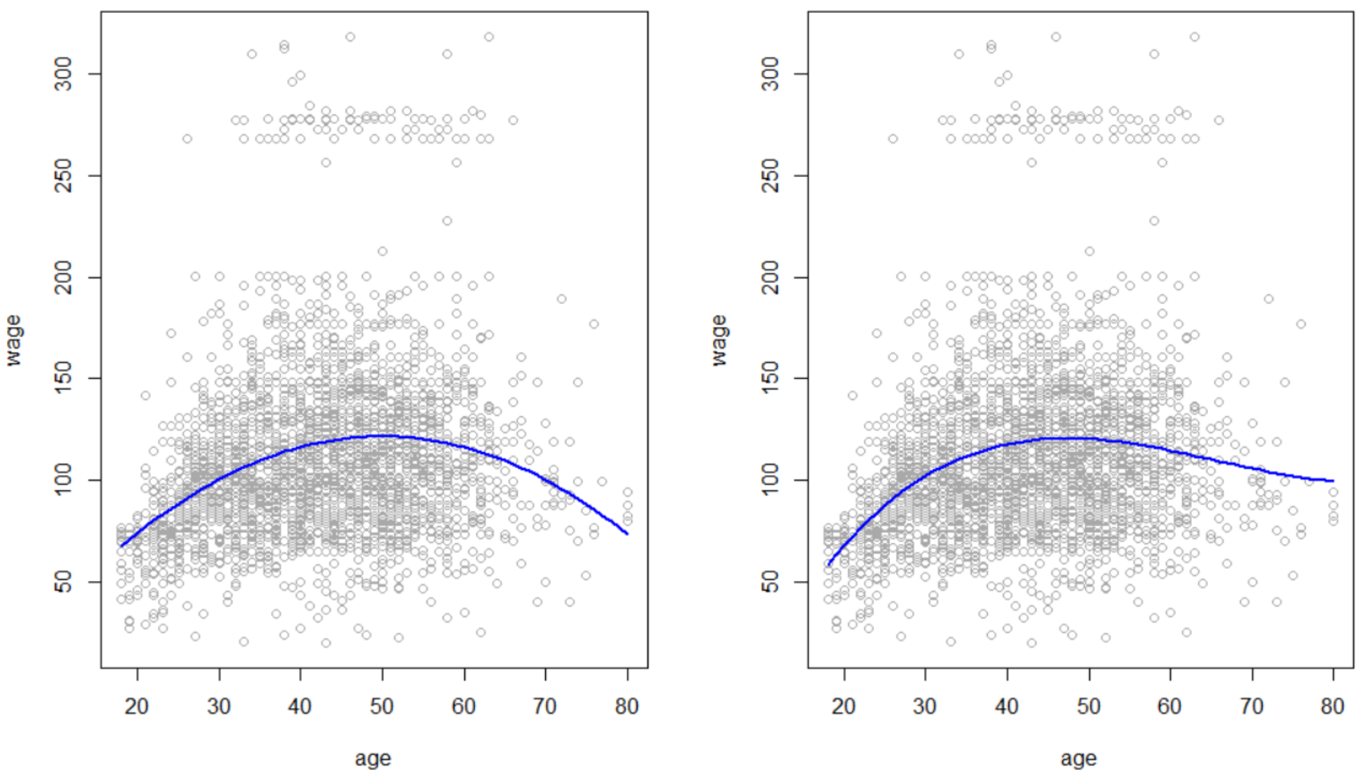
```
    col = "blue", cex = 2, pch = 20)
print(paste('Polynom power for min MSE: ', which.min(deltas)))
cat("\n")

fit1 = lm(wage ~ age, data = Wage)
fit2 = lm(wage ~ poly(age, 2), data = Wage)
fit3 = lm(wage ~ poly(age, 3), data = Wage)
fit4 = lm(wage ~ poly(age, 4), data = Wage)
fit5 = lm(wage ~ poly(age, 5), data = Wage)
fit6 = lm(wage ~ poly(age, 6), data = Wage)
fit7 = lm(wage ~ poly(age, 7), data = Wage)
fit8 = lm(wage ~ poly(age, 8), data = Wage)
fit9 = lm(wage ~ poly(age, 9), data = Wage)
fit10 = lm(wage ~ poly(age, 10), data = Wage)
print(anova(fit1, fit2, fit3, fit4, fit5,
 fit6, fit7, fit8, fit9, fit10))

PlotFunc = function(poly_pow = 0, is_poly = FALSE, is_cut = FALSE) {
    plot(wage ~ age, data = Wage, col = "darkgray")

    age.grid = seq(from = range(Wage$age)[1],
     to = range(Wage$age)[2])

    if (is_poly) {
        fit = lm(wage ~ poly(age, poly_pow), data = Wage)
        pred = predict(fit, newdata = list(age = age.grid))
        lines(age.grid, pred, col = "blue", lwd = 2)
    }
    if (is_cut) {
        fit = lm(wage ~ cut(age, 8), data = Wage)
        pred = predict(fit, newdata = list(age = age.grid))
        lines(age.grid, pred, col = "blue", lwd = 2)
    }
}

par(mfrow = c(1, 2))
PlotFunc(2, TRUE)
PlotFunc(3, TRUE)
```




```
"Polynom power for min MSE:  9"
```

З огляду на результат перехресної перевірки із значенням k=10 бачимо, що при d=9 в нас є оптимальний степінь для полінома в сенсі мінімальної тестової помилки.

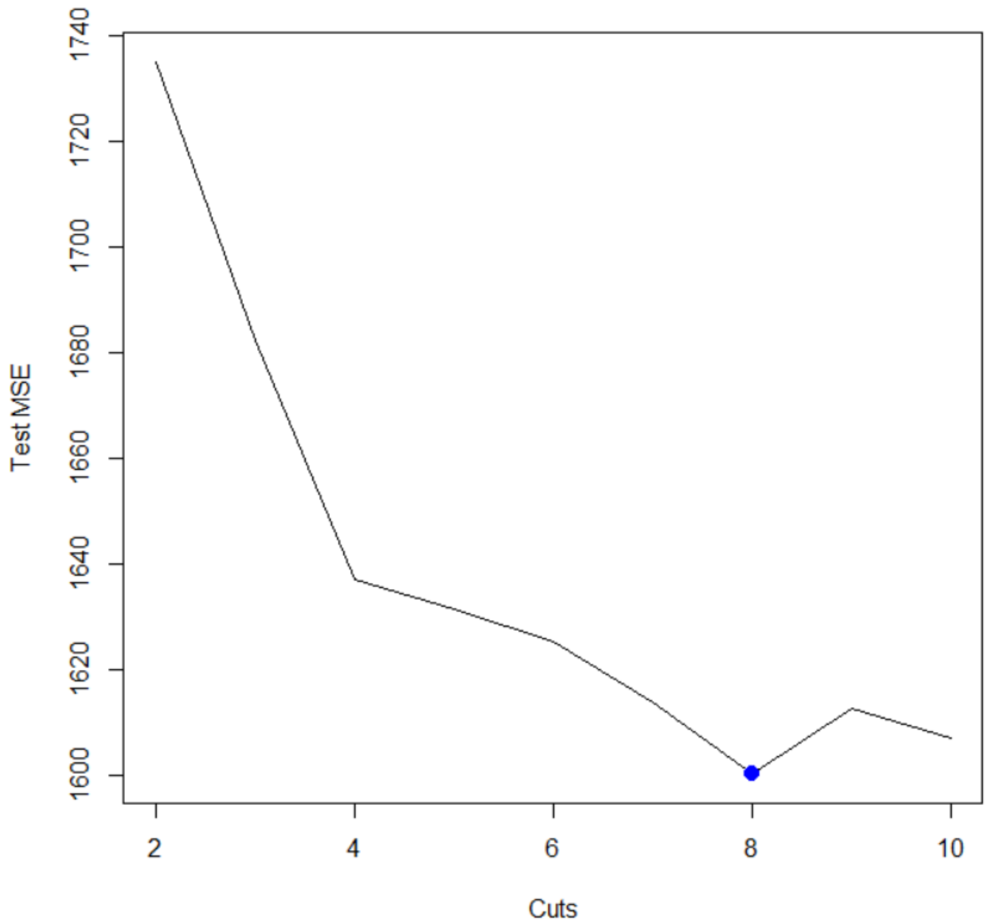
```
Analysis of Variance Table

Model  1: wage ~ age
Model  2: wage ~ poly(age, 2)
Model  3: wage ~ poly(age, 3)
Model  4: wage ~ poly(age, 4)
Model  5: wage ~ poly(age, 5)
Model  6: wage ~ poly(age, 6)
Model  7: wage ~ poly(age, 7)
Model  8: wage ~ poly(age, 8)
Model  9: wage ~ poly(age, 9)
Model 10: wage ~ poly(age, 10)
   Res.Df     RSS Df Sum of Sq        F    Pr(>F)    
1    2998 5022216                                    
2    2997 4793430  1    228786 143.7638 < 2.2e-16 ***
3    2996 4777674  1     15756   9.9005  0.001669 ** 
4    2995 4771604  1      6070   3.8143  0.050909 .  
5    2994 4770322  1      1283   0.8059  0.369398    
6    2993 4766389  1      3932   2.4709  0.116074    
7    2992 4763834  1      2555   1.6057  0.205199    
8    2991 4763707  1       127   0.0796  0.777865    
9    2990 4756703  1      7004   4.4014  0.035994 *  
10   2989 4756701  1         3   0.0017  0.967529    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



З результатів перевірки гіпотез з використанням ANOVA видно, що зважаючи на p-значення, ми можемо побачити, що поліном 2 і 3 степенів забезпечує найкращий результат. Проте варто зауважити, що поліном 9-го степеня також має p-значення нижче за 0.05, що також демонструє хорошу кореляцію між залежною змінною.

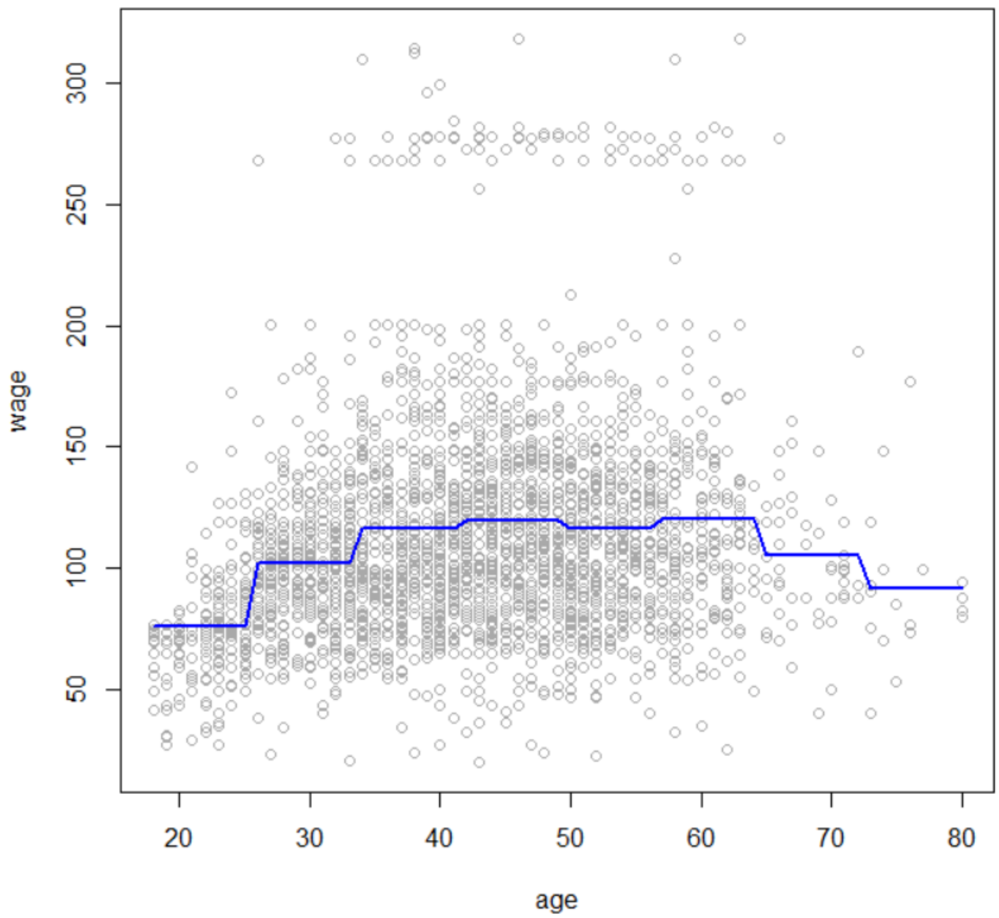
1.2. Використайте східчасту функцію для прогнозування wage за age та проведіть перехресну перевірку для вибору оптимальної кількості розрізів. Побудуйте графік з отриманими результатами.

