Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №4
З дисципліни «Методи наукових досліджень»
За темою:
«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-91
Гутов В.В.
Номер у списку - 8

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

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.

$$y_{i\max} = 200 + x_{cp\max}$$

$$y_{i\min} = 200 + x_{cp\min}$$

де $x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}$, $x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$

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.

№ варіанту	x_1		x_2		x_3	
	min	max	min	max	min	max
108	-5	15	-15	35	15	30

Програмний код

```
import random
from beautifultable import BeautifulTable

# Гутов Віталій
# Варіант 108:
# x1_min = -5, x1_max = 15,
# x2_min = -15, x2_max = 35,
# x3_min = 15, x3_max = 30
# y_min = 200 + xc_min
# y_max = 200 + xc_max


def main():
    global x1_min, x1_max, x2_min, x2_max, x3_min, x3_max
    global m
    global y_matrix
    global average_y
    global n
    global b0, b1, b2, b3, b12, b13, b23, b123
    global plan_matrix, plan_matrix_normal
    m = 3
    n = 8

    x1_min = -5
    x1_max = 15
    x2_min = -15
    x2_max = 35
    x3_min = 15
    x3_max = 30

    xc_min = (x1_min + x2_min + x3_min) / 3
    xc_max = (x1_max + x2_max + x3_max) / 3
```

```
    y_min = 200 + xc_min
    y_max = 200 + xc_max

    y_matrix = [[random.randint(int(y_min), int(y_max)) for _ in range(m)]
for _ in range(n)]

    average_y = [round(sum(i) / len(i), 3) for i in y_matrix]

    norm_x = [[-1, -1, -1],
              [-1, -1, 1],
              [-1, 1, -1],
              [-1, 1, 1],
              [1, -1, -1],
              [1, -1, 1],
              [1, 1, -1],
              [1, 1, 1]]

    b0 = sum(average_y) / n
    b1 = sum([average_y[i] * norm_x[i][0] for i in range(n)]) / n
    b2 = sum([average_y[i] * norm_x[i][1] for i in range(n)]) / n
    b3 = sum([average_y[i] * norm_x[i][2] for i in range(n)]) / n
    b12 = sum([average_y[i] * norm_x[i][0] * norm_x[i][1] for i in range(n)])
/ n
    b13 = sum([average_y[i] * norm_x[i][0] * norm_x[i][2] for i in range(n)])
/ n
    b23 = sum([average_y[i] * norm_x[i][1] * norm_x[i][2] for i in range(n)])
/ n
    b123 = sum([average_y[i] * norm_x[i][0] * norm_x[i][1] * norm_x[i][2] for
i in range(n)]) / n

    plan_matrix = [[x1_min, x2_min, x3_min, x1_min * x2_min, x1_min * x3_min,
x2_min * x3_min, x1_min * x2_min * x3_min],
                   [x1_min, x2_min, x3_max, x1_min * x2_min, x1_min * x3_max,
x2_min * x3_max, x1_min * x2_min * x3_max],
                   [x1_min, x2_max, x3_min, x1_min * x2_max, x1_min * x3_min,
x2_max * x3_min, x1_min * x2_max * x3_min],
                   [x1_min, x2_max, x3_max, x1_min * x2_max, x1_min * x3_max,
x2_max * x3_max, x1_min * x2_max * x3_max],
                   [x1_max, x2_min, x3_min, x1_max * x2_min, x1_max * x3_min,
x2_min * x3_min, x1_max * x2_min * x3_min],
                   [x1_max, x2_min, x3_max, x1_max * x2_min, x1_max * x3_max,
x2_min * x3_max, x1_max * x2_min * x3_max],
                   [x1_max, x2_max, x3_min, x1_max * x2_max, x1_max * x3_min,
x2_max * x3_min, x1_max * x2_max * x3_min],
                   [x1_max, x2_max, x3_max, x1_max * x2_max, x1_max * x3_max,
x2_max * x3_max, x1_max * x2_max * x3_max]]

    plan_matrix_normal = [[-1, -1, -1, 1, 1, 1, -1],
                          [-1, -1, 1, 1, -1, -1, 1],
                          [-1, 1, -1, -1, 1, -1, 1],
                          [-1, 1, 1, -1, -1, 1, -1],
                          [1, -1, -1, -1, -1, 1, 1],
                          [1, -1, 1, -1, 1, -1, -1],
                          [1, 1, -1, 1, -1, -1, -1],
                          [1, 1, 1, 1, 1, 1, 1]]

    result_y = []
    for i in range(n):
        result_y.append(b0 + b1 * plan_matrix[i][0] + b2 * plan_matrix[i][1]
+ b3 * plan_matrix[i][2] +
                        b12 * plan_matrix[i][3] + b13 * plan_matrix[i][4] +
b23 * plan_matrix[i][5] +
                        b123 * plan_matrix[i][6])
```

```
def kohren():
    global m
    global gp
    global s
    s = [sum([(y_matrix[j][i] - average_y[i]) ** 2 for i in range(m)]) / m
for j in range(n)]
    gp = max(s) / sum(s)
    gt = 0.5157

    if gp > gt:
        m += 1
        main()


def studens():
    global n, m
    global average_y
    global b0, b1, b2, b3, b12, b13, b23, b123
    global d
    global blist
    global sb

    d = 8

    sb = sum(s) / n
    s_beta_2 = sb / (n * m)
    s_beta = s_beta_2 ** (1 / 2)

    bb = [b0, b1, b2, b3, b12, b13, b23, b123]
    t = [abs(bb[i]) / s_beta for i in range(n)]
    tt = 2.120
    blist = [b0, b1, b2, b3, b12, b13, b23, b123]

    for i in range(n):
        if t[i] < tt:
            blist[i] = 0
            d -= 1

    return t


def fish():
    global n
    global d
    global blist
    global plan_matrix_for_output
    global average_y
    global sb
    global fp
    y_reg = [b0 + b1 * plan_matrix[i][0] + b2 * plan_matrix[i][1] + b3 *
plan_matrix[i][2] +
             b12 * plan_matrix[i][3] + b13 * plan_matrix[i][4] + b23 *
plan_matrix[i][5] +
             b123 * plan_matrix[i][6] for i in range(n)]
    sad = (m / (n - d)) * int(sum([(y_reg[i] - average_y[i]) ** 2 for i in
range(n)]))
    fp = sad / sb
    if fp > 4.5:
        return 'The regression equation is inadequate to the original at a
significance level of 0.05'
    else:
```

```
        return 'The regression equation is adequate to the original at a
significance level of 0.05'


main()
kohren()
t_list = studens()
fisher = fish()




plan_table = BeautifulTable()
headers_x = ['X{}'.format(i) for i in range(0, m+1)]
headers_x.extend(['X12', 'X13', 'X23', 'X123'])
headers_y = ['Y{}'.format(i) for i in range(1, m+1)]
headers_y.extend(['av_Y', 'S^2'])
plan_table.columns.header = [*headers_x, *headers_y]
x0 = [[1] for _ in range(n)]
for i in range(n):
    plan_table.rows.append([*x0[i], *plan_matrix[i], *y_matrix[i],
average_y[i], s[i]])
print('Матриця планування:')
print(plan_table)

norm_table = BeautifulTable()
headers_x = ['X{}'.format(i) for i in range(0, m+1)]
headers_x.extend(['X12', 'X13', 'X23', 'X123'])
headers_y = ['Y{}'.format(i) for i in range(1, m+1)]
headers_y.extend(['av_Y', 'S^2'])
norm_table.columns.header = [*headers_x, *headers_y]
x0 = [[1] for _ in range(n)]
for i in range(n):
    norm_table.rows.append([*x0[i], *plan_matrix_normal[i], *y_matrix[i],
average_y[i], s[i]])

print('Нормована матриця:')
print(norm_table)

print('Kohren check')
print('Gp = {} < 0.7679'.format(round(gp, 3)))

print('Studens')
for i in range(len(t_list)):
    print('t{} = {}'.format(i, round(t_list[i], 3)))

print('Fisher')
print(fisher)

print('The equation')
print('y = {} + {} * x1 + {} * x2 + {} * x3 + {} * x1x2 + {} * x1x3 + {} *
x2x3 + {} * x1x2x3'
      .format(round(b0, 3), round(b1, 3), round(b2, 3), round(b3, 3),
round(b12, 3), round(b13, 3), round(b23, 3),
              round(b123, 3)))
```

