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 №3
З дисципліни «Методи наукових досліджень»
За темою:
«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

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 дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Завдання:

1. Скласти матрицю планування для дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку У. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

$$y_{max} = 200 + x_{cp\,max};$$

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

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

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.

3. Провести 3 статистичні перевірки.

4. Написати комп'ютерну програму, яка усе це виконує.

Варіанти обираються по номеру в списку в журналі викладача.

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
108	-30	0	-35	10	0	20

Програмний код

```
import random
from beautifultable import BeautifulTable
import numpy as np
import os
import sys

# Гутов Віталій
# Варіант 108:
# x1_min = -30, x1_max = 0, x2_min = -35, x2_max = 10, x3_min = 0, x3_max =
20
# y_min = 200 + xc_min
# y_max = 200 + xc_max

def main():
    global m
    global y_matrix
    global average_y
    global plan_matrix_x
    global plan_matrix_for_output
    global b0, b1, b2, b3
    global xn
    global n
    x0 = 1
    x1_min = -30
    x1_max = 0
```

```
    x2_min = -35
    x2_max = 10
    x3_min = 0
    x3_max = 20

    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

    n = 4

    xn = [[-1, -1, -1],
          [-1, 1, 1],
          [1, -1, 1],
          [1, 1, -1]]

    plan_matrix_for_output = [[x1_min, x2_min, x3_min],
                              [x1_min, x2_max, x3_max],
                              [x1_max, x2_min, x3_max],
                              [x1_max, x2_max, x3_min]]
    plan_matrix_x = [[x1_min, x1_min, x1_max, x1_max],
                     [x2_min, x2_max, x2_min, x2_max],
                     [x3_min, x3_max, x3_max, x3_min]]

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

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

    mx1 = sum(plan_matrix_x[0]) / len(plan_matrix_x[0])
    mx2 = sum(plan_matrix_x[1]) / len(plan_matrix_x[1])
    mx3 = sum(plan_matrix_x[2]) / len(plan_matrix_x[2])
    my = sum(average_y) / len(average_y)

    a1 = sum([plan_matrix_x[0][i] * average_y[i] for i in
range(len(plan_matrix_x[0]))]) / len(plan_matrix_x[0])
    a2 = sum([plan_matrix_x[1][i] * average_y[i] for i in
range(len(plan_matrix_x[1]))]) / len(plan_matrix_x[1])
    a3 = sum([plan_matrix_x[2][i] * average_y[i] for i in
range(len(plan_matrix_x[2]))]) / len(plan_matrix_x[2])
    a11 = sum([i ** 2 for i in plan_matrix_x[0]]) / len(plan_matrix_x[0])
    a22 = sum([i ** 2 for i in plan_matrix_x[1]]) / len(plan_matrix_x[1])
    a33 = sum([i ** 2 for i in plan_matrix_x[2]]) / len(plan_matrix_x[2])
    a12 = sum([plan_matrix_x[0][i] * plan_matrix_x[1][i] for i in
range(len(plan_matrix_x[0]))]) / len(plan_matrix_x[0])
    a13 = sum([plan_matrix_x[0][i] * plan_matrix_x[2][i] for i in
range(len(plan_matrix_x[0]))]) / len(plan_matrix_x[0])
    a23 = sum([plan_matrix_x[1][i] * plan_matrix_x[2][i] for i in
range(len(plan_matrix_x[0]))]) / len(plan_matrix_x[0])
    a32 = a23
    a31 = a13
    a21 = a12

    main_denominator = np.array([[1, mx1, mx2, mx3],
                                 [mx1, a11, a12, a13],
                                 [mx2, a12, a22, a32],
                                 [mx3, a13, a23, a33]])
    b0_numerator = np.array([[my, mx1, mx2, mx3],
                             [a1, a11, a12, a13],
                             [a2, a12, a22, a32],
                             [a3, a13, a23, a33]])
    b1_numerator = np.array([[1, my, mx2, mx3],
```

```
                              [mx1, a1, a12, a13],
                              [mx2, a2, a22, a32],
                              [mx3, a3, a23, a33]])
    b2_numerator = np.array([[1, mx1, my, mx3],
                              [mx1, a11, a1, a13],
                              [mx2, a12, a2, a32],
                              [mx3, a13, a3, a33]])

    b3_numerator = np.array([[1, mx1, mx2, my],
                              [mx1, a11, a12, a1],
                              [mx2, a12, a22, a2],
                              [mx3, a13, a23, a3]])

    b0 = np.linalg.det(b0_numerator) / np.linalg.det(main_denominator)
    b1 = np.linalg.det(b1_numerator) / np.linalg.det(main_denominator)
    b2 = np.linalg.det(b2_numerator) / np.linalg.det(main_denominator)
    b3 = np.linalg.det(b3_numerator) / np.linalg.det(main_denominator)


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(len(plan_matrix_x[0]))]
    gp = max(s) / sum(s)
    gt = 0.7679

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


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

    d = 4

    x = [[1, 1, 1, 1],
         [-1, -1, 1, 1],
         [-1, 1, -1, 1],
         [-1, 1, 1, -1]]

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

    bb = [sum([average_y[i] * x[j][i] for i in range(n)]) / n for j in
range(n)]
    t = [abs(bb[i]) / s_beta for i in range(n)]
    tt = 2.306
    blist = [b0, b1, b2, b3]
    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 = [blist[0] + blist[1] * plan_matrix_for_output[i][0] + blist[2] *
plan_matrix_for_output[i][1] + blist[3] * plan_matrix_for_output[i][2] for i
in range(4)]
    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'


m = 3
main()
kohren()
t_list = studens()
fisher = fish()

plan_table = BeautifulTable()
headers_x = ['X{}'.format(i) for i in range(1, m+1)]
headers_y = ['Y{}'.format(i) for i in range(1, m+1)]
plan_table.columns.header = [*headers_x, *headers_y]
for i in range(len(xn)):
    plan_table.rows.append([*plan_matrix_for_output[i], *y_matrix[i]])
print('Матриця планування:')
print(plan_table)

print('Normalized equations:')
result_y = []
for i in range(len(plan_matrix_x[0])):
    result_y.append(b0 + b1 * plan_matrix_for_output[i][0] + b2 *
plan_matrix_for_output[i][1] + b3 * plan_matrix_for_output[i][2])
    if round(result_y[i], 3) == round(average_y[i], 3):
        print('y = {} + {} * x1 + {} * x2 + {} * x3 = {} = {} - average
y'.format(round(b0, 3), round(b1, 3), round(b2, 3), round(b3, 3),
round(result_y[i], 3), average_y[i]))
    else:
        print('The obtained values do not coincide with the average values of
the response function')

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('Fp = ', round(fp, 3))
print(fisher)
```

