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
Гришин О.С.
Номер у списку - 07

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.

107	-5	15	-15	35	15	30

Програмний код

```
from script import Experiment

if __name__ == '__main__':
    exp = Experiment([[-5, 15], [- 15, 35], [15, 30]])
    exp.show_info = True
    exp.make_experiment()
```

```
from random import *
from pprint import pprint
import numpy as np
from math import sqrt


class Experiment:
    m, N, d = 3, 4, 4
    mat_1X = [[1, -1, -1, -1],
              [1, -1, 1, 1],
              [1, 1, -1, 1],
              [1, 1, 1, -1]]
    show_info = False

    def __init__(self, x_list):
        """

        :param x_list: список иксов
        """

        """матрица иксов"""
        self.mat_sX = x_list
        self.mat_X = [[self.mat_sX[0][0], self.mat_sX[1][0],
self.mat_sX[2][0]],
                      [self.mat_sX[0][0], self.mat_sX[1][1],
```

```
self.mat_sX[2][1]],
                        [self.mat_sX[0][1], self.mat_sX[1][0],
self.mat_sX[2][1]],
                        [self.mat_sX[0][1], self.mat_sX[1][1],
self.mat_sX[2][0]]]

        """Т-матрица иксов"""
        self.trans_sx = [list(i) for i in zip(*self.mat_1X)]
        self.trans_x = [list(i) for i in zip(*self.mat_X)]

        self.x_min_max_av = [sum(self.mat_sX[i][k] for i in range(3)) / 3 for
k in range(2)]
        self.y_min_max = [int(200 + self.x_min_max_av[i]) for i in range(2)]

        """матрица игреков"""
        self.mat_Y = [[randint(self.y_min_max[0], self.y_min_max[1]) for _ in
range(self.m)] for _ in range(self.N)]

    def get_average_y(self):
        """середні значення функцій"""
        return [sum(self.mat_Y[k1]) / self.m for k1 in range(self.N)]

    def get_dispersion(self):
        return [sum([((k1 - self.get_average_y()[j]) ** 2) for k1 in
self.mat_Y[j]]) / self.m for j in range(self.N)]

    def show_dets(self, check, b_list):
        print('Матрица X')
        pprint(self.mat_X)
        print('Матрица Y')
        pprint(self.mat_Y)

        print(f"\nУравнение регресии\ny = {b_list[0]} + {b_list[1]}*x1 +
{b_list[2]}*x2 + {b_list[3]}*x3")
        print("Дисперсии:")
        print(self.get_dispersion())

        print('Среднее Y')
        print(self.get_average_y())
        print("Проверка сравнением со средним Y:\n", check)
        print()

    def get_mx_my(self):
        """Знайдемо коефіцієнти рівняння регресії"""
        mx = [sum(self.mat_X[i][k] for i in range(self.N)) / self.N for k in
range(self.m)]
        my = sum(self.get_average_y()) / self.N
        return mx, my

    def get_ai_aii(self):
        ai = [sum(self.trans_x[k][i] * self.get_average_y()[i] for i in
range(self.N)) / self.N for k in range(self.m)]
        aii = [sum(self.trans_x[k][i] ** 2 for i in range(self.N)) / self.N
for k in range(self.m)]
        return ai, aii

    def find_cf(self):
        tran = self.trans_x
        mx, my = self.get_mx_my()
        ai, aii = self.get_ai_aii()

        a12 = a21 = (tran[0][0] * tran[1][0] + tran[0][1] * tran[1][1] +
tran[0][2] * tran[1][2] + tran[0][3] * tran[1][3]) / self.N
        a13 = a31 = (tran[0][0] * tran[2][0] + tran[0][1] * tran[2][1] +
```

```
tran[0][2] * tran[2][2] + tran[0][3] * tran[2][3]) / self.N
        a23 = a32 = (tran[1][0] * tran[2][0] + tran[1][1] * tran[2][1] +
tran[1][2] * tran[2][2] + tran[1][3] * tran[2][3]) / self.N
        znamen = np.linalg.det(np.matrix(
            [[1, mx[0], mx[1], mx[2]],
             [mx[0], aii[0], a12, a13],
             [mx[1], a12, aii[1], a32],
             [mx[2], a13, a23, aii[2]]]))

        b0 = np.linalg.det(np.matrix([[my, mx[0], mx[1], mx[2]],
                                      [ai[0], aii[0], a12, a13],
                                      [ai[1], a12, aii[1], a32],
                                      [ai[2], a13, a23, aii[2]]])) / znamen

        b1 = np.linalg.det(np.matrix([[1, my, mx[1], mx[2]],
                                      [mx[0], ai[0], a12, a13],
                                      [mx[1], ai[1], aii[1], a32],
                                      [mx[2], ai[2], a23, aii[2]]])) / znamen

        b2 = np.linalg.det(np.matrix([[1, mx[0], my, mx[2]],
                                      [mx[0], aii[0], ai[0], a13],
                                      [mx[1], a12, ai[1], a32],
                                      [mx[2], a13, ai[2], aii[2]]])) / znamen

        b3 = np.linalg.det(np.matrix([[1, mx[0], mx[1], my],
                                      [mx[0], aii[0], a12, ai[0]],
                                      [mx[1], a12, aii[1], ai[1]],
                                      [mx[2], a13, a23, ai[2]]])) / znamen

        check = [b0 + b1 * tran[0][i] + b2 * tran[1][i] + b3 * tran[2][i] for
i in range(4)]
        b_list = [b0, b1, b2, b3]
        return check, b_list

    def make_experiment(self):
        mat_disY = self.get_dispersion()
        check, b_list = self.find_cf()
        if self.show_info:
            self.show_dets(check, b_list)

        print('\nПроверка однородности за Кохрена:')
        if max(mat_disY) / sum(mat_disY) < 0.7679:
            print('Дисперсия однородная')
        else:
            print('Дисперсия неоднородная')

        print('\nПроверка значимости:\n')
        S2b = sum(mat_disY) / self.N
        S2bs = S2b / (self.m * self.N)
        Sbs = sqrt(S2bs)
        bb = [sum(self.get_average_y()[k] * self.trans_sx[i][k] for k in
range(self.N))/self.N for i in range(self.N)]
        t = [abs(bb[i])/Sbs for i in range(self.N)]
        if self.show_info:
            print('Sbs:\n', Sbs)
            print('bi:\n', bb, '\nti:\n', t)
        for i in range(self.N):
            if t[i] < 2.306:
                print('Незначительный ', b_list[i])
                b_list[i] = 0
                self.d -= 1

        y_reg = [b_list[0] + b_list[1] * self.mat_X[i][0] + b_list[2] *
self.mat_X[i][1] + b_list[3] * self.mat_X[i][2] for i in range(self.N)]
```

```
        print('Значения y:\n')
        [print(f"{b_list[0]} + {b_list[1]}*x1 + {b_list[2]}*x2 +
{b_list[3]}*x3 = {b_list[0] + b_list[1] * self.mat_X[i][0] + b_list[2] *
self.mat_X[i][1] + b_list[3] * self.mat_X[i][2]}") for i in range(self.N)]

        print('\nПроверка адекватности за Фишера:\n')
        Sad = (self.m / (self.N - self.d)) * int(sum(y_reg[i] -
self.get_average_y()[i] for i in range(self.N)) ** 2)
        Fp = Sad / S2b
        print('FP  =', Fp)
        if Fp > 4.5:
            print('Неадекватно при 0.05')
        else:
            print('Адекватно при 0.05')
```

