Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

з дисципліни «Методи оптимізації та планування експерименту»

# на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів»

Виконав:

студент 2-го курсу ФІОТ

групи ІВ-82

Сірокомський Микола

Варiант: 218

Перевірив:  
Регіда П.Г.

Київ – 2020

**Код**ї

**import** random  
**import** numpy **as** np  
**import** sklearn.linear\_model **as** lm  
**from** scipy.stats **import** f, t  
**from** functools **import** partial  
**from** pyDOE2 **import** ccdesign  
  
  
x\_range = ((-5, 5), (-1, 6), (-10, 1))  
  
x\_aver\_max = sum([x[1] **for** x **in** x\_range]) / 3 *# середнє Xmax*x\_aver\_min = sum([x[0] **for** x **in** x\_range]) / 3 *# середнє Xmin*y\_max = 200 + int(x\_aver\_max)  
y\_min = 200 + int(x\_aver\_min)  
  
  
**def** s\_kv(y, y\_aver, n, m): *# квадратична дисперсія* res = []  
 **for** i **in** range(n):  
 s = sum([(y\_aver[i] - y[i][j]) \*\* 2 **for** j **in** range(m)]) / m *# (Yсер - Y)^2* res.append(round(s, 3))  
 **return** res  
  
  
**def** regression(x, b):  
 y = sum([x[i] \* b[i] **for** i **in** range(len(x))])  
 **return** y  
  
  
**def** plan\_matrix5(n, m):  
 print(**f'\nГереруємо матрицю планування для n = {n}, m = {m}'**)  
  
 y = np.zeros(shape=(n, m)) *# створюємо матрицю з нулів* **for** i **in** range(n):  
 **for** j **in** range(m):  
 y[i][j] = random.randint(y\_min, y\_max) *# заповнюємо цю матрицю ігриками* **if** n > 14:  
 no = n - 14  
 **else**:  
 no = 1  
 x\_norm = ccdesign(3, center=(0, no)) *# Central-Composite designs* x\_norm = np.insert(x\_norm, 0, 1, axis=1)  
  
 **for** i **in** range(4, 11):  
 x\_norm = np.insert(x\_norm, i, 0, axis=1)  
  
 l = 1.215  
  
 *# матриця планування з нормовaними значеннями* **for** i **in** range(len(x\_norm)):  
 **for** j **in** range(len(x\_norm[i])):  
 **if** x\_norm[i][j] < -1 **or** x\_norm[i][j] > 1:  
 **if** x\_norm[i][j] < 0:  
 x\_norm[i][j] = -l  
 **else**:  
 x\_norm[i][j] = l  
  
 **def** add\_sq\_nums(x): *# рахуємо квадратні числа* **for** i **in** range(len(x)):  
 x[i][4] = x[i][1] \* x[i][2]  
 x[i][5] = x[i][1] \* x[i][3]  
 x[i][6] = x[i][2] \* x[i][3]  
 x[i][7] = x[i][1] \* x[i][3] \* x[i][2]  
 x[i][8] = x[i][1] \*\* 2  
 x[i][9] = x[i][2] \*\* 2  
 x[i][10] = x[i][3] \*\* 2  
 **return** x  
  
 x\_norm = add\_sq\_nums(x\_norm) *# додаємо їх в матрицю* x = np.ones(shape=(len(x\_norm), len(x\_norm[0])), dtype=np.int64) *# заповнюємо матрицю одиницями  
 # матриця планування з натуральними значеннями факторів* **for** i **in** range(8):  
 **for** j **in** range(1, 4):  
 **if** x\_norm[i][j] == -1:  
 x[i][j] = x\_range[j - 1][0]  
 **else**:  
 x[i][j] = x\_range[j - 1][1]  
  
 **for** i **in** range(8, len(x)):  
 **for** j **in** range(1, 3):  
 x[i][j] = (x\_range[j - 1][0] + x\_range[j - 1][1]) / 2  
  
 dx = [x\_range[i][1] - (x\_range[i][0] + x\_range[i][1]) / 2 **for** i **in** range(3)]  
  
 x[8][1] = l \* dx[0] + x[9][1]  
 x[9][1] = -l \* dx[0] + x[9][1]  
 x[10][2] = l \* dx[1] + x[9][2]  
 x[11][2] = -l \* dx[1] + x[9][2]  
 x[12][3] = l \* dx[2] + x[9][3]  
 x[13][3] = -l \* dx[2] + x[9][3]  
  
 x = add\_sq\_nums(x) *# додаємо квадратні числа в матрицю за натуральними значеннями* print(**'\nX:\n'**, x)  
 print(**'\nX нормоване:\n'**)  
 **for** i **in** x\_norm:  
 print([round(x, 2) **for** x **in** i])  
 print(**'\nY:\n'**, y)  
  
