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

ЗВІТ

з лабораторної роботи №3

з навчальної дисципліни «Методи Оптимізації та Планування Експеременту»

Виконав:

Студент 2 курсу кафедри ОТ ФІОТ,

Навчальної групи ІО-93

Верцанов С. С.

Перевірив:

Ассистент кафедри ОТ ФІОТ

Регіда П. Г.

Київ 2021

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

Варіант завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
305	-30	20	-25	10	-30	-15

Лістинг програми:

```
import random as rand
from prettytable import PrettyTable
import numpy as np
disp_not_homogen = True
while disp_not_homogen:
    x1_min = -30
    x1_max = 20
    x2_min = -25
    x2_max = 10
    x3_min = -30
    x3_max = -15
    y_max = 5
    y_min = -25
    exp_matrix = []
    y_average = []
    y_i = []
    exp_matrix_names = ['x₁', 'x₂', 'x₃', 'y₁', 'y₂', 'y₃']
    for i in range(4):
        y_norm = []
        for _ in range(3):
            y_norm.append(rand.randint(y_min, y_max))
        y_average.append(sum(y_norm) / 3)
        exp_matrix.append(y_norm)
        y_i.append(y_norm.copy())

    exp_matrix[0].insert(0, x3_min)
    exp_matrix[0].insert(0, x2_min)
    exp_matrix[0].insert(0, x1_min)
    exp_matrix[1].insert(0, x3_max)
    exp_matrix[1].insert(0, x2_max)
    exp_matrix[1].insert(0, x1_min)
    exp_matrix[2].insert(0, x3_max)
    exp_matrix[2].insert(0, x2_min)
    exp_matrix[2].insert(0, x1_max)
    exp_matrix[3].insert(0, x3_min)
    exp_matrix[3].insert(0, x3_max)
    exp_matrix[3].insert(0, x3_max)

    exp_table = PrettyTable()
    exp_table.field_names = exp_matrix_names
    exp_table.add_rows(exp_matrix)
```

```
    print('Normed matrix of experiment:')
    print(exp_table)

    mx_n = []
    my = sum(y_average) / 4
    for i in range(3):
        mx_i = 0
        for j in range(4):
            mx_i += exp_matrix[i][j]
        mx_n.append(mx_i)

    a1 = (exp_matrix[0][0] * y_average[0] + exp_matrix[0][1] * y_average[1] +
exp_matrix[0][2] * y_average[2] +
          exp_matrix[0][3] * y_average[3]) / 4
    a2 = (exp_matrix[1][0] * y_average[0] + exp_matrix[1][1] * y_average[1] +
exp_matrix[1][2] * y_average[2] +
          exp_matrix[1][3] * y_average[3]) / 4
    a3 = (exp_matrix[2][0] * y_average[0] + exp_matrix[2][1] * y_average[1] +
exp_matrix[2][2] * y_average[2] +
          exp_matrix[2][3] * y_average[3]) / 4

    a11 = (exp_matrix[0][0] ** 2 + exp_matrix[0][1] ** 2 + exp_matrix[0][2] ** 2 +
exp_matrix[0][3] ** 2) / 4
    a22 = (exp_matrix[1][0] ** 2 + exp_matrix[1][1] ** 2 + exp_matrix[1][2] ** 2 +
exp_matrix[1][3] ** 2) / 4
    a33 = (exp_matrix[2][0] ** 2 + exp_matrix[2][1] ** 2 + exp_matrix[2][2] ** 2 +
exp_matrix[2][3] ** 2) / 4

    a12 = (exp_matrix[0][0] * exp_matrix[1][0] + exp_matrix[0][1] * exp_matrix[1][1]
+ exp_matrix[0][2] *
           exp_matrix[1][2] + exp_matrix[0][3] * exp_matrix[1][3]) / 4
    a13 = (exp_matrix[2][0] ** 2 + exp_matrix[2][1] ** 2 + exp_matrix[2][2] ** 2 +
exp_matrix[2][3] ** 2) / 4
    a23 = (exp_matrix[1][0] * exp_matrix[2][0] + exp_matrix[1][1] * exp_matrix[2][1]
+ exp_matrix[1][2] *
           exp_matrix[2][2] + exp_matrix[1][3] * exp_matrix[2][3]) / 4

    b_01 = np.array([[my, mx_n[0], mx_n[1], mx_n[2]], [a1, a11, a12, a13], [a2, a12,
a22, a23], [a3, a13, a23, a33]])
