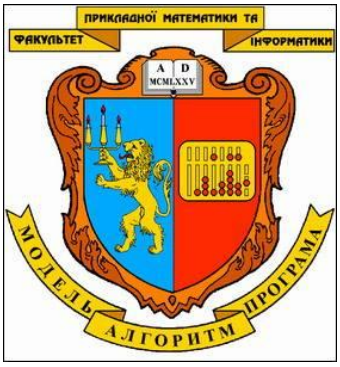
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Факультет прикладної математики та інформатики
Кафедра програмування



Індивідуальне завдання № 6

Нелінійні моделі

Виконала:
студентка групи ПМОм-11
Кравець Ольга

Львів 2025

Хід роботи

Варіант - 3

Визначила variant. Встановила set.seed та згенерувала redundant.

```
> variant=3
> variant
[1] 3
> set.seed(variant)
> redundant=floor(runif(1,5,25))
> redundant
[1] 8
```

Завдання 1.

Модифікувала дані Auto

```
> set.seed(variant)
> Auto_new=Auto[-sample(1:length(Auto[,1]), round((redundant / 100) * length(Auto[,1]))), ]
> fix(Auto_new)
```

	row.names	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	var11	var12	var13
1	1	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu			
2	2	15	8	350	165	3693	11.5	70	1	buick skylark 320			
3	3	18	8	318	150	3436	11	70	1	plymouth satellite			
4	4	16	8	304	150	3433	12	70	1	amc rebel sst			
5	5	17	8	302	140	3449	10.5	70	1	ford torino			
6	6	15	8	429	198	4341	10	70	1	ford galaxie 500			
7	7	14	8	454	220	4354	9	70	1	chevrolet impala			
8	8	14	8	440	215	4312	8.5	70	1	plymouth fury iii			
9	9	14	8	455	225	4425	10	70	1	pontiac catalina			
10	10	15	8	390	190	3850	8.5	70	1	amc ambassador dpl			
11	11	15	8	383	170	3563	10	70	1	dodge challenger se			
12	13	15	8	400	150	3761	9.5	70	1	chevrolet monte carlo			
13	14	14	8	455	225	3086	10	70	1	buick estate wagon (sw)			
14	16	22	6	198	95	2833	15.5	70	1	plymouth duster			
15	17	18	6	199	97	2774	15.5	70	1	amc hornet			
16	18	21	6	200	85	2587	16	70	1	ford maverick			
17	19	27	4	97	88	2130	14.5	70	3	datsun pl510			
18	20	26	4	97	46	1835	20.5	70	2	volkswagen 1131 deluxe sedan			

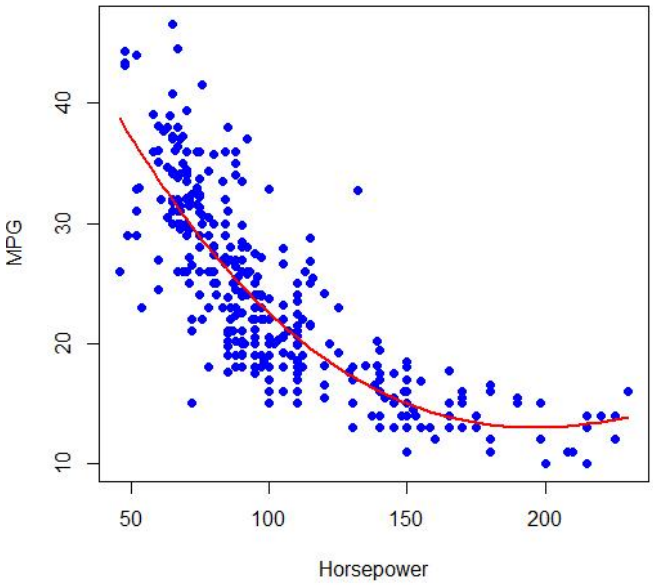
Використовуючи poly(), встановила кубічну поліноміальну регресію для передбачення mpg за допомогою horsepower.

```
> PolReg = lm(mpg~poly(horsepower,3,raw=T), data=Auto_new)
> summary(PolReg)

Call:
lm(formula = mpg ~ poly(horsepower, 3, raw = T), data = Auto_new)

Residuals:
     Min       1Q   Median       3Q      Max
-14.7795  -2.4654  -0.0825   2.1506  15.7202

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                    6.064e+01  4.739e+00  12.797  < 2e-16 ***
poly(horsepower, 3, raw = T)1 -5.660e-01  1.229e-01  -4.604 5.78e-06 ***
poly(horsepower, 3, raw = T)2  2.059e-03  9.909e-04   2.078   0.0384 *
poly(horsepower, 3, raw = T)3 -2.099e-06  2.486e-06  -0.844   0.3991
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.443 on 357 degrees of freedom
Multiple R-squared:  0.6821,    Adjusted R-squared:  0.6794
F-statistic: 255.3 on 3 and 357 DF,  p-value: < 2.2e-16
```