Результати роботи програми

Матриця планування:

X0	X1	X2	X3	X1 2	X1 3	X2 3	X12 3	Y1	Y2	Y3	av_Y	S^2
1	-5	-15	15	75	-7 5	-2 25	112 5	221	224	202	215.667	177.967
1	-5	-15	30	75	-1 50	-4 50	225 0	221	211	222	218.0	26.74
1	-5	35	15	-1 75	-7 5	52 5	-26 25	222	223	226	223.667	23.517
1	-5	35	30	-1 75	-1 50	10 50	-52 50	201	216	207	208.0	165.63
1	15	-15	15	-2 25	22 5	-2 25	-33 75	219	220	223	220.667	5.185
1	15	-15	30	-2 25	45 0	-4 50	-67 50	201	226	200	209.0	279.74
1	15	35	15	52 5	22 5	52 5	787 5	226	203	208	212.333	192.409
1	15	35	30	52 5	45 0	10 50	157 50	202	215	222	213.0	66.189

Нормована матриця:

X0	X1	X2	X3	X12	X13	X23	X12 3	Y1	Y2	Y3	av_Y	S^2
1	-1	-1	-1	1	1	1	-1	221	224	202	215.667	177.967
1	-1	-1	1	1	-1	-1	1	221	211	222	218.0	26.74
1	-1	1	-1	-1	1	-1	1	222	223	226	223.667	23.517
1	-1	1	1	-1	-1	1	-1	201	216	207	208.0	165.63
1	1	-1	-1	-1	-1	1	1	219	220	223	220.667	5.185
1	1	-1	1	-1	1	-1	-1	201	226	200	209.0	279.74
1	1	1	-1	1	-1	-1	-1	226	203	208	212.333	192.409
1	1	1	1	1	1	1	1	202	215	222	213.0	66.189

```
Kohren check
Gp = 0.298 < 0.7679
Studens
t0 = 97.322
t1 = 0.585
t2 = 0.358
t3 = 1.377
t4 = 0.132
t5 = 0.132
t6 = 0.321
t7 = 1.716
Fisher
The regression equation is inadequate to the original at a significance level of 0.05
The equation
y = 215.042 + -1.292 * x1 + -0.792 * x2 + -3.042 * x3 + -0.292 * x1x2 + 0.292 * x1x3 + -0.708 * x2x3 + 3.792 * x1x2x3

Process finished with exit code 0
```

Висновок

Виконуючи дану лабораторну роботу, я провів трьохфакторний експеримент. Склав матрицю планування та знайшов коефіцієнти рівняння регресії, провів статистичні перевірки.

Результати роботи програми наведені вище. Під час виконання роботи проблем не виникло.