Результати роботи програми

```
Матриця планування:
+-----+-----+----+-----+-----+-----+
| X1  | X2  | X3 | Y1  | Y2  | Y3  |
+-----+-----+----+-----+-----+-----+
| -30 | -35 | 0  | 208 | 181 | 203 |
+-----+-----+----+-----+-----+-----+
| -30 | 10  | 20 | 199 | 181 | 179 |
+-----+-----+----+-----+-----+-----+
|  0  | -35 | 20 | 188 | 179 | 191 |
+-----+-----+----+-----+-----+-----+
|  0  | 10  | 0  | 191 | 179 | 202 |
+-----+-----+----+-----+-----+-----+
Normalized equations:
y = 191.371 + -0.117 * x1 + -0.07 * x2 + -0.392 * x3 = 197.333 = 197.333 - average y
y = 191.371 + -0.117 * x1 + -0.07 * x2 + -0.392 * x3 = 186.333 = 186.333 - average y
y = 191.371 + -0.117 * x1 + -0.07 * x2 + -0.392 * x3 = 186.0 = 186.0 - average y
y = 191.371 + -0.117 * x1 + -0.07 * x2 + -0.392 * x3 = 190.667 = 190.667 - average y
Kohren check
Gp = 0.42 < 0.7679
Studens
t0 = 71.17
t1 = 0.655
t2 = 0.593
t3 = 1.466
Fisher
Fp =  1.051
The regression equation is adequate to the original at a significance level of 0.05

Process finished with exit code 0
```

Висновок

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

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

Контрольні запитання:

1. **Що називається дробовим факторним експериментом?**

ДФЕ – це частина ПФЕ, який мінімізує число дослідів, за рахунок тієї інформації, яка не дуже істотна для побудови лінійної моделі.

2. **Для чого потрібно розрахункове значення Кохрена?**

Критерій Кохрена використовують для порівняння трьох і більше виборок однакового обсягу n

3. **Для чого перевіряється критерій Стьюдента?**

За допомогою критерія Стьюдента перевіряється значущість коефіцієнта рівняння регресії.

Тобто, якщо виконується нерівність $t_s < t_{табл}$, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт βs є статистично незначущим і його слід виключити з рівняння регресії.

Якщо $t_s > t_{табл}$ то гіпотеза не підтверджується, тобто βs – значимий коефіцієнт і він залишається в рівнянні регресії.

4. **Чим визначається критерій Фішера і як його застосовувати?**

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності.

Адекватність моделі перевіряють за F-критерієм Фішера.

Якщо $F_{прак} < F_{теор}$, то отримана математична модель з прийнятим рівнем статистичної значимості q адекватна експериментальним даним.