Результати регресії: третій степінь не є значущим для моделі, оскільки його значення Pr(>|t|) = 0.3991 перевищує поріг 0.05.

Побудувала графік даних та поліноміальної регресії.

```
> horsepower_sorted = sort(Auto_new$horsepower)
> predicted_mpg_sorted = predict(PolReg, newdata = data.frame(horsepower = horsepower_sorted))
> plot(Auto_new$horsepower, Auto_new$mpg, xlab = "Horsepower", ylab = "MPG", pch = 16, col = "blue")
> lines(horsepower_sorted, predicted_mpg_sorted, col = "red", lwd = 2)
```

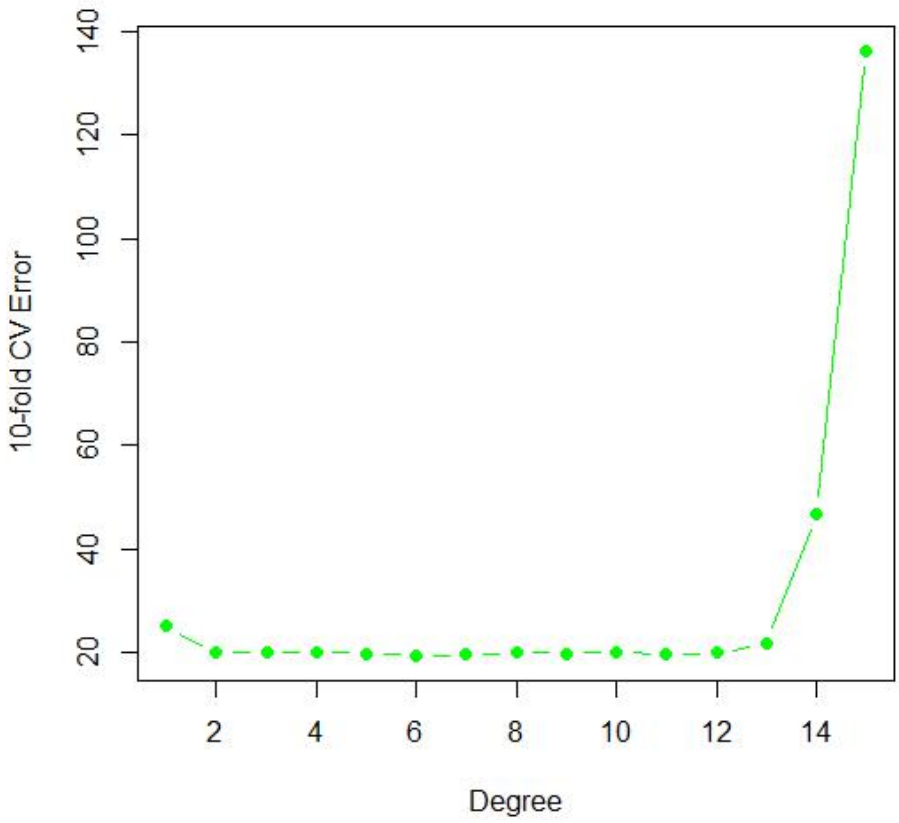


Побудувала поліноміальні моделі для степенів 1-15 і їхні RSS.

```
> rss_values = numeric(15)
> for (d in 1:15) {
+   model = lm(mpg ~ poly(horsepower, d, raw = TRUE), data = Auto_new)
+   predictions = predict(model, newdata = Auto_new)
+   residuals = Auto_new$mpg - predictions
+   rss_values[d] = sum(residuals^2)
+ }
> for (d in 1:15) {
+   cat(sprintf("%d\t%.2f\n", d, rss_values[d]))
+ }
1       8926.08
2       7062.84
3       7048.77
4       7010.73
5       6841.03
6       6768.89
7       6701.40
8       6692.02
9       6677.53
10      6664.46
11      6651.97