```
par(mfrow = c(1, 1))
cvs = rep(Inf, 10)

for (i in 2:10) {
    Wage$age.cut = cut(Wage$age, i)
    fit = glm(wage ~ age.cut, data = Wage)
    cvs[i] = cv.glm(Wage, fit, K = 10)$delta[1]
}

cat("\n")
plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")
points(which.min(cvs), cvs[which.min(cvs)],
    col = "blue", cex = 2, pch = 20)
print(paste('Number of cuts for min MSE: ', which.min(cvs)))

PlotFunc(, , TRUE)
```




```
"Number of cuts for min MSE:  8"
```

Результат показує, що тестова помилка буде мінімальною з використанням східчастої функції з 8-ма зрізами.



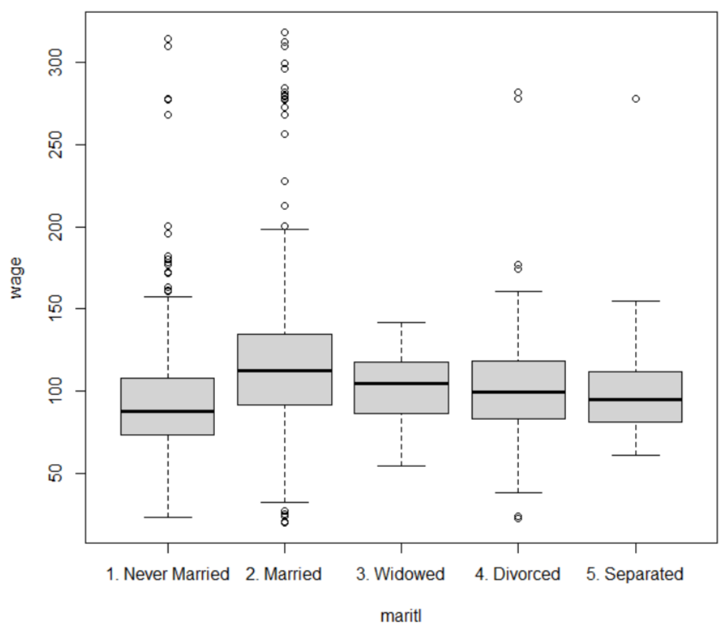
2. Набір даних Wage містить інші змінні такі як, сімейний стан (maritl), робочий клас (jobclass) тощо. Дослідіть зв'язки між деякими з цих інших предикторів та wage, а також використовуючи нелінійні методи пристосуйте гнучкі моделі до даних. Побудуйте графіки отриманих результатів, та підсумуйте свої висновки.

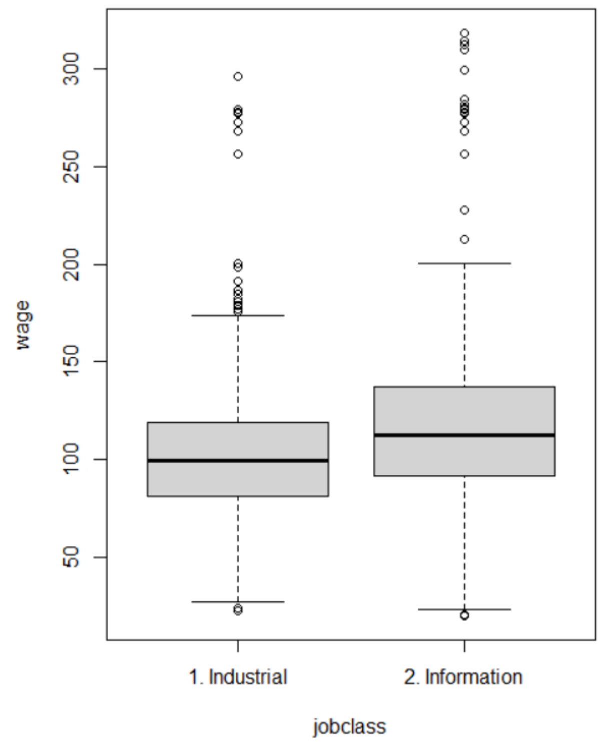
```
par(mfrow = c(1, 2))
plot(Wage$maritl, Wage$wage,  xlab='maritl', ylab='wage')
plot(Wage$jobclass, Wage$wage,  xlab='jobclass', ylab='wage')

library(gam)
fit0 = gam(wage ~ lo(year, span = 0.6) +
s(age, df = 2) + education, data = Wage)
```

```r
fit1 = gam(wage ~ lo(year, span = 0.6) +
s(age, df = 2) + education + jobclass, data = Wage)
fit2 = gam(wage ~ lo(year, span = 0.6) +
s(age, df = 2) + education + maritl, data = Wage)
fit3 = gam(wage ~ lo(year, span = 0.6) +
s(age, df = 2) + education + jobclass + maritl, data = Wage)
print(anova(fit0, fit1, fit2, fit3))

par(mfrow = c(2, 2))
plot(fit2, se = T, col = "blue")
```

```
1. Never Married        2. Married        3. Widowed        4. Divorced
             648              2074                19                204
    5. Separated
              55
 1. Industrial 2. Information
          1544          1456
```

Загальна характеристика якісних змінних maritl та jobclass





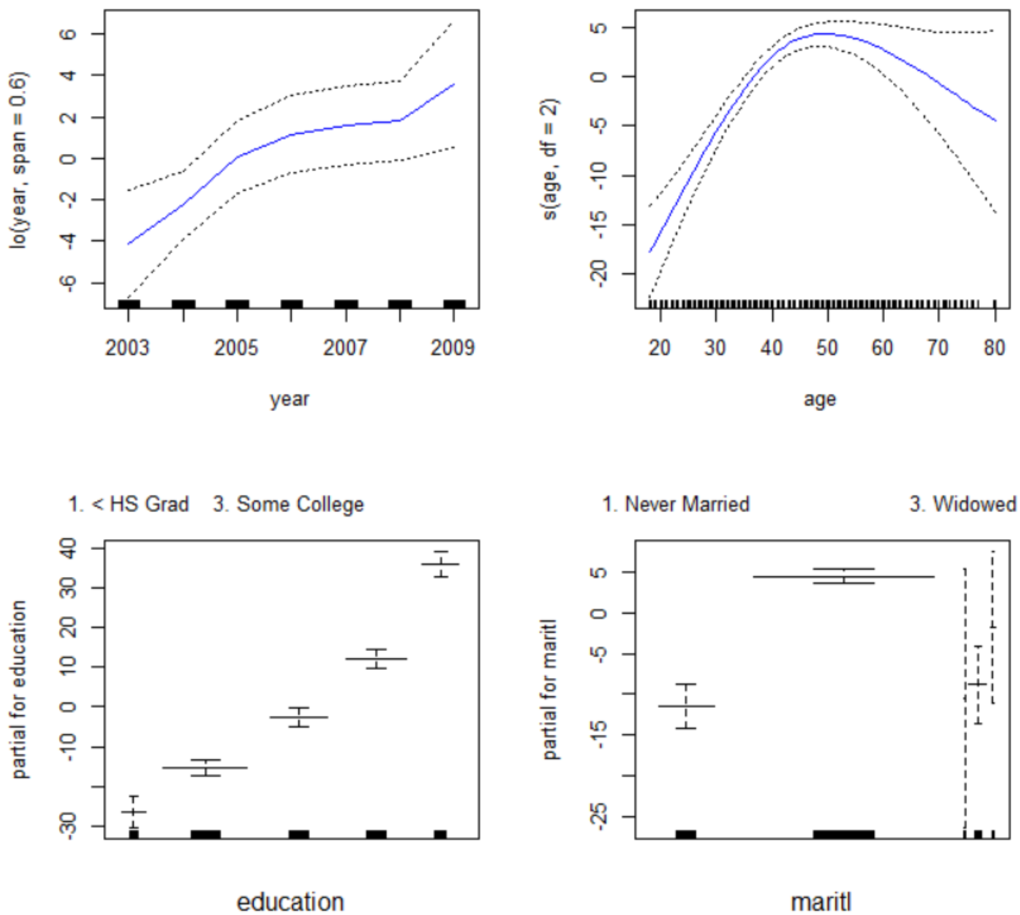
З огляду на візуально представлені зв’язки між змінними maritl, jobclass та залежною wage, бачимо, що подружжя в середньому заробляє більше грошей ніж решта категорій, а також, що працівник в інформаційній сфері в середньому заробляє більше ніж в індустріальній.