Результати роботи програми

```
Матрица X
[[-5, -15, 15], [-5, 35, 30], [15, -15, 30], [15, 35, 15]]
Матрица Y
[[210, 211, 211], [215, 205, 199], [208, 221, 209], [217, 200, 198]]

Уравнение регресии
y = 207.2833333333335 + 0.016666666666666004*x1 + -0.12000000000000012*x2 + 0.11111111111111992*x3
Дисперсии:
[0.22222222222222224, 43.55555555555555, 34.888888888888886, 72.66666666666667]
Среднее Y
[210.66666666666666, 206.33333333333334, 212.66666666666666, 205.0]
Проверка сравнением со средним Y:
 [210.66666666666697, 206.33333333333374, 212.6666666666671, 205.00000000000028]

 Проверка однородности за Кохрена:
 Дисперсия однородная

 Проверка значимости:

 Незначительный  0.016666666666666004
 Незначительный  -0.12000000000000012
 Незначительный  0.11111111111111992
 Значения y:

 207.2833333333335 + 0*x1 + 0*x2 + 0*x3 = 207.2833333333335
 207.2833333333335 + 0*x1 + 0*x2 + 0*x3 = 207.2833333333335
 207.2833333333335 + 0*x1 + 0*x2 + 0*x3 = 207.2833333333335
 207.2833333333335 + 0*x1 + 0*x2 + 0*x3 = 207.2833333333335

 Проверка адекватности за Фишера:

 FP  = 0.7929515418502203
 Адекватно при 0.05
```

Контрольні запитання:

1. **Дробовий факторний експеримент – частина ПФЕ, який мінімізує число дослідів, за рахунок тієї інформації, яка не дуже істотна для побудови моделі**
2. **Значення Кохрена використовують для перевірки однорідності дисперсії**
3. **Критерій Стьюдента перевіряє значущість коефіцієнтів рівняння**
4. **Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об’єкту**