    b_02 = np.array([[1, mx_n[0], mx_n[1], mx_n[2]], [mx_n[0], a11, a12, a13],
[mx_n[1], a12, a22, a23],
                     [mx_n[2], a13, a23, a33]])
    b_11 = np.array([[1, my, mx_n[1], mx_n[2]], [mx_n[0], a1, a12, a13], [mx_n[1],
a2, a22, a23],
                     [mx_n[2], a3, a23, a33]])
    b_21 = np.array([[1, mx_n[0], my, mx_n[2]], [mx_n[0], a11, a1, a13], [mx_n[1],
a12, a2, a23],
                     [mx_n[2], a13, a3, a33]])
    b_31 = np.array([[1, mx_n[0], mx_n[1], my], [mx_n[0], a11, a12, a1], [mx_n[1],
a12, a22, a2],
                     [mx_n[2], a13, a23, a3]])

    b0 = round(np.linalg.det(b_01)/np.linalg.det(b_02), 2)
    b1 = round(np.linalg.det(b_11)/np.linalg.det(b_02), 2)
    b2 = round(np.linalg.det(b_21)/np.linalg.det(b_02), 2)
    b3 = round(np.linalg.det(b_31)/np.linalg.det(b_02), 2)
    b_i = [b0, b1, b2, b3]
    print('\nNormed equation of regression: y = {} + {}x₁ + {}x₂ + {}x₃'.format(b0,
b1, b2, b3))

    S_y1 = ((y_i[0][0] - y_average[0])**2 + (y_i[0][1]-y_average[0])**2 + (y_i[0][2]-
y_average[0])**2)/3
```

```
    S_y2 = ((y_i[1][0] - y_average[1])**2 + (y_i[1][1]-y_average[1])**2 + (y_i[1][2]-
y_average[1])**2)/3
    S_y3 = ((y_i[2][0] - y_average[2])**2 + (y_i[2][1]-y_average[2])**2 + (y_i[2][2]-
y_average[2])**2)/3
    S_y4 = ((y_i[3][0] - y_average[3])**2 + (y_i[3][1]-y_average[3])**2 + (y_i[3][2]-
y_average[3])**2)/3
    S_yi = [S_y1, S_y2, S_y3, S_y4]
    Gp = max(S_yi)/sum(S_yi)
    if Gp < 0.7679:
        print('\nGp:', Gp, '\nDispersion is homogeneous with probability of 95% (by
Cohren)')
        disp_not_homogen = False
    S_b = sum(S_yi)/4
    S_beta_sqr = S_b/12
    S_beta = (S_b/12)**0.5
    beta_0 = sum(y_average)/4
    beta_1 = (-y_average[0] - y_average[1] + y_average[2] + y_average[3])/4
    beta_2 = (-y_average[0] + y_average[1] - y_average[2] + y_average[3])/4
    beta_3 = (-y_average[0] + y_average[1] + y_average[2] - y_average[3])/4
    t_s_i = [abs(beta_0)/S_beta, abs(beta_1)/S_beta, abs(beta_2)/S_beta,
abs(beta_3)/S_beta]

    d = 4

    for i in range(4):
        if t_s_i[i] < 2.306:
            print('\nb{} value is insignificant (by Student)'.format(i + 1))
            b_i[i] = 1
            d -= 1
    y_std_i = []
    for i in range(4):
        y_std_i.append(b_i[0] + b_i[1]*exp_matrix[i][0] + b_i[2]*exp_matrix[i][1] +
b_i[3]*exp_matrix[i][2])

    yij = 0
    for i in range(4):
        yij += (y_std_i[i] - y_average[i])**2

    F_p = (3 * (4 - d) * yij)/S_b
    if F_p > 4.5:
        print('F_p:', F_p, "\nRegression isn't adequate according to original (by
Fisher)")
    else:
        print('F_p:', F_p, "\nRegression adequate according to original ( by
Fisher)")
```

Висновок:

В даній лабораторній роботі я провів дробовий трьохфакторний експеримент з трьома статистичними перевірками і отримав коефіцієнти рівняння регресії.

Контрольні запитання:

1. Що називається дробовим факторним експериментом?
   Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту.

2. Для чого потрібно розрахункове значення Кохрена?
   Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3. Для чого перевіряється критерій Стьюдента?
   За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння регресії.

4. Чим визначається критерій Фішера і як його застосовувати?
   Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об’єкту.