```
Analysis of Deviance Table

Model 1: wage ~ lo(year, span = 0.6) + s(age, df = 2) + education
Model 2: wage ~ lo(year, span = 0.6) + s(age, df = 2) + education + jobclass
Model 3: wage ~ lo(year, span = 0.6) + s(age, df = 2) + education + maritl
Model 4: wage ~ lo(year, span = 0.6) + s(age, df = 2) + education + jobclass +
    maritl
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1    2990.1    3723735
2    2989.1    3711032  1    12703  0.001180 **
3    2986.1    3618109  3    92923 < 2.2e-16 ***
4    2985.1    3603843  1    14266  0.000587 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Використавши різні нелінійні методи було пристосовано гнучкі моделі до даних і зроблено перевірку гіпотез з використанням ANOVA.

Для проектування моделей було використано згладжувальні сплайни з функцією s(), з бібліотеки gam. Вказуємо всюди, що функція від age повинна мати

2 ступеня свободи. Також використано всюди локальну регресію для змінної year з інтервалом 0.6. Різниця між моделями полягає у використанні різних додаткових якісних змінних для прогнозування нашої залежної змінної wage.

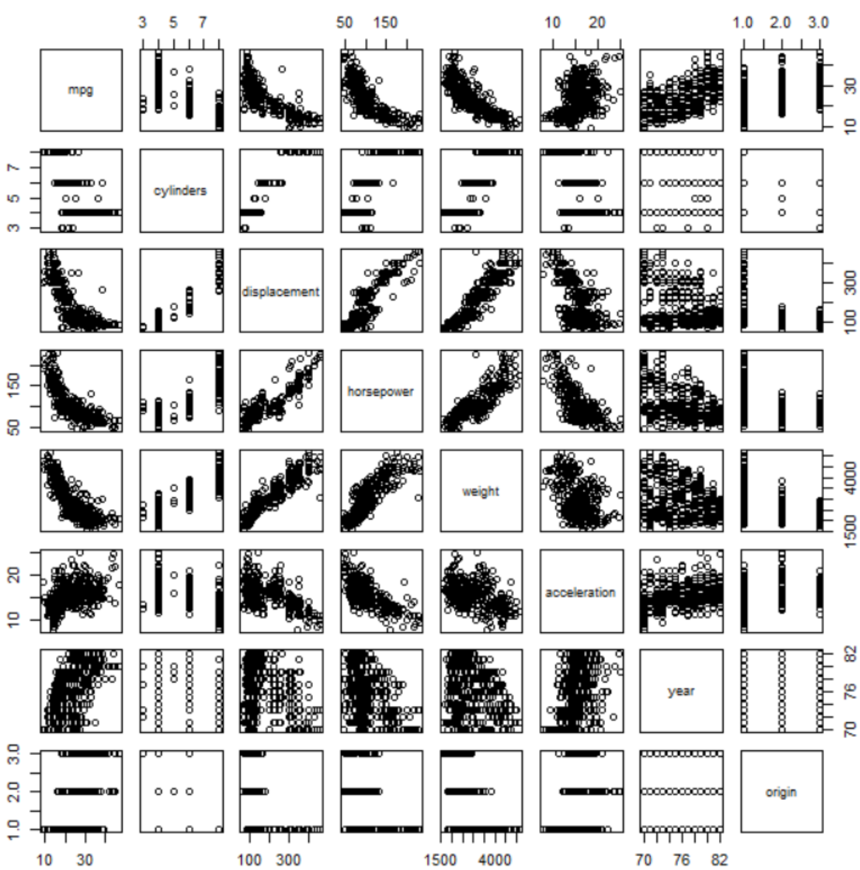


У висновку бачимо, що для прогнозування найкраще підходить третя модель з використанням двох якісних змінних education та maritl.

3. Пристосуйте деякі нелінійні моделі на наборі даних Auto. Чи є якісь докази нелінійних взаємозв'язків в цьому наборі даних? Побудуйте кілька інформативних графіків, щоб обґрунтувати свою відповідь.

Дослідимо змінні та їх залежності з датасету Auto.Для початку використано функцію cor(), щоб побачити настільки сильна чи слаба кореляція між змінними. А також за допомогою функції pairs() виведено графічну залежність всіх змінних.

```
                    mpg  cylinders displacement horsepower     weight
mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
             acceleration       year     origin
mpg             0.4233285  0.5805410  0.5652088
cylinders      -0.5046834 -0.3456474 -0.5689316
displacement   -0.5438005 -0.3698552 -0.6145351
horsepower     -0.6891955 -0.4163615 -0.4551715
weight         -0.4168392 -0.3091199 -0.5850054
acceleration    1.0000000  0.2903161  0.2127458
year            0.2903161  1.0000000  0.1815277
origin          0.2127458  0.1815277  1.0000000
```

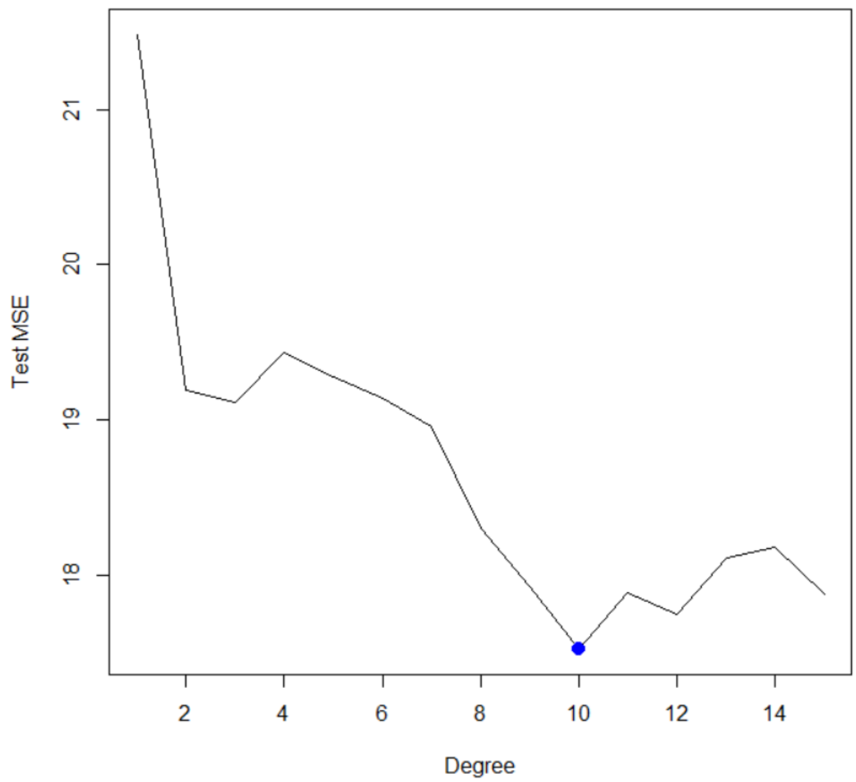


Матриця діаграм розсіювання для даних Auto

Розглядаючи mpg як залежну змінну бачимо, що вона має досить сильну залежність між змінними cylinders, displacement, horsepower та weight, але тут варто наголосити що це є від'ємна кореляція.

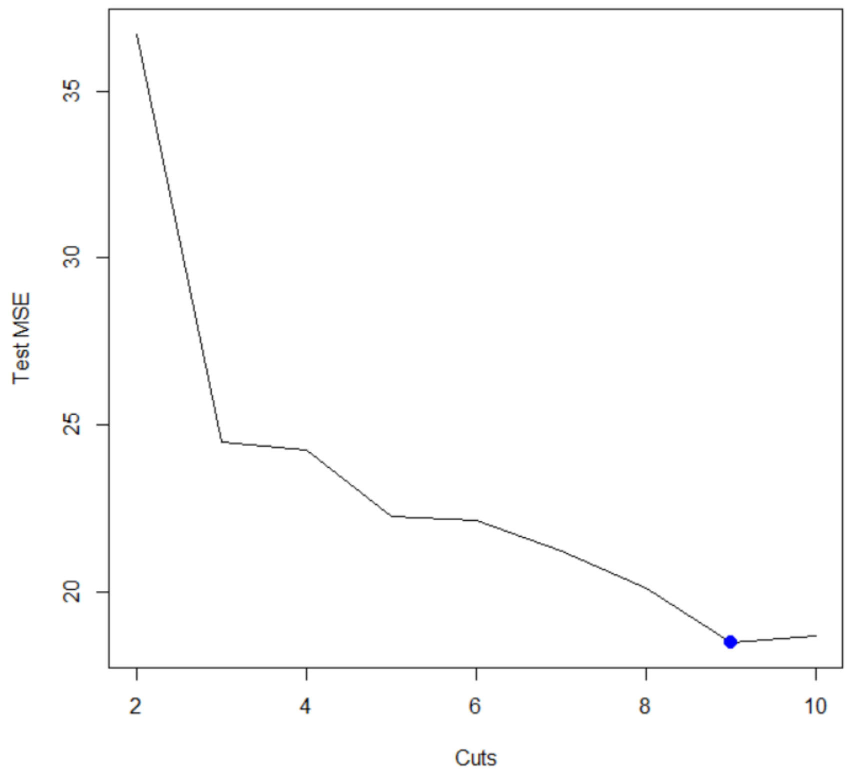
```r
deltas = rep(0, 15)
for (i in 1:15) {
    fit = glm(mpg ~ poly(displacement, i), data = autos)
    deltas[i] = cv.glm(autos, fit, K = 10)$delta[1]
}
plot(1:15, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")
points(which.min(deltas), deltas[which.min(deltas)],
 col = "blue", cex = 2, pch = 20)
print(paste('Polynom power for min MSE: ', which.min(deltas)))
cat("\n")

cvs = rep(Inf, 10)
for (i in 2:10) {
    autos$dis.cut = cut(displacement, i)
    fit = glm(mpg ~ dis.cut, data = autos)
    cvs[i] = cv.glm(autos, fit, K = 10)$delta[1]
}
plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")
points(which.min(cvs), cvs[which.min(cvs)],
 col = "blue", cex = 2, pch = 20)
print(paste('Number of cuts for min MSE: ', which.min(cvs)))
cat("\n")

cvs = rep(Inf, 10)
for (i in 3:10) {
    fit = glm(mpg ~ ns(displacement, df = i), data = autos)
    cvs[i] = cv.glm(autos, fit, K = 10)$delta[1]
}
plot(3:10, cvs[-c(1, 2)], xlab = "Degree", ylab = "Test MSE", type = "l")
points(which.min(cvs), cvs[which.min(cvs)],
col = "blue", cex = 2, pch = 20)
```

```
print(paste('Degrees of freedom for min MSE [splines]: ', which.min(cvs)))
cat("\n")

fit = gam(mpg ~ s(displacement, df = 2) +
 s(horsepower, df = 2), data = autos)
print(summary(fit))
```



