Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №4**

«Проведення трьохфакторного експерименту

при використанні рівняння регресії з урахуванням ефекту взаємодії**»**

**Виконав:**

студент II курсу ФІОТ

групи ІВ-91

Бурбело Сергій

**Перевірив:**

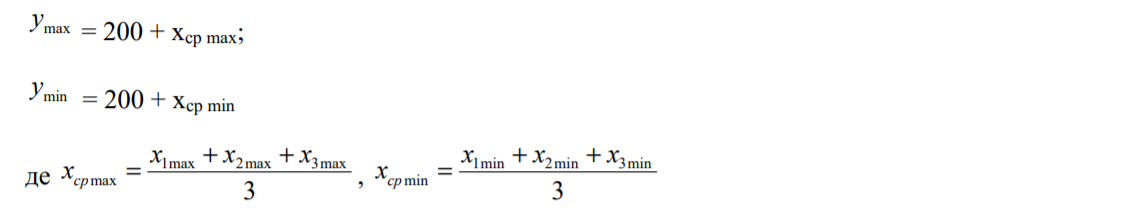
Регіда П.Г.

Київ – 2021

**Мета роботи:** Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

**Завдання на лабораторну роботу:**

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.

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

3. Провести 3 статистичні перевірки.

4. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.

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

**Варіант завдання:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Варіант | X1 | | X2 | | X3 | |
| min | max | min | max | min | max |
| 104 | -20 | 30 | 30 | 80 | 45 | 50 |

Довірча ймовірність дорівнює 0.95, а рівень значимості q = 0.05.

**Роздруківка тексту програми:**

from prettytable import PrettyTable

import sklearn.linear\_model as lm

from itertools import compress

from \_pydecimal import Decimal

from functools import partial

from scipy.stats import f, t

from functools import reduce

from random import randint

import math, os, sys

import numpy as np

import math

class Experiment:

    def \_\_init\_\_(self, n, m):

        self.n = n

        self.m = m

        self.f1 = self.m - 1

        self.f2 = self.n

        self.f3 = self.f1\*self.f2

        self.p = 0.95

        self.q = 1 - self.p

        self.N = [i+1 for i in range(self.n+1)]

        self.x\_min = [-20, 30, 30]

        self.x\_max = [30, 80, 45]

        self.average\_x\_min = round(np.average(self.x\_min))

        self.average\_x\_max = round(np.average(self.x\_max))

        self.y\_min = 200 + self.average\_x\_min

        self.y\_max = 200 + self.average\_x\_max

        self.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],

                [-1,1,1,-1,1,-1,-1,1]]

        self.x12 = [self.norm\_x[1][i]\*self.norm\_x[2][i] for i in range(self.n)]

        self.x13 = [self.norm\_x[1][i]\*self.norm\_x[3][i] for i in range(self.n)]

        self.x23 = [self.norm\_x[2][i]\*self.norm\_x[3][i] for i in range(self.n)]

        self.x123 = [self.norm\_x[1][i]\*self.norm\_x[2][i]\*self.norm\_x[3][i] for i in range(self.n)]

        self.norm\_x += [self.x12, self.x13, self.x23, self.x123]

        self.norm\_xt = np.array(self.norm\_x)

        self.norm\_xt = self.norm\_xt.transpose()

        self.factors\_x = [[-20,-20,30,30,-20,-20,30,30],

                        [30,80,30,80,30,80,30,80],

                        [30,45,45,30,45,30,30,45]]

        self.xf12 = [self.factors\_x[0][i]\*self.factors\_x[1][i] for i in range(self.n)]

        self.xf13 = [self.factors\_x[0][i]\*self.factors\_x[2][i] for i in range(self.n)]

        self.xf23 = [self.factors\_x[1][i]\*self.factors\_x[2][i] for i in range(self.n)]

        self.xf123 = [self.factors\_x[0][i]\*self.factors\_x[1][i]\*self.factors\_x[2][i] for i in range(self.n)]

        self.factors\_x += [self.xf12, self.xf13, self.xf23, self.xf123]

        self.factors\_xt = np.array(self.factors\_x)

        self.factors\_xt = self.factors\_xt.transpose()

        # Матриця відгуків:

        self.y\_t = np.array([[randint((self.y\_min), (self.y\_max)) for i in range(self.n)] for j in range(self.m)])

        self.y = self.y\_t.transpose()

        # Cереднє значення функції відгуку в рядку:

        self.av\_y = [round(sum(i) / len(i), 2) for i in self.y]

        # Дисперсії по рядках:

        self.S2 = [round(np.var(i),2) for i in self.y]