12      6651.97
13      6650.06
14      6650.06
15      6649.35
```

На основі перехресної перевірки вибрала «оптимальний» степінь для поліноміальної регресії.

```
> set.seed(variant)
> cv_error = sapply(1:15, function(d) {
+   cv.glm(Auto_new, glm(mpg ~ poly(horsepower, d, raw = TRUE), data = Auto_new), K = 10)$delta[1]
+ })
> plot(1:15, cv_error, type = "b", pch = 19, col = "darkgreen",
+      xlab = "Degree", ylab = "10-fold CV Error")
>
> set.seed(variant)
> cv_error = sapply(1:15, function(d) {
+   cv.glm(Auto_new, glm(mpg ~ poly(horsepower, d, raw = TRUE), data = Auto_new), K = 10)$delta[1]
+ })
> plot(1:15, cv_error, type = "b", pch = 19, col = "green",
+      xlab = "Degree", ylab = "10-fold CV Error")
```



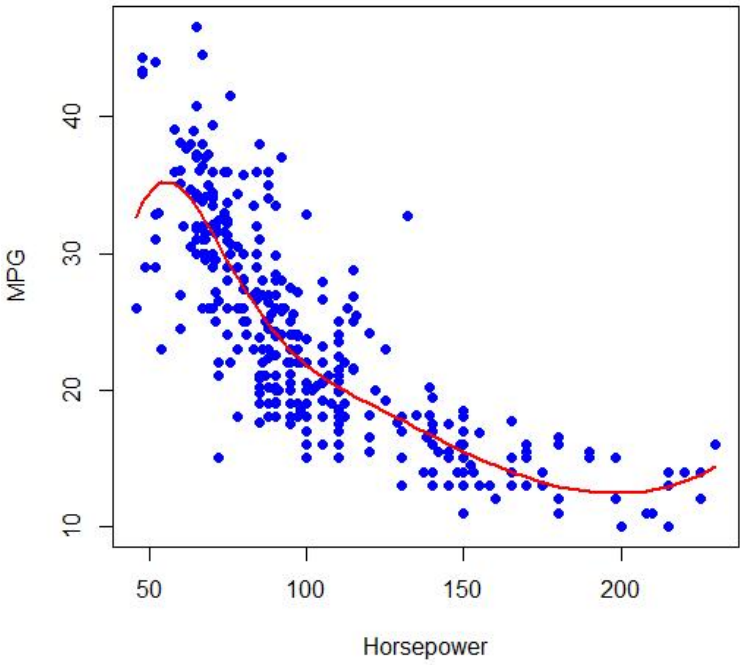
З графіка видно, що найнижча помилка досягається при степенях від 3 до ~10. Там помилка є стабільно низькою та майже не змінюється. Після степеня 12-13 помилка стрімко зростає, особливо при 15. Оптимальний степінь полінома - 3 або трохи більший (до 6–8), оскільки в цьому діапазоні помилка мінімальна і модель ще не перенавчається.

