

Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

Звіт
з лабораторної роботи №4
з навчальної дисципліни «Методи оптимізації та планування експерименту»
Тема: Проведення трьохфакторного експерименту при використанні рівняння
регресії з урахуванням ефекту взаємодії.

Виконав:
Студент 2 курсу кафедри ОТ ФІОТ
Групи ІО-93
Куцурайс Георгій

Перевірив:
Регіда П. Г.

Київ 2021

Мета: провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання:

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.
3. Знайти коефіцієнти рівняння регресії і записати його .
4. Провести 3 статистичні перевірки – за критеріями Кохрена , Стьюдента , Фішера .
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії .
6. Написати комп'ютерну програму , яка усе це моделює .

Варіант:

316	10	50	25	65	50	65
-----	----	----	----	----	----	----

$$x_{1min}=10 ; x_{2min}=25 ; x_{3min}=50;$$

$$x_{1max}=50 ; x_{2max}=65 ; x_{3max}=65;$$

$$y_{min}=228 ; y_{max}=260;$$

Лістинг програми

```
"""
* Copyright © 2021 drewg3r
* https://github.com/drewg3r/DOX-labs

main.py: main file to run the program.
"""

from tabulate import tabulate
import math
import random

class Lab4:
    N = 8
    m = 3

    X0 = [1, 1, 1, 1, 1, 1, 1, 1]
    X1 = [-1, -1, -1, -1, 1, 1, 1, 1]
    X2 = [-1, -1, 1, 1, -1, -1, 1, 1]
    X3 = [-1, 1, -1, 1, -1, 1, -1, 1]
    X12 = [1, 1, -1, -1, -1, -1, 1, 1]
    X13 = [1, -1, 1, -1, -1, 1, -1, 1]
    X23 = [1, -1, -1, 1, 1, -1, -1, 1]
    X123 = [-1, 1, 1, -1, 1, -1, -1, 1]
    X = [X1, X2, X3, X12, X13, X23, X123]
    Xf = [X0, X1, X2, X3]

    def __init__(self):
        print
        ("GeorgeLab_4" )
        x1min = 10
        x2min = 25
        x3min = 50

        x1max = 50
        x2max = 65
        x3max = 65

        ymin = round((x1min + x2min + x3min) / 3) + 200
        ymax = round((x1max + x2max + x3max) / 3) + 200

        self.x_range = ((x1min, x1max), (x2min, x2max), (x3min, x3max))

        x = []
        x.append([random.randint(x1min, x1max) for i in range(8)])
        x.append([random.randint(x2min, x2max) for i in range(8)])
        x.append([random.randint(x3min, x3max) for i in range(8)])
        x.append([x[0][i] * x[1][i] for i in range(8)])
        x.append([x[0][i] * x[2][i] for i in range(8)])
        x.append([x[1][i] * x[2][i] for i in range(8)])
        x.append([x[0][i] * x[1][i] * x[2][i] for i in range(8)])

        y = []
        y.append([random.randint(ymin, ymax) for i in range(8)])
        y.append([random.randint(ymin, ymax) for i in range(8)])
        y.append([random.randint(ymin, ymax) for i in range(8)])

        self.yavg = [sum(x) / len(x) for x in zip(*y)]

        print("PFE Matrix:")
```

```

print(
    tabulate(
        zip(*x + y),
        headers=[
            "X1",
            "X2",
            "X3",
            "X12",
            "X13",
            "X23",
            "X123",
            "Y1",
            "Y2",
            "Y3",
        ],
        floatfmt=".4f",
        tablefmt="fancy_grid",
    )
)

self.x = x
self.y = y

def normalize_x(self):
    x0 = [(self.x_range[i][0] + self.x_range[i][1]) / 2 for i in range(3)]
    dx = [x0[i] - self.x_range[i][0] for i in range(3)]
    xn = []
    for i in range(3):
        xn.append([(self.x[i][j] - x0[i]) / dx[i] for j in range(8)])
    self.xn = xn
    print("\nNormalized factors")
    print(
        tabulate(
            zip(*xn),
            headers=["X1", "X2", "X3"],
            floatfmt=".4f",
            tablefmt="fancy_grid",
        )
    )

def calculate_b(self):
    y_sum = sum(self.yavg)
    b = [0] * 8
    b[0] = y_sum
    for i in range(8):
        b[1] += self.yavg[i] * self.xn[0][i]
        b[2] += self.yavg[i] * self.xn[1][i]
        b[3] += self.yavg[i] * self.xn[2][i]
        b[4] += self.yavg[i] * self.xn[0][i] * self.xn[1][i]
        b[5] += self.yavg[i] * self.xn[0][i] * self.xn[2][i]
        b[6] += self.yavg[i] * self.xn[1][i] * self.xn[2][i]
        b[7] += self.yavg[i] * self.xn[0][i] * self.xn[1][i] * self.xn[2][i]

    for i in range(8):
        b[i] = b[i] / self.N

    print("\nRegression equation:")
    print(
        "y = {:.3f} + {:.3f}*x1 + {:.3f}*x2 + {:.3f}*x3 + {:.3f}*x1*x2 + {:.3f}*x1*x3 + {:.3f}*x2*x3 + {:.3f}*x1*x2*x3\n".format(
            *b
        )
    )