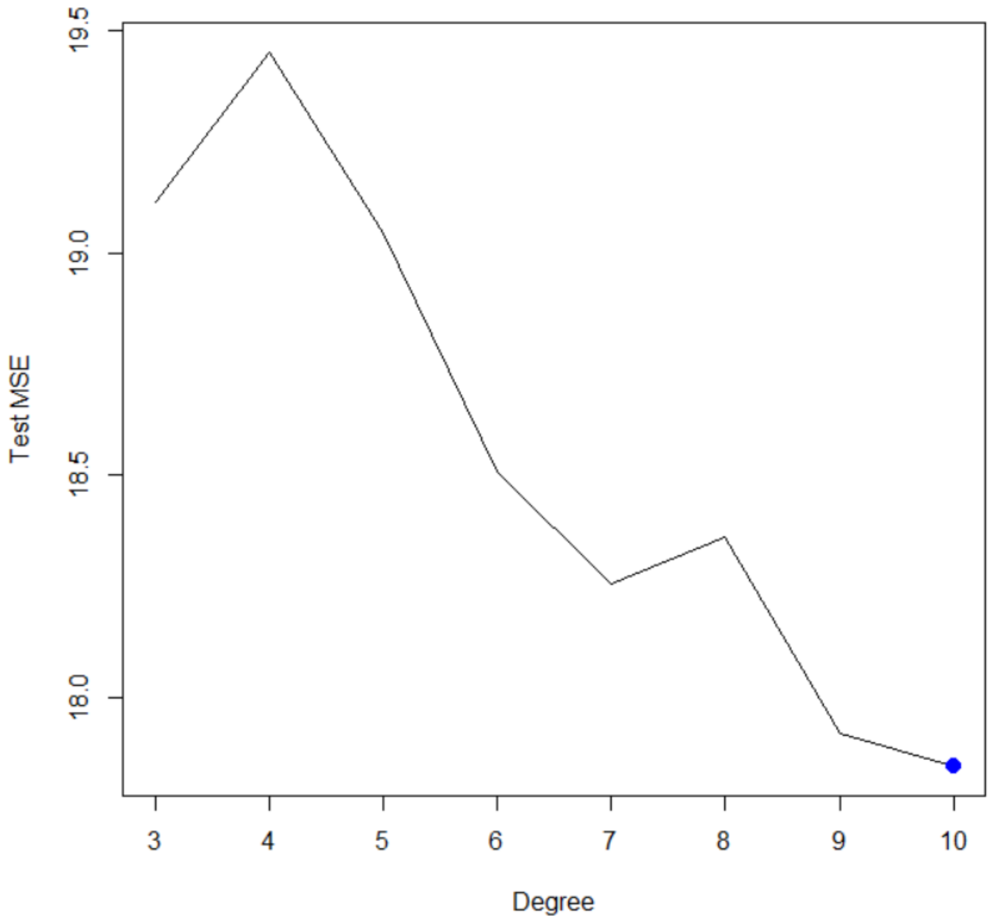

```
"Polynom power for min MSE:  10"
```

З огляду на графік наведений нижче можемо побачити, що d=10 є оптимальним степенем для полінома в сенсі тестової помилки.




```
"Number of cuts for min MSE:  9"
```

Зважаючи на графік наведений нижче можемо побачити, що мінімальна тестова помилка буде досягатися для східчастої функції з 9-ма зрізами.




```
"Degrees of freedom for min MSE [splines]:  10"
```

Зважаючи на графік наведений нижче, можемо побачити, що тестова похибка мінімальна для 10 ступенів свободи з використанням природного сплайну для змінної displacement.

```
Anova for Parametric Effects
                         Df  Sum Sq Mean Sq F value    Pr(>F)
s(displacement, df = 2)   1 15210.3 15210.3 953.995 < 2.2e-16 ***
s(horsepower, df = 2)     1   837.8   837.8  52.547 2.299e-12 ***
Residuals               387  6170.3    15.9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Anova for Nonparametric Effects
                        Npar Df Npar F     Pr(F)
(Intercept)
s(displacement, df = 2)       1 46.025 4.389e-11 ***
s(horsepower, df = 2)         1 40.816 4.813e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Для проектування УАМ моделі було використано згладжувальні сплайни з функцією s(). Вказуємо, що функція від displacement та horsepower повинна мати 2 ступеня свободи. І загалом модель показує, що всі обидві змінні є значущі для моделювання mpg.

4. Використаємо змінні dis (зважене середнє відстані до п'яти центрів зайнятості в Бостоні) та nox (концентрація оксидів азоту у частинах на 10 мільйонів) з набору даних Boston. Розглянемо dis як предиктор та nox як залежну змінну.

```
      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
```

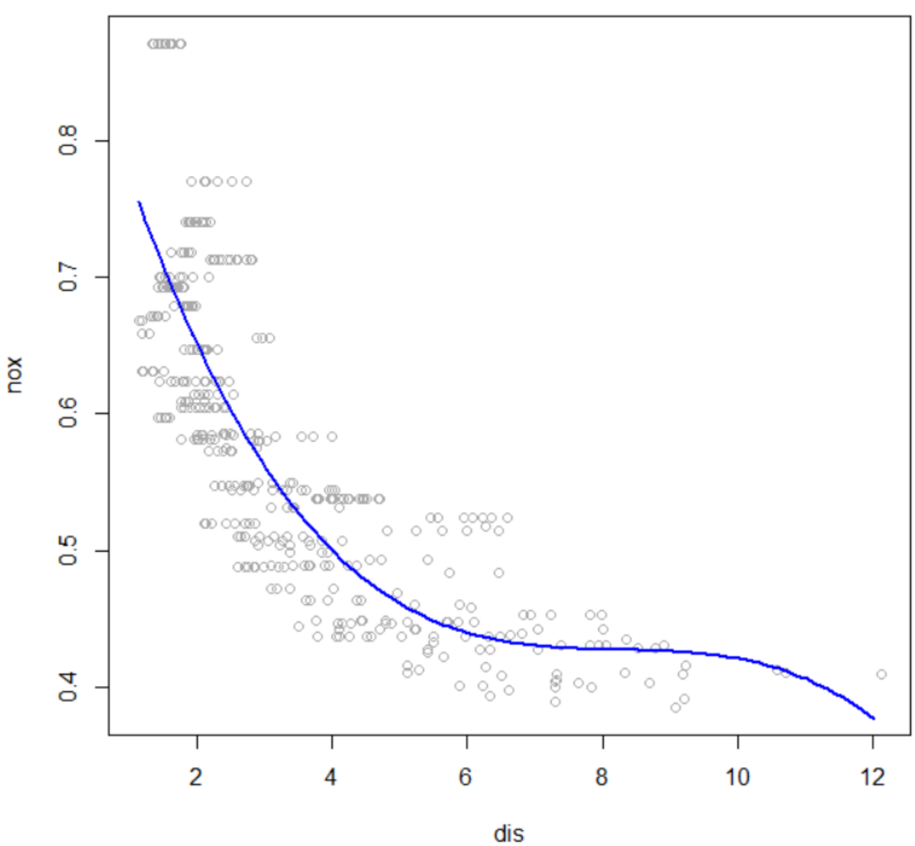
Характеристика даних Boston

4.1. Використовуючи функцію poly(), встановіть кубічну поліноміальну регресію для передбачення nox за допомогою dis. Опишіть результати регресії та побудуйте графік даних та поліноміальної регресії.

```
fit = lm(nox ~ poly(dis, 3), data = Boston)
print(summary(fit))

dis.grid = seq(from = range(Boston$dis)[1],
 to = range(Boston$dis)[2], by = 0.1)
preds = predict(fit, list(dis = dis.grid))

plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, preds, col = "blue", lwd = 2)
```

```
Call:
lm(formula = nox ~ poly(dis, 3), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.121130 -0.040619 -0.009738  0.023385  0.194904 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06207 on 502 degrees of freedom
Multiple R-squared:  0.7148,	Adjusted R-squared:  0.7131 
F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

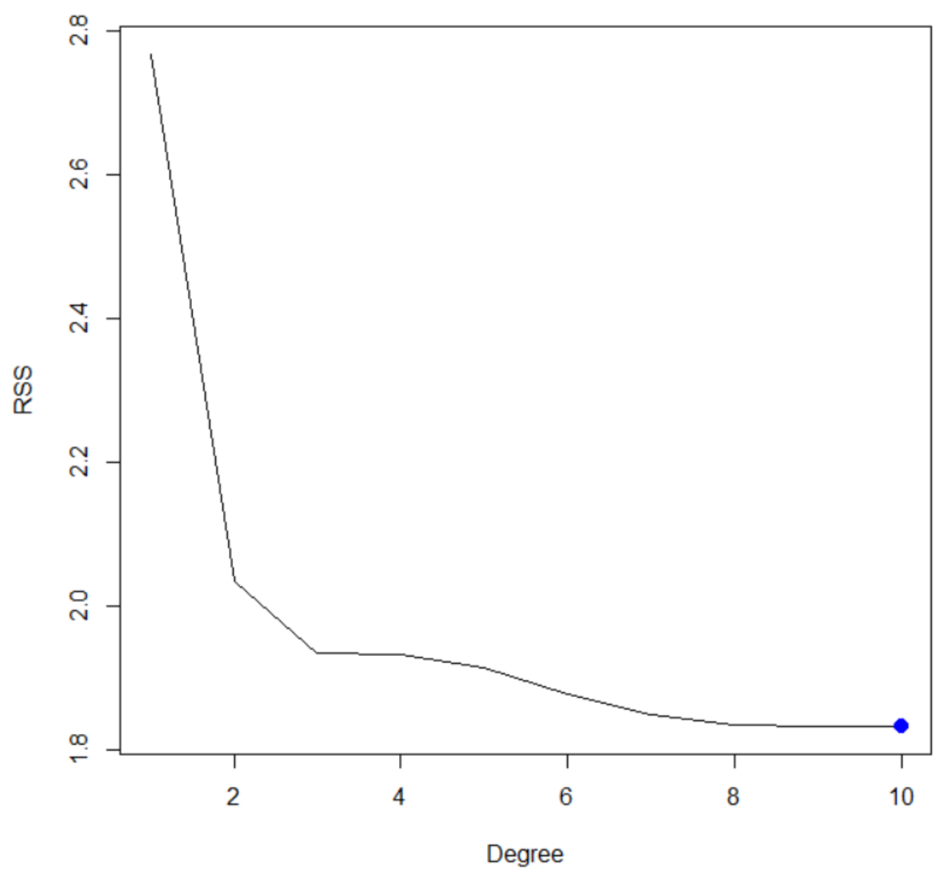
Результат аналізу нашої регресії показав, що всі степені кубічного поліному є значущими для залежної змінної nox.