Використовуючи bs(), пристосувала сплайн регресію для прогнозування mpg за допомогою horsepower.

```
> knots = quantile(Auto_new$horsepower, probs = c(0.25, 0.5, 0.75))
> SplineReg = lm(mpg ~ bs(horsepower, knots = knots), data = Auto_new)
> summary(SplineReg)

Call:
lm(formula = mpg ~ bs(horsepower, knots = knots), data = Auto_new)

Residuals:
     Min       1Q   Median       3Q      Max
-15.7146  -2.5515  -0.2403   2.2368  15.1387

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                       32.650      1.979  16.502  < 2e-16 ***
bs(horsepower, knots = knots)1     5.635      3.511   1.605    0.109
bs(horsepower, knots = knots)2    -2.059      2.027  -1.016    0.310
bs(horsepower, knots = knots)3   -10.860      2.282  -4.758 2.85e-06 ***
bs(horsepower, knots = knots)4   -16.685      2.777  -6.007 4.69e-09 ***
bs(horsepower, knots = knots)5   -23.128      3.781  -6.117 2.52e-09 ***
bs(horsepower, knots = knots)6   -18.265      2.860  -6.386 5.37e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.354 on 354 degrees of freedom
Multiple R-squared:  0.6973,    Adjusted R-squared:  0.6922
F-statistic: 135.9 on 6 and 354 DF,  p-value: < 2.2e-16
```

Побудувала графік даних та отриманої сплайн регресії.

```
> horsepower_grid = sort(Auto_new$horsepower)
> spline_predictions = predict(SplineReg, newdata = data.frame(horsepower = horsepower_grid))
> plot(Auto_new$horsepower, Auto_new$mpg, pch = 16, col = "blue", xlab = "Horsepower", ylab = "MPG")
> lines(horsepower_grid, spline_predictions, col = "red", lwd = 2)
```



Пристосувала сплайн регресію для діапазону ступенів свободи та навела відповідні RSS.

```
> rss_spline = numeric(13)
> for (df in 3:15) {
+   model = lm(mpg ~ bs(horsepower, df = df), data = Auto_new)
+   preds = predict(model, newdata = Auto_new)
+   residuals = Auto_new$mpg - preds
+   rss_spline[df - 2] = sum(residuals^2)
+ }
> for (i in 1:13) {
+   cat(sprintf("%d\t%.2f\n", i + 2, rss_spline[i]))
+ }
3       7048.77
4       6874.26
5       6724.81
6       6711.22
7       6730.13
8       6685.25
9       6566.16
10      6585.23
11      6516.58
12      6412.70
13      6449.28
14      6427.76
15      6417.09
```

На основі перехресної перевірки вибрала найкращий ступінь свободи для сплайн регресії на цих даних.

```
> set.seed(variant)
> cv_error_spline = numeric(13)
> for (df in 3:15) {
+   glm_fit = glm(mpg ~ bs(horsepower, df = df), data = Auto_new)
+   cv_result = cv.glm(Auto_new, glm_fit, K = 10)
+   cv_error_spline[df - 2] = cv_result$delta[1]
+ }
> for (i in 1:13) {
+   cat(sprintf("%d\t%.2f\n", i + 2, cv_error_spline[i]))
+ }
3       19.88
4       19.54
5       19.46
6       19.42
7       19.76
8       19.55
9       19.69
10      19.98
11      19.87
12      19.69
13      19.14
14      19.83
15      20.09
```

Найкращий ступінь свободи = 13, оскільки помилка = 19.14 - найменший серед усіх.

Порівняла найкращу поліноміальну регресію та найкращу сплайн регресію.

```
> set.seed(variant)
> auto_poly = poly(Auto_new$horsepower, 6, raw = TRUE)
> auto_poly_model = glm(Auto_new$mpg ~ auto_poly)
> auto_poly_summary = summary(auto_poly_model)
> approx_cv_error = auto_poly_summary$dispersion * (nrow(Auto_new) - auto_poly_summary$df[1]) / nrow(Auto_new)
> approx_cv_error
[1] 18.75039
> set.seed(variant)
> auto_spline = bs(Auto_new$horsepower, df = 13)
> auto_spline_model = glm(Auto_new$mpg ~ auto_spline)
> auto_spline_summary = summary(auto_spline_model)
> approx_cv_error_spline = auto_spline_summary$dispersion * (nrow(Auto_new) - auto_spline_summary$df[1]) / nrow(Auto_new)
> approx_cv_error_spline
[1] 17.86504
```

Сплайн-регресія (df=13) показала меншу середню помилку (17.87) порівняно з поліноміальною регресією ступеня 6 (18.75), тому вона є точнішою моделлю.