```

```

    )
    self.b = b

def cochrans_test(self):
    # Calculate dispersions for every row
    d = []
    print("Dispersions:")
    for i in range(8):
        d.append(sum([(x[0] - self.yavg[i])**2 for x in self.y]) / 8)
        print("d{} = {:.3f}".format(i + 1, d[-1]))
    gp = max(d) / sum(d)
    self.d = d
    print("\ngp = {:.3f}".format(gp))
    if gp < 0.7679:
        print("Cochran's C test passed\n")
    else:
        print("Cochran's C test failed\n")

def student_crit_check(self):
    N = len(self.d)
    sbsq = sum(self.d) / N
    self.sbsq = sbsq
    sbssq = sbsq / (self.m * N)
    sbs = math.sqrt(sbssq)

    print("Sb = {:.3f}".format(sbs))

    b = []
    for i in range(4):
        b.append(sum([self.yavg[j] * self.Xf[i][j] for j in range(N)]) / N)

    t = [abs(x) / sbs for x in b]

    print("Beta:")
    for i in range(4):
        print("b{} = {:.3f}".format(i, b[i]))

    print("\nt:")
    for i in range(4):
        print("t{} = {:.3f}".format(i, t[i]))

    print()

    t_tabl = 2.306
    regr_eq = ""
    nm = []
    xs = ["x1", "x2", "x3"]
    d = 0

    yd = [0, 0, 0, 0]
    for i in range(4):
        if t[i] < t_tabl:
            nm.append(i)
            d += 1
        else:
            for j in range(4):
                if i == 0:
                    yd[j] += self.b[i]
                else:
                    yd[j] += self.b[i] * self.X[2][j]

```

```

        if i == 0:
            regr_eq += "{:.3f} + ".format(self.b[0])
        else:
            regr_eq += "{:.3f}*{} + ".format(self.b[i], xs[i - 1])
    if len(regr_eq) != 0:
        regr_eq = regr_eq[0:-2]

    nmt = ", ".join(["t" + str(x) for x in nm])
    nmb = ", ".join(["b" + str(x + 1) for x in nm])
    print("{} < t_tabl(t_tabl=2.306)".format(nmt))
    print("Factors {} can be excluded".format(nmb))
    print("Regression equation without excluded factors:")
    print("y = {}\n".format(regr_eq))

    for i in range(4):
        print("y{} = {:.3f}".format(i + 1, yd[i]))

    self.d = d
    self.yd = yd

def fisher_crit_check(self):
    print("d = {}".format(self.d))
    sadsq = (self.m / (self.N - self.d)) * sum(
        [(self.yd[i] - self.yavg[i]) ** 2 for i in range(4)]
    )
    Fp = sadsq / self.sbsq
    print("S2ad = {:.3f}\nFp = {:.3f}".format(sadsq, Fp))
    table = [5.3, 4.5, 4.1, 3.8, 3.7, 3.6, 3.3, 3.1, 2.9]
    tablec = table[self.N - self.d - 1]
    if Fp < tablec:
        print(" F-test passed/model is adequate")
    else:
        print(" F-test failed/model is NOT adequate")

def calc_y(self, x1, x2, x3):
    return (
        self.b[0]
        + self.b[1] * x1
        + self.b[2] * x2
        + self.b[3] * x3
        + self.b[4] * x1 * x2
        + self.b[5] * x1 * x3
        + self.b[6] * x2 * x3
        + self.b[7] * x1 * x2 * x3
    )

lab4 = Lab4()
lab4.normalize_x()
lab4.calculate_b()
lab4.cochran_test()
lab4.student_crit_check()
lab4.fisher_crit_check()