З графіка даних бачимо обернену залежність наших змінних та графік поліноміальної регресії.

4.2. Побудуйте поліноміальні моделі для різних степенів (скажімо, від 1 до 10), і наведіть їхні RSS.

```
rss = rep(0, 10)
for (i in 1:10) {
    fit = lm(nox ~ poly(dis, i), data = Boston)
    rss[i] = sum(fit$residuals^2)
}
plot(1:10, rss, xlab = "Degree", ylab = "RSS", type = "l")
points(which.min(rss), rss[which.min(rss)],
 col = "blue", cex = 2, pch = 20)
print(paste('Polynom power for min RSS: ', which.min(rss)))
```

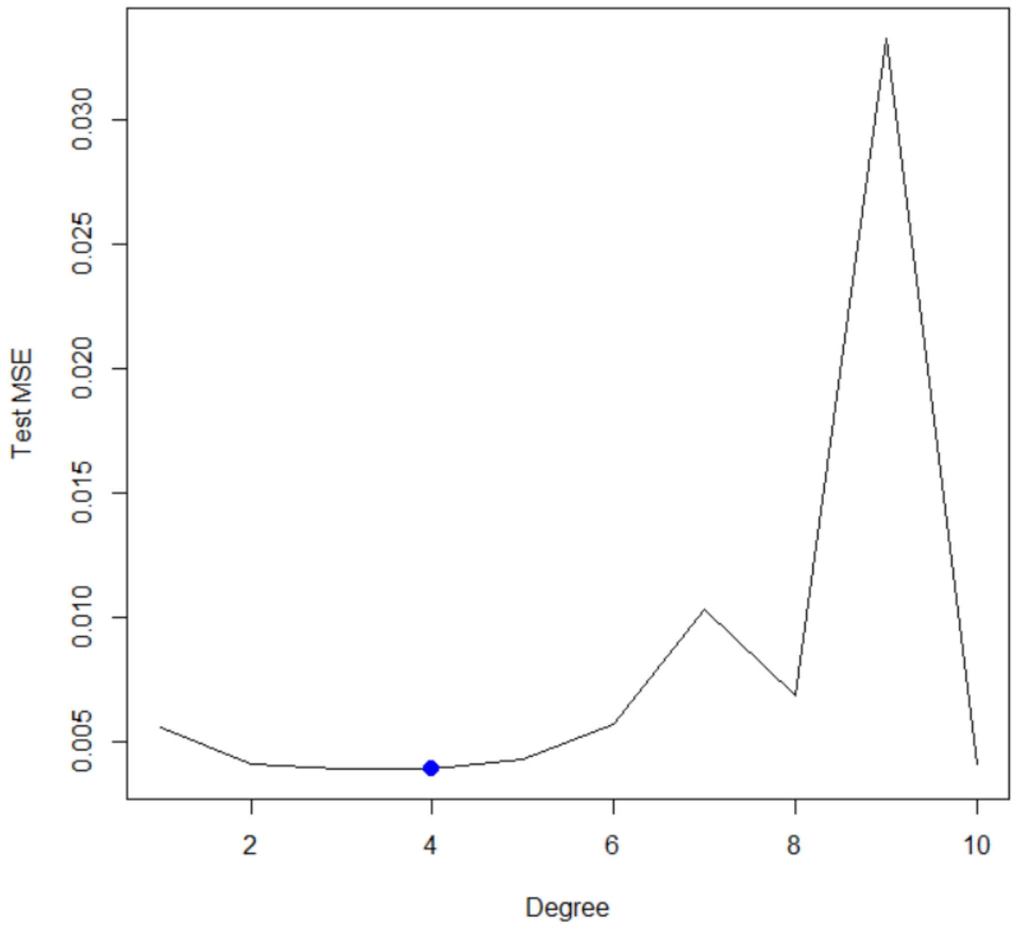


```
"Polynom power for min RSS:  10"
```

Можна побачити, що RSS (residual sum of squares) зменшується зі збільшенням степеня полінома, відповідно для 10-го степеня RSS буде мінімальним.

4.3. Використайте перехресну перевірку або інший підхід для вибору оптимального степеня для поліноміальної регресії та поясніть отримані результати.

```
deltas = rep(0, 10)
for (i in 1:10) {
    fit = glm(nox ~ poly(dis, i), data = Boston)
    deltas[i] = cv.glm(Boston, fit, K = 10)$delta[1]
}
plot(1:10, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")
points(which.min(deltas), deltas[which.min(deltas)],
 col = "blue", cex = 2, pch = 20)
print(paste('Polynom power for min MSE: ', which.min(deltas)))
```




```
"Polynom power for min MSE:  4"
```

З результатів перехресної перевірки найменше MSE досягається при степені полінома чотири і досить різко зростає для 9-го степеня.

4.4. Використовуючи функцію bs (), пристосуйте сплайн регресію для прогнозування nox за допомогою dis. Опишіть результати отримані з використанням чотирьох ступенів свободи. Як ви вибрали вузли? Побудуйте графік отриманої моделі.

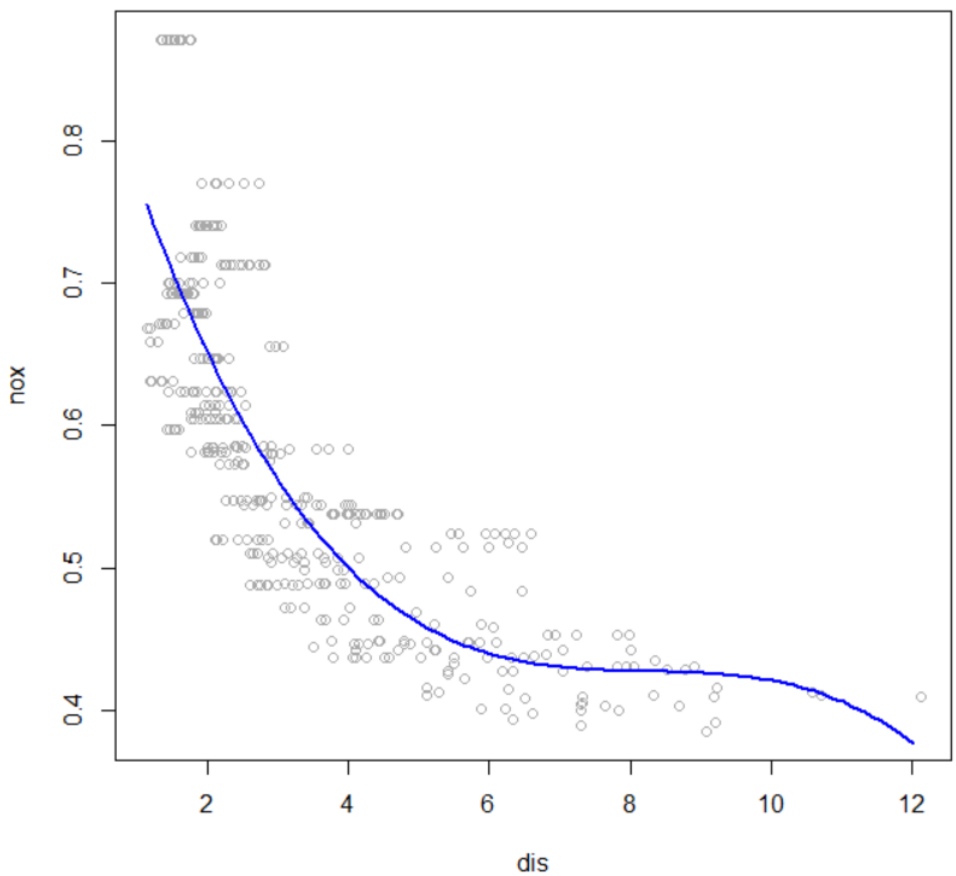
```
fit = lm(nox ~ bs(dis, df = 4, knots = c(4, 7, 10)), data = Boston)
```

```r
print(summary(fit))


pred = predict(fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, preds, col = "blue", lwd = 2)
```

```
Call:
lm(formula = nox ~ bs(dis, df = 4, knots = c(4, 7, 10)), data = Boston)

Coefficients:
                                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)                             0.73918    0.01332  55.510  < 2e-16 ***
bs(dis, df = 4, knots = c(4, 7, 10))1 -0.08839    0.02506  -3.526  0.00046 ***
bs(dis, df = 4, knots = c(4, 7, 10))2 -0.31363    0.01684 -18.625  < 2e-16 ***
bs(dis, df = 4, knots = c(4, 7, 10))3 -0.27037    0.02791  -9.686  < 2e-16 ***
bs(dis, df = 4, knots = c(4, 7, 10))4 -0.37989    0.03801  -9.995  < 2e-16 ***
bs(dis, df = 4, knots = c(4, 7, 10))5 -0.28983    0.06615  -4.381 1.44e-05 ***
bs(dis, df = 4, knots = c(4, 7, 10))6 -0.32971    0.06324  -5.214 2.71e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06185 on 499 degrees of freedom
Multiple R-squared:  0.7185,	Adjusted R-squared:  0.7151 
F-statistic: 212.3 on 6 and 499 DF,  p-value: < 2.2e-16
```