 **return** x, y, x\_norm  
  
  
**def** find\_coef(X, Y, norm=**False**):  
 skm = lm.LinearRegression(fit\_intercept=**False**) *# знаходимо коефіцієнти рівняння регресії* skm.fit(X, Y)  
 B = skm.coef\_  
  
 **if** norm == 1:  
 print(**'\nКоефіцієнти рівняння регресії з нормованими X:'**)  
 **else**:  
 print(**'\nКоефіцієнти рівняння регресії:'**)  
 B = [round(i, 3) **for** i **in** B]  
 print(B)  
 print(**'\nРезультат рівняння зі знайденими коефіцієнтами:\n'**, np.dot(X, B))  
 **return** B  
  
  
**def** kriteriy\_cochrana(y, y\_aver, n, m):  
 f1 = m - 1 *# степені свободи* f2 = n  
 q = 0.05 *# рівень значимості* S\_kv = s\_kv(y, y\_aver, n, m)  
 Gp = max(S\_kv) / sum(S\_kv)  
 print(**'\nПеревірка за критерієм Кохрена'**)  
 **return** Gp  
  
  
**def** cohren(f1, f2, q=0.05):  
 q1 = q / f1  
 fisher\_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) \* f2)  
 **return** fisher\_value / (fisher\_value + f1 - 1)  
  
  
*# оцінки коефіцієнтів***def** bs(x, y\_aver, n):  
 res = [sum(1 \* y **for** y **in** y\_aver) / n]  
  
 **for** i **in** range(len(x[0])):  
 b = sum(j[0] \* j[1] **for** j **in** zip(x[:, i], y\_aver)) / n  
 res.append(b)  
 **return** res  
  
  
**def** kriteriy\_studenta(x, y, y\_aver, n, m):  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 s\_kv\_aver = sum(S\_kv) / n  
  
 *# статиcтична оцінка дисперсії* s\_Bs = (s\_kv\_aver / n / m) \*\* 0.5 *# статистична оцінка дисперсії* Bs = bs(x, y\_aver, n)  
 ts = [round(abs(B) / s\_Bs, 3) **for** B **in** Bs]  
  
 **return** ts  
  
  
**def** kriteriy\_fishera(y, y\_aver, y\_new, n, m, d):  
 S\_ad = m / (n - d) \* sum([(y\_new[i] - y\_aver[i]) \*\* 2 **for** i **in** range(len(y))])  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 S\_kv\_aver = sum(S\_kv) / n  
  
 **return** S\_ad / S\_kv\_aver  
  
  
**def** check(X, Y, B, n, m):  
 print(**'\n\tПеревірка рівняння:'**)  
 f1 = m - 1  
 f2 = n  
 f3 = f1 \* f2  
 q = 0.05  
  
 *### табличні значення* student = partial(t.ppf, q=1 - q)  
 t\_student = student(df=f3)  
  
 G\_kr = cohren(f1, f2)  
 *###* y\_aver = [round(sum(i) / len(i), 3) **for** i **in** Y]  
 print(**'\nСереднє значення y:'**, y\_aver)  
  
 disp = s\_kv(Y, y\_aver, n, m)  
 print(**'Дисперсія y:'**, disp)  
  
 Gp = kriteriy\_cochrana(Y, y\_aver, n, m)  
 print(**f'Gp = {Gp}'**)  
 **""" Перевірюємо критерій кохрена, якщо не адекватно, m+1 і починаємо спочатку"""  
 if** Gp < G\_kr:  
 print(**f'З ймовірністю {1-q} дисперсії однорідні.'**)  
 **else**:  
 print(**"Необхідно збільшити кількість дослідів"**)  
 m += 1  
 main(n, m)  
  
 ts = kriteriy\_studenta(X[:, 1:], Y, y\_aver, n, m)  
 print(**'\nКритерій Стьюдента:\n'**, ts)  
 res = [t **for** t **in** ts **if** t > t\_student]  
 final\_k = [B[i] **for** i **in** range(len(ts)) **if** ts[i] **in** res]  
 print(**'\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з рівняння.'**.format(  
 [round(i, 3) **for** i **in** B **if** i **not in** final\_k]))  
  
 y\_new = []  
 **for** j **in** range(n):  
 y\_new.append(regression([X[j][i] **for** i **in** range(len(ts)) **if** ts[i] **in** res], final\_k))  
  
 print(**f'\nЗначення "y" з коефіцієнтами {final\_k}'**)  
 print(y\_new)  
  
 d = len(res)  
 **if** d >= n:  
 print(**'\nF4 <= 0'**)  
 print(**''**)  
 **return** f4 = n - d  
  