Завдання 2.

Застосувала метод підгонки для моделі множинної лінійної регресії.

Згенерувала предиктори, визначила ε, обчислила залежну змінну Y за формулою зі завдання.

```
> variant = 3
> variant
[1] 3
>
> DegFd = variant / 15
> n = 100
>
> set.seed(variant)
> X1 = rt(n, df = DegFd)
> X2 = rt(n, df = DegFd)
> e = rnorm(n, mean = 0, sd = 1)
> Y = variant * X1 + 2 * (variant + 1) * X2 + e
```

Використовуючи rchisq(), ініціалізувала β1.

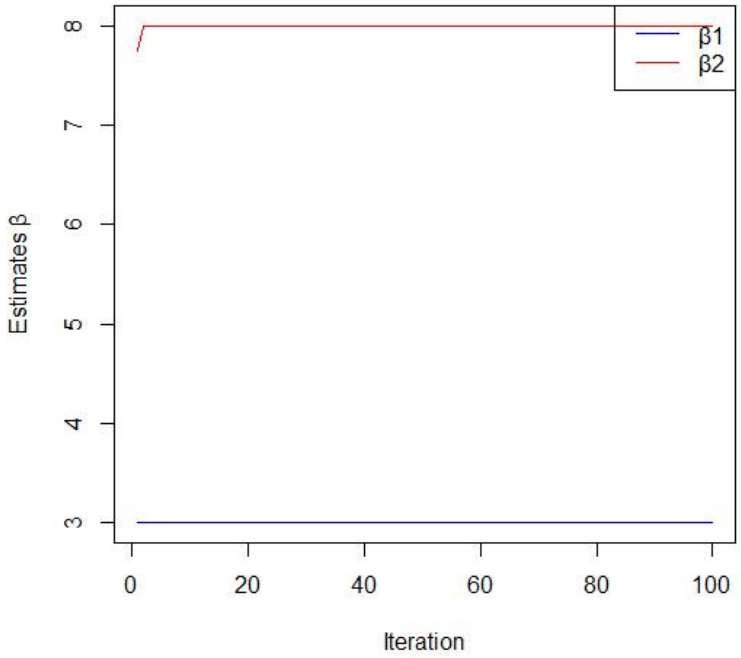
```
> beta1 = rchisq(1, df = variant / 20)
```

Запустила цикл з 100 ітерацій, в якому оцінки β1 і β2 оновлюються за допомогою lm(). Зберегла оцінки на кожній ітерації в матрицю results.

```
> set.seed(variant)
> beta1 = rchisq(1, df = variant / 20)
>
> result = matrix(NA, nrow = 100, ncol = 2)
> for (i in 1:100) {
+   beta2 = lm(I(Y - beta1 * X1) ~ X2)$coef[2]
+   beta1 = lm(I(Y - beta2 * X2) ~ X1)$coef[2]
+   result[i, ] <- c(beta1, beta2)
+ }
```

Побудувала графіки.

```
> plot(result[,1], type = "l", col = "blue", ylim = range(result), ylab = "Estimates β", xlab = "Iteration")
> lines(result[,2], col = "red")
> legend("topright", legend = c("β1", "β2"), col = c("blue", "red"), lty = 1)
```



Порівняла параметри з моделлю Y ~ X1 + X2.

```
> summary(lm(Y ~ X1 + X2))$coefficients
               Estimate   Std. Error       t value  Pr(>|t|)
(Intercept) -0.1506697 9.286340e-02 -1.622487e+00 0.1079449
X1           3.0000000 1.734057e-10  1.730047e+10 0.0000000
X2           8.0000000 2.801701e-09  2.855408e+09 0.0000000
```

Метод множинної регресії дає точніші та стабільніші оцінки, тоді як метод підгонки має більшу варіативність і потребує більше ітерацій для стабільності.