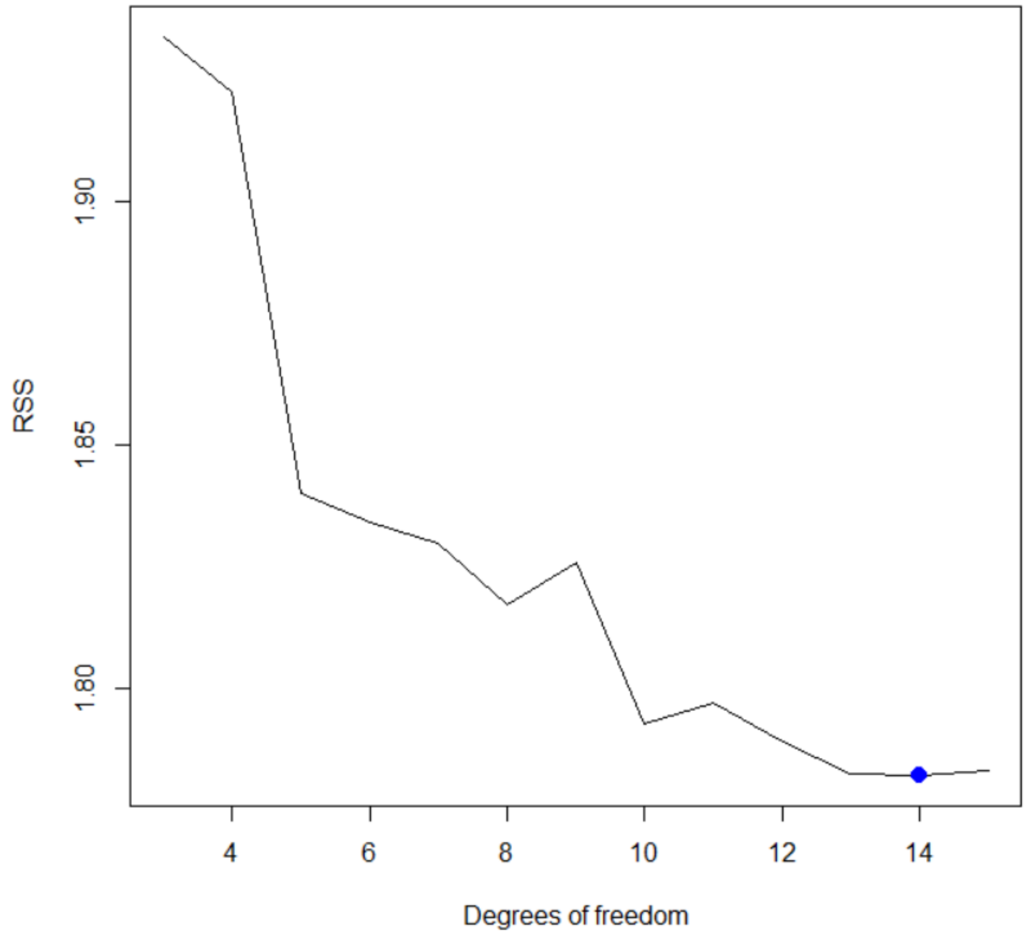
Для того, щоб застосувати регресійні сплайни, ми використовуємо бібліотеку splines. Функція bs() генерує всю матрицю базисних функцій для сплайнів із заданим набором вузлів. Ступінь свободи вказаний як 4, а вузли у dis 4, 7 і 10, що забезпечувало найкращі результати. Також варто додати, що в нас використовується кубічний сплайн.

Результати показують, що всі базові функції є значущі у сплайні, при чому p-значення всюди є нижче ніж 0.001.



4.5. Пристосуйте сплайн регресію для діапазону ступенів свободи, і побудуйте графік результатів. Наведіть відповідні RSS. Опишіть отримані результати.

```
rss = rep(Inf, 15)
for (i in 3:15) {
    fit = lm(nox ~ bs(dis, df = i), data = Boston)
    rss[i] = sum(fit$residuals^2)
}
plot(3:15, rss[-c(1, 2)], xlab = "Degrees of freedom", ylab = "RSS", type = "l")
points(which.min(rss), rss[which.min(rss)],
 col = "blue", cex = 2, pch = 20)
print(paste('Degrees of freedom for min RSS: ', which.min(rss)))
```



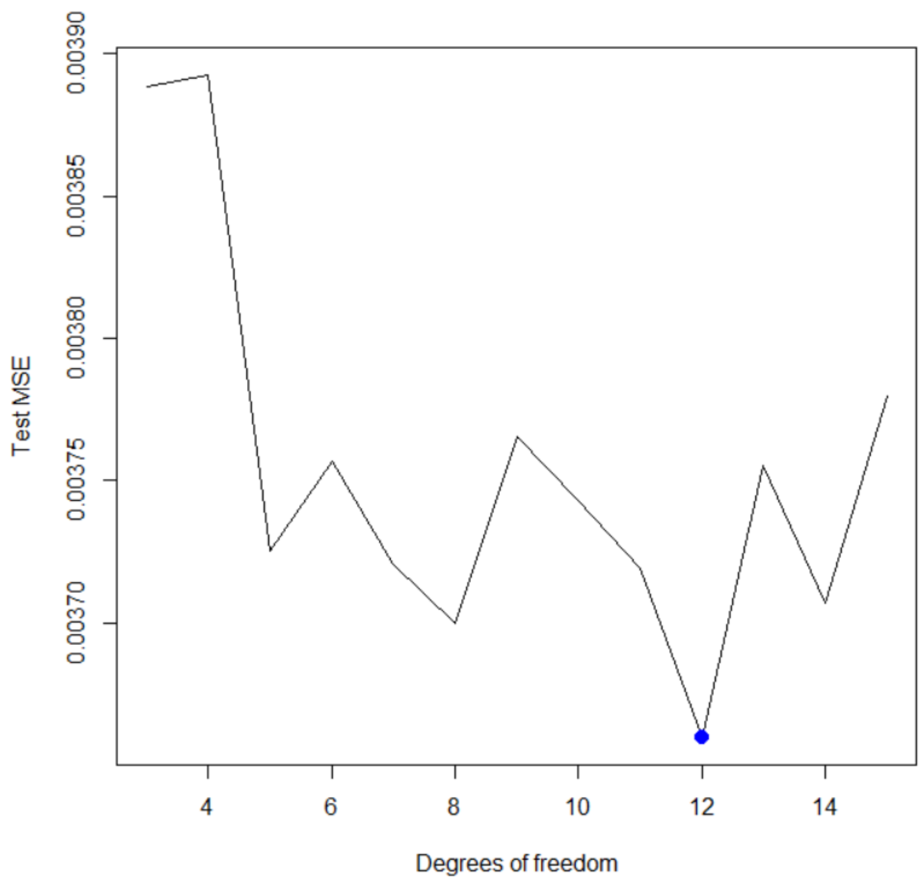

```
"Degrees of freedom for min RSS:  14"
```

Можемо зауважити, що RSS спадає до 14 ступеня свободи, а потім починає зростати.

4.6. Використайте перехресну перевірку або інший підхід, щоб вибрати найкращий ступінь свободи для сплайн регресії на цих даних. Опишіть свої результати.

```
cv = rep(Inf, 15)
for (i in 3:15) {
    fit = glm(nox ~ bs(dis, df = i), data = Boston)
```

```
    cv[i] = cv.glm(Boston, fit, K = 10)$delta[1]
}
plot(3:15, cv[-c(1, 2)], xlab = "Degrees of freedom", ylab = "Test MSE", type = "l")
points(which.min(cv), cv[which.min(cv)],
 col = "blue", cex = 2, pch = 20)
print(paste('Degrees of freedom for min MSE: ', which.min(cv)))
```




```
"Degrees of freedom for min MSE:  12"
```

Результати перехресної перевірки із значенням k=10 показують, що тестова помилка мінімальна для 12-ти ступенів свободи для сплайн регресії. Для більших степенів вона зростає.

5. Використайте набір даних College.

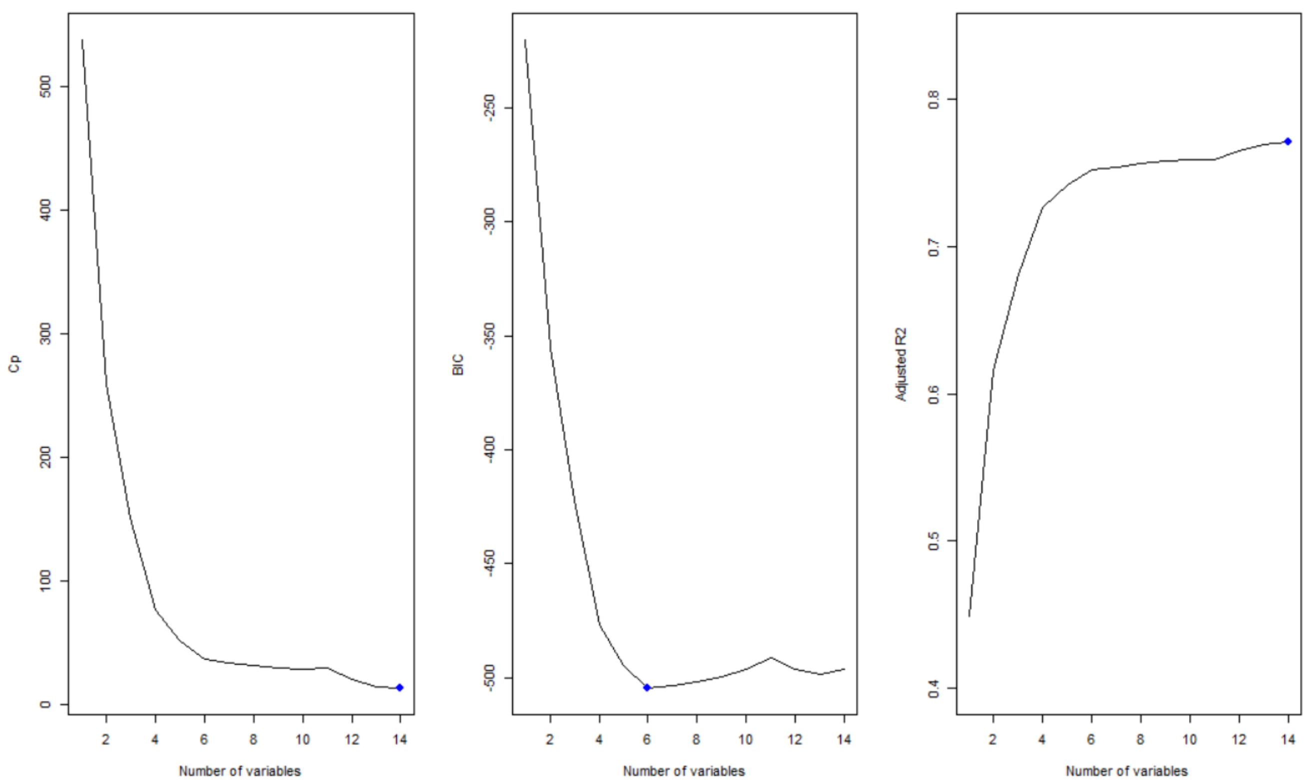
```
 Private        Apps           Accept          Enroll       Top10perc
 No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
 Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
   Room.Board       Books           Personal         PhD
 Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
 1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
 Median :4200   Median : 500.0   Median :1200   Median : 75.00
 Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
 3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
 Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
    Terminal       S.F.Ratio      perc.alumni        Expend
 Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
 1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
 Median : 82.0   Median :13.60   Median :21.00   Median : 8377
 Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
 3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
 Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
   Grad.Rate
 Min.   : 10.00
 1st Qu.: 53.00
 Median : 65.00
 Mean   : 65.46
 3rd Qu.: 78.00
 Max.   :118.00
```