```

Результат виконання роботи

GeorgeLab_4

PFE Matrix:

X1	X2	X3	X12	X13	X23	X123	Y1	Y2	Y3
47	58	59	2726	2773	3422	160834	228	232	235
38	26	50	988	1900	1300	49400	258	244	247
45	25	57	1125	2565	1425	64125	249	254	249
48	31	58	1488	2784	1798	86304	241	238	251
44	57	58	2508	2552	3306	145464	244	258	229
47	27	55	1269	2585	1485	69795	250	232	229
15	49	58	735	870	2842	42630	254	230	239
48	42	56	2016	2688	2352	112896	258	233	234

>>>

Normalized factors

X1	X2	X3
0.8500	0.6500	0.2000
0.4000	-0.9500	-1.0000
0.7500	-1.0000	-0.0667
0.9000	-0.7000	0.0667
0.7000	0.6000	0.0667
0.8500	-0.9000	-0.3333
-0.7500	0.2000	0.0667
0.9000	-0.1500	-0.2000

Regression equation:

$$y = 242.333 + 139.069 \cdot x_1 + -70.344 \cdot x_2 + -37.356 \cdot x_3 + -56.990 \cdot x_1 \cdot x_2 + -21.218 \cdot x_1 \cdot x_3 + 45.496 \cdot x_2 \cdot x_3 + 24.270 \cdot x_1 \cdot x_2 \cdot x_3$$

Dispersions:

$$d_1 = 3.083$$

$$d_2 = 124.583$$

$$d_3 = 138.458$$

$$d_4 = 54.125$$

$$d_5 = 57.083$$

$$d_6 = 13.750$$

$$d_7 = 35.750$$

$$d_8 = 40.583$$

$$g_p = 0.296$$

✓Cochran's C test passed

$$S_b = 1.560$$

Beta:

$$b_0 = 242.333$$

$$b_1 = -1.500$$

$$b_2 = 1.833$$

$$b_3 = 0.583$$

t:

$$t_0 = 155.314$$

$$t_1 = 0.961$$

$$t_2 = 1.175$$

$$t_3 = 0.374$$

$t_1, t_2, t_3 < t_{\text{tabl}}(t_{\text{tabl}}=2.306)$

Factors b_2, b_3, b_4 can be excluded

Regression equation without excluded factors:

$y = 242.333$

$y_1 = 242.333$

$y_2 = 242.333$

$y_3 = 242.333$

$y_4 = 242.333$

$d = 3$

$S_{2ad} = 142.800$

$F_p = 2.444$

✓ F-test passed/model is adequate

```
Cochran' s C test passed

Sb = 1.927
Beta:
b0 = 202.750
b1 = 0.417
b2 = 2.417
b3 = -4.750

t:
t0 = 105.200
t1 = 0.216
t2 = 1.254
t3 = 2.465

t1,t2 < t_tabl(t_tabl=2.306)
Factors b2,b3 can be excluded
Regression equation without excluded factors:
y = 202.750 + -11.537*x3

y1 = 214.288
y2 = 191.213
y3 = 214.288
y4 = 191.213
d = 2
S2ad = 168.842
Fp = 1.894
F-test passed/model is adequate
```

Висновки:

Під час виконання лабораторної роботи було проведено трьохфакторний експеримент, складено матрицю планування, знайдено коефіцієнти рівняння регресії з урахуванням ефекту взаємодії, проведено 3 статистичні перевірки, закріплено отримані знання практичним їх використанням при написанні програми, що реалізує завдання на лабораторну роботу.