 F\_p = kriteriy\_fishera(Y, y\_aver, y\_new, n, m, d)  
  
 fisher = partial(f.ppf, q=0.95)  
 f\_t = fisher(dfn=f4, dfd=f3) *# табличне значення* print(**'\nПеревірка адекватності за критерієм Фішера'**)  
 print(**'Fp ='**, F\_p)  
 print(**'F\_t ='**, f\_t)  
 **if** F\_p < f\_t:  
 print(**'Математична модель адекватна експериментальним даним'**)  
 **else**:  
 print(**'Математична модель не адекватна експериментальним даним'**)  
  
  
**def** main(n, m):  
 X5, Y5, X5\_norm = plan\_matrix5(n, m)  
  
 y5\_aver = [round(sum(i) / len(i), 3) **for** i **in** Y5]  
 B5 = find\_coef(X5, y5\_aver)  
  
 check(X5\_norm, Y5, B5, n, m)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main(15, 3)

**Результат**

Гереруємо матрицю планування для n = 15, m = 3

X:

[[ 1 -5 -1 -10 5 50 10 -50 25 1 100]

[ 1 5 -1 -10 -5 -50 10 50 25 1 100]

[ 1 -5 6 -10 -30 50 -60 300 25 36 100]

[ 1 5 6 -10 30 -50 -60 -300 25 36 100]

[ 1 -5 -1 1 5 -5 -1 5 25 1 1]

[ 1 5 -1 1 -5 5 -1 -5 25 1 1]

[ 1 -5 6 1 -30 -5 6 -30 25 36 1]

[ 1 5 6 1 30 5 6 30 25 36 1]

[ 1 6 2 1 12 6 2 12 36 4 1]

[ 1 -6 2 1 -12 -6 2 -12 36 4 1]

[ 1 0 6 1 0 0 6 0 0 36 1]

[ 1 0 -2 1 0 0 -2 0 0 4 1]

[ 1 0 2 7 0 0 14 0 0 4 49]

[ 1 0 2 -5 0 0 -10 0 0 4 25]

[ 1 0 2 1 0 0 2 0 0 4 1]]

X нормоване:

[1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0]

[1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

[1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0]

[1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0]

[1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0]

[1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0]

[1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0]

[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

[1.0, -1.22, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 1.48, 0.0, 0.0]

[1.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0, 0.0]

[1.0, 0.0, -1.22, 0.0, -0.0, 0.0, -0.0, -0.0, 0.0, 1.48, 0.0]

[1.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0]

[1.0, 0.0, 0.0, -1.22, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 1.48]

[1.0, 0.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48]

[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Y:

[[202. 202. 204.]

[202. 198. 197.]

[197. 197. 203.]

[198. 202. 202.]

[202. 202. 204.]

[199. 204. 203.]

[196. 203. 201.]

[203. 201. 200.]

[204. 196. 200.]

[204. 195. 200.]

[200. 195. 201.]

[204. 201. 199.]

[197. 204. 201.]

[199. 197. 201.]

[199. 198. 201.]]

Коефіцієнти рівняння регресії:

[200.013, -0.057, -0.68, 0.163, 0.033, 0.023, -0.008, -0.004, 0.035, 0.092, 0.009]

Результат рівняння зі знайденими коефіцієнтами:

[202.65 199.05 199.115 200.625 202.155 201.525 199.544 200.944 200.581

200.293 199.369 201.929 200.491 198.511 199.177]

Перевірка рівняння:

Середнє значення y: [202.667, 199.0, 199.0, 200.667, 202.667, 202.0, 200.0, 201.333, 200.0, 199.667, 198.667, 201.333, 200.667, 199.0, 199.333]

Дисперсія y: [0.889, 4.667, 8.0, 3.556, 0.889, 4.667, 8.667, 1.556, 10.667, 13.556, 6.889, 4.222, 8.222, 2.667, 1.556]

Перевірка за критерієм Кохрена

Gp = 0.1680426428659973

З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:

[579.687, 0.335, 0.404, 0.509, 1.414, 0.514, 0.257, 0.643, 423.741, 423.836, 423.741]

Коефіцієнти [-0.057, -0.68, 0.163, 0.033, 0.023, -0.008, -0.004] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [200.013, 0.035, 0.092, 0.009]

[200.149, 200.149, 200.149, 200.149, 200.149, 200.149, 200.149, 200.149, 200.064667875, 200.064667875, 200.1488127, 200.1488127, 200.02628602500002, 200.02628602500002, 200.013]

Перевірка адекватності за критерієм Фішера

Fp = 1.3249027780738314

F\_t = 2.125558760875511

Математична модель адекватна експериментальним даним