Характеристика даних College

5.1. Розбийте дані на навчальний та тестовий набори. Використайте Outstate як залежну змінну, а інші змінні як предиктори. Виконайте покроковий вибір вперед на навчальному наборі, щоб визначити задовільну модель, яка використовує лише підмножину предикторів.

```
train = sample(length(Outstate), length(Outstate) / 2)

College.train = College[train, ]
College.test = College[-train, ]

fit = regsubsets(Outstate ~ ., data = College.train,
  nvmax = 14, method = "forward")
fit.summary = summary(fit)

par(mfrow = c(1, 3))
plot(fit.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
```

```
points(which.min(fit.summary$cp), fit.summary$cp[which.min(fit.summary$cp)],
col = "blue", cex = 2, pch = 20)

plot(fit.summary$bic, xlab = "Number of variables", ylab = "BIC", type='l')
points(which.min(fit.summary$bic), fit.summary$bic[which.min(fit.summary$bic)],
col = "blue", cex = 2, pch = 20)

plot(fit.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2",
 type = "l", ylim = c(0.4, 0.84))
points(which.max(fit.summary$adjr2), fit.summary$adjr2[which.max(fit.summary$adjr2)],
col = "blue", cex = 2, pch = 20)

fit = regsubsets(Outstate ~ ., data = College, method = "forward")
coeffs = coef(fit, id = 6)
cat("\n")
print(names(coeffs))
```

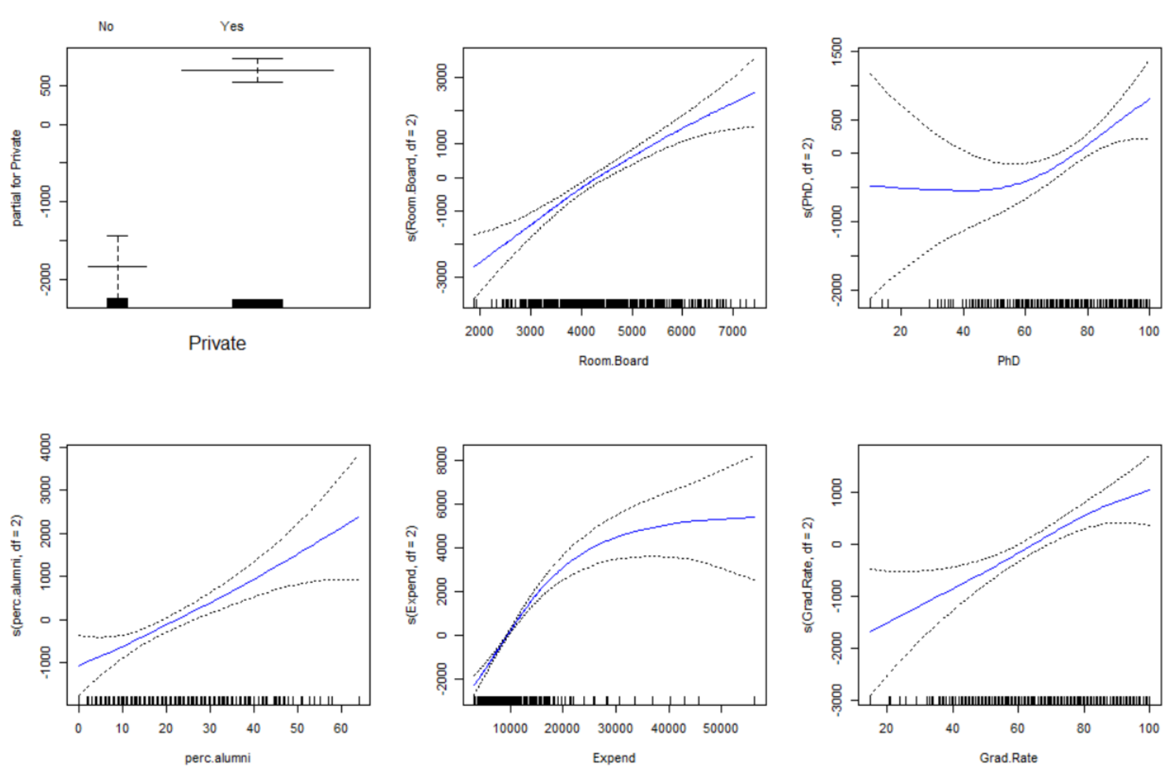


Як можна бачити з графіків методи Cp, BIC і скорегований R^2 показують різні результати, проте BIC якраз визначив на навчальному наборі задовільну модель, яка використовує лише 6 предикторів, що і буде мінімальним розміром підножини предикторів.

```
"(Intercept)" "PrivateYes"   "Room.Board"   "PhD"          "perc.alumni"
"Expend"      "Grad.Rate"
```

5.2. Оцініть УАМ модель на навчальних даних, використовуючи Outstate як залежну змінну та ознаки обрані на попередньому кроці як предиктори. Побудуйте графік результатів та поясніть свої висновки.

```
fit = gam(Outstate ~ Private + s(Room.Board, df = 2) +
s(PhD, df = 2) + s(perc.alumni, df = 2) +
 s(Expend, df = 2) +
  s(Grad.Rate, df = 2),
  data=College.train)
par(mfrow = c(2, 3))
plot(fit, se = T, col = "blue")
```



З графіка бачимо пряму залежність змінних та згладжувальних сплайнів, які використовувалися для прогнозування змінної Outstate.

5.3. Застосуйте модель на тестовому наборі даних, поясніть отримані результати.

```
preds = predict(fit, College.test)

err = mean((College.test$Outstate - preds)^2)
print(paste("Test error :", round(err, 2)))

tss = mean((College.test$Outstate - mean(College.test$Outstate))^2)
rss = 1 - err / tss
print(paste("RSS :", round(rss, 2)))
```

```
"Test error : 3438191.98"
"RSS : 0.76"
```

Ми отримуємо R^2 0,766 для тестових даних, використовуючи УАМ з 6 предикторами.

5.4. Для яких змінних, якщо такі є, є докази нелінійності взаємозв'язку з залежною змінною?

```
Anova for Nonparametric Effects
                        Npar Df Npar F     Pr(F)    
(Intercept)                                         
Private                                             
s(Room.Board, df = 2)         1  2.005   0.15763    
s(PhD, df = 2)                1  3.781   0.05258 .  
s(perc.alumni, df = 2)        1  0.314   0.57572    
s(Expend, df = 2)             1 47.156 2.725e-11 ***
s(Grad.Rate, df = 2)          1  1.057   0.30446    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ANOVA показує чіткий нелінійний зв'язок між Outstate та Expend. Також бачимо слабкий нелінійний зв'язок між Outstate та PhD.

6. УАМ, як правило, оцінюють на основі методу підгонки. Розглянемо цей метод на основі множинної лінійної регресії. Припустимо, що ми хочемо використати множинну лінійну регресію, але у нас немає програмного забезпечення для цього. Натомість ми маємо лише програмне забезпечення для оцінки простої лінійної регресії. Тому ми використаємо наступний ітераційний підхід: ми фіксуємо всі оцінки коефіцієнтів, крім одного на поточному значенні та оновлюємо лише одну оцінку коефіцієнта за допомогою простої лінійної регресії. Процес продовжується поки не досягнеться збіжність, тобто поки оцінки коефіцієнтів не перестануть змінюватися. Застосуємо цей підхід на штучному прикладі.

6.1. Згенеруйте залежну змінну *Y* і два предиктори *X*1 і *X*2, з *n* = 100.

```
set.seed(1)

X1 = rnorm(100)
X2 = rnorm(100)
eps = rnorm(100, sd = 0.1)

betas = runif(3, min=-5, max=5)
cat("\n")
print("Beta-values list:")
print(betas)

Y = betas[1] + betas[2] * X1 +
 betas[3] * X2 + eps
```

Як можна побачити визначення значень β_i відбувається випадково (seed = 1), в межах від -5 до 5 будь-які дробові числа.

```
"Beta-values list:"
 3.142518  4.287772 -3.525190
```

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$$

6.2. Ініціалізуйте оцінку *β*1 довільним значенням на свій вибір.

```
beta0 = rep(0, 1000)
beta1 = rep(0, 1000)
beta2 = rep(0, 1000)
beta1[1] = runif(1, min=-5, max=5)
```

```
"Beta1[1] :  2.49821664998308"
```

6.3. Не змінюючи *β*1 оцініть модель

$$\boldsymbol{Y - \beta 1 X1 = \beta 0 + \beta 2 X2 + \varepsilon.}$$

Це можна зробити наступним чином:

> a=y-beta1 *x1

> beta2=lm(a~x2)$coef [2]

6.4. Зафіксувавши оцінку *β*2, оцініть модель

$$\boldsymbol{Y - \beta 2 X2 = \beta 0 + \beta 1 X1 + \varepsilon.}$$

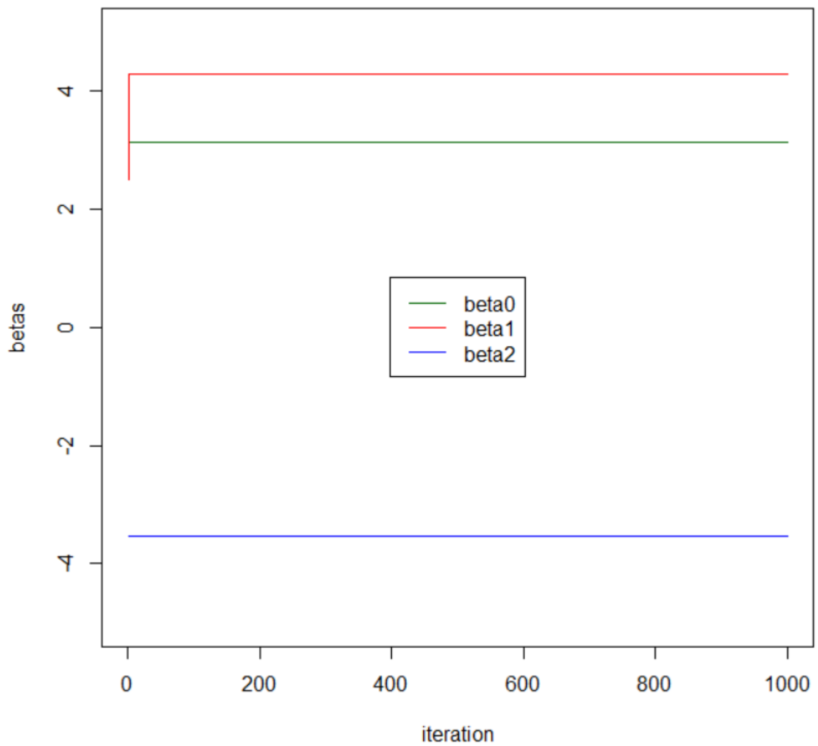
Це можна зробити наступним чином:

> a=y-beta2 *x2

> beta1=lm(a~x1)$coef [2]

6.5. Використайте for для організації циклу з повторень кроків 6.3 та 6.4 1,000 разів. Наведіть оцінки параметрів $\beta 0$, $\beta 1$ і $\beta 2$ на кожній ітерації. Побудуйте графіки на яких відображено ці значення для $\beta 0$, $\beta 1$ і $\beta 2$ різними кольорами.

```r
for (i in 1:1000) {
    a = Y - beta1[i] * X1
    beta2[i] = lm(a ~ X2)$coef[2]

    a = Y - beta2[i] * X2
    lm.fit = lm(a ~ X1)

    if (i < 1000) {
        beta1[i + 1] = lm.fit$coef[2]
    }

    beta0[i] = lm.fit$coef[1]
}

plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas",
    ylim = c(-5, 5), col = "darkgreen")
lines(1:1000, beta1, col = "red")
lines(1:1000, beta2, col = "blue")
legend("center", c("beta0", "beta1", "beta2"),
    col = c("darkgreen", "red", "blue"))
```

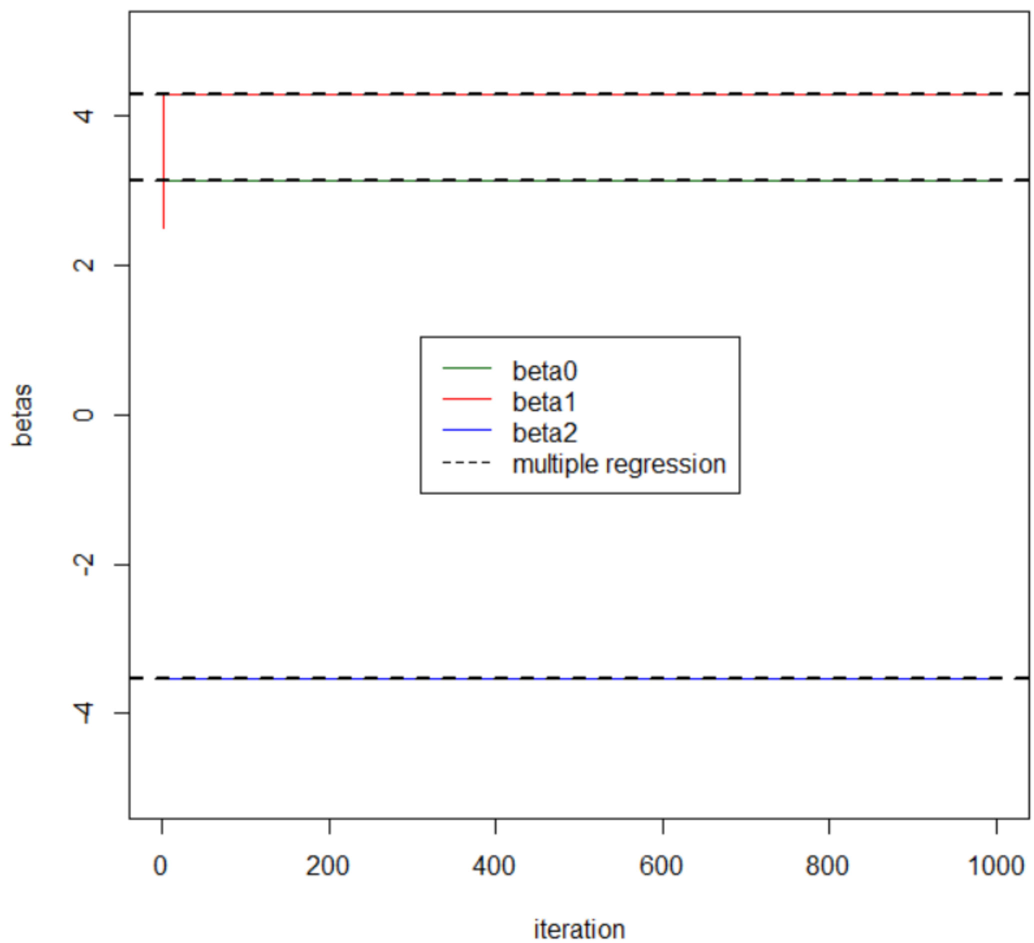


Як бачимо коефіцієнти швидко досягають своїх справжніх значень. Тільки для β1 в нас не є постійно горизонтальна пряма, оскільки ми ініціалізували оцінку β1 і від неї проводили подальші дії.

6.6. Порівняйте результати 6.5 з результатами оцінки множинної регресії *Y* на *X*1 і *X*2. Використайте функцію abline() для накладання цих значень на графік отриманий в 6.5.

```
lm.fit = lm(Y ~ X1 + X2)

plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas",
 ylim = c(-5, 5), col = "darkgreen")
lines(1:1000, beta1, col = "red")
lines(1:1000, beta2, col = "blue")

abline(h = lm.fit$coef[1], lty = "dashed", lwd = 2, col = "black")
abline(h = lm.fit$coef[2], lty = "dashed", lwd = 2, col = "black")
```

```
abline(h = lm.fit$coef[3], lty = "dashed", lwd = 2, col = "black")

legend("center", c("beta0", "beta1", "beta2", "multiple regression"), lty = c(1,
    1, 1, 2), col = c("darkgreen", "red", "blue", "black"))
```



Пунктирні лінії на графіку показують, що коефіцієнти множинної регресії збігаються з коефіцієнтами отриманими в минулому пункті за допомогою backfitting.

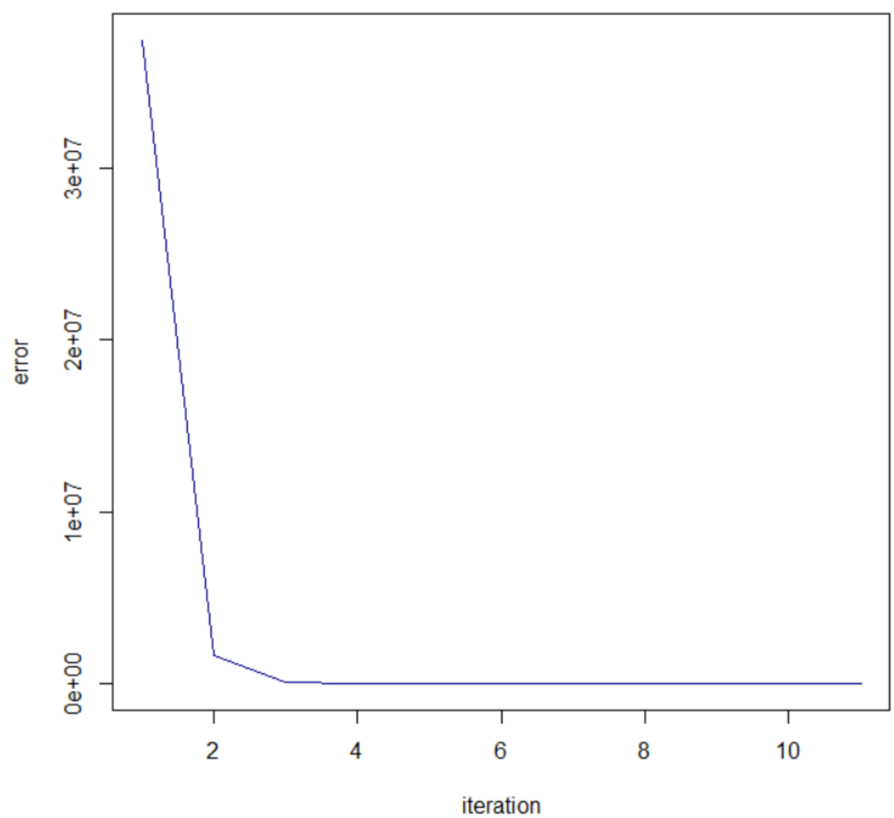
6.7. Для цього набору даних, скільки ітерацій підгонки потрібно було для отримання «доброго» наближення до оцінок коефіцієнтів множинної регресії?

Коли зв’язок між залежною змінною та предикторами є лінійний, однієї ітерації достатньо для досягнення хорошого наближення до реальних значень коефіцієнтів регресії.

7. Продовжимо розгляд попереднього прикладу. Покажіть, що у випадку *p* = 100 можна отримати оцінки коефіцієнтів множинної регресії повторно застосовуючи метод підгонки. Скільки ітерацій підгонки потрібно було для отримання «доброго» наближення до оцінок коефіцієнтів множинної регресії? Побудуйте графік для підтвердження своїх висновків.

```
set.seed(1)

p = 100
n = 1000
x = matrix(ncol = p, nrow = n)

coefi = rep(0, p)

for (i in 1:p) {
    x[, i] = rnorm(n)
    coefi[i] = rnorm(1) * 100
}

y = x %*% coefi + rnorm(n)

beta = rep(0, p)
max_iterations = 1000
errors = rep(0, max_iterations + 1)
iter = 2
errors[1] = Inf
errors[2] = sum((y - x %*% beta)^2)
threshold = 1e-04
cat("\n")
```

```r
while (iter < max_iterations && errors[iter - 1] - errors[iter] > threshold) {
    for (i in 1:p) {
        a = y - x %*% beta + beta[i] * x[, i]
        beta[i] = lm(a ~ x[, i])$coef[2]
    }
    iter = iter + 1
    errors[iter] = sum((y - x %*% beta)^2)
    print(paste(iter - 2, "  ", round(errors[iter - 1], 4), round(errors[iter], 4)))

    print(paste('Error diff : ', round(errors[iter - 1] - errors[iter], 4)))

}

plot(1:11, errors[3:13], type = "l",
 xlab = "iteration", ylab = "error", col = "darkblue")
```



```
"1    1016122216.2326 37472750.5654"
"Error diff :  978649465.6672"
"2    37472750.5654 1669889.4203"
"Error diff :  35802861.145"
"3    1669889.4203 77923.7545"
"Error diff :  1591965.6659"
"4    77923.7545 6157.4249"
"Error diff :  71766.3295"
"5    6157.4249 1277.0458"
"Error diff :  4880.3792"
"6    1277.0458 928.3072"
"Error diff :  348.7386"
"7    928.3072 904.7608"
"Error diff :  23.5464"
"8    904.7608 903.2173"
"Error diff :  1.5435"
"9    903.2173 903.1259"
"Error diff :  0.0914"
"10    903.1259 903.1232"
"Error diff :  0.0027"
"11    903.1232 903.1239"
"Error diff :  -7e-04"
```

Десять ітерацій достатньо, щоб отримати хорошу апроксимацію (з точністю до 0.01) визначену пороговим значенням суми квадратичних помилок між наступними ітераціями.