        # Натуралізовані значення Хів та у:

        self.x1 = np.array(list(zip(\*self.factors\_xt))[0])

        self.x2 = np.array(list(zip(\*self.factors\_xt))[1])

        self.x3 = np.array(list(zip(\*self.factors\_xt))[2])

        self.natural\_bi = self.naturalizedB(self.n, self.x1, self.x2, self.x3, self.av\_y)

        # +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        # Вивід даних:

        # +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        print("ŷ = b0 + b1\*x1 + b2\*x2 + b3\*x3 + b12\*x1\*x2 + b13\*x1\*x3 + b23\*x2\*x3 + b123\*x1\*x2\*x3")

        # Нормована матриця планування експерименту

        th = ["N", "x0", "x1", "x2", "x3", "x1\*x2", "x1\*x3", "x2\*x3", "x1\*x2\*x3"]

        th += ["y"+str(i+1) for i in range(self.m)]

        th.append("<y>")

        th.append("S^2")

        columns = len(th)

        table = PrettyTable(th)

        table.title = "Нормована матриця планування експерименту"

        for i in range(self.n):

            td = [self.N[i], self.norm\_x[0][i], self.norm\_x[1][i], self.norm\_x[2][i], self.norm\_x[3][i], \

                    self.x12[i], self.x13[i], self.x23[i], self.x123[i]]

            td += [self.y\_t[j][i] for j in range(self.m)]

            td.append(self.av\_y[i])

            td.append(self.S2[i])

            td\_data = td[:]

            while td\_data:

                table.add\_row(td\_data[:columns])

                td\_data = td\_data[columns:]

        print(table)

        # Матриця планування експерименту

        th = ["N", "x1", "x2", "x3", "x1\*x2", "x1\*x3", "x2\*x3", "x1\*x2\*x3"]

        th += ["y"+str(i+1) for i in range(self.m)]

        th.append("<y>")

        th.append("S^2")

        columns = len(th)

        table = PrettyTable(th)

        table.title = "Матриця планування експерименту"

        for i in range(self.n):

            td = [self.N[i], self.factors\_x[0][i], self.factors\_x[1][i], self.factors\_x[2][i], \

                self.xf12[i], self.xf13[i], self.xf23[i], self.xf123[i]]

            td += [self.y\_t[j][i] for j in range(self.m)]

            td.append(self.av\_y[i])

            td.append(self.S2[i])

            td\_data = td[:]

            while td\_data:

                table.add\_row(td\_data[:columns])

                td\_data = td\_data[columns:]

        print(table)

        self.kohrenCriteriy(self.m, self.n, self.y, self.p, self.q, self.f1, self.f2)

        self.studentCriteriy(self.m, self.n, self.y, self.av\_y, self.norm\_xt, self.f3, self.q)

        self.fisherCriteriy(self.m, self.n, 1, self.f3, self.q, self.factors\_xt, self.y, self.natural\_bi, self.y\_x)

    def naturalizedB(self, n, x1, x2, x3, av\_y):

        def m\_ij(\*arrays):

            return np.average(reduce(lambda accum, el: accum\*el, arrays))

        koef = [[n, m\_ij(x1), m\_ij(x2), m\_ij(x3), m\_ij(x1\*x2), m\_ij(x1\*x3), m\_ij(x2\*x3), m\_ij(x1\*x2\*x3)],

                [m\_ij(x1), m\_ij(x1\*\*2), m\_ij(x1\*x2), m\_ij(x1\*x3), m\_ij(x1\*\*2\*x2), m\_ij(x1\*\*2\*x3), m\_ij(x1\*x2\*x3), m\_ij(x1\*\*2\*x2\*x3)],

                [m\_ij(x2), m\_ij(x1\*x2), m\_ij(x2\*\*2), m\_ij(x2\*x3), m\_ij(x1\*x2\*\*2), m\_ij(x1\*x2\*x3), m\_ij(x2\*\*2\*x3), m\_ij(x1\*x2\*\*2\*x3)],

                [m\_ij(x3), m\_ij(x1\*x3), m\_ij(x2\*x3), m\_ij(x3\*\*2), m\_ij(x1\*x2\*x3), m\_ij(x1\*x3\*\*2), m\_ij(x2\*x3\*\*2), m\_ij(x1\*x2\*x3\*\*2)],

                [m\_ij(x1\*x2), m\_ij(x1\*\*2\*x2), m\_ij(x1\*x2\*\*2), m\_ij(x1\*x2\*x3), m\_ij(x1\*\*2\*x2\*\*2), m\_ij(x1\*\*2\*x2\*x3), m\_ij(x1\*x2\*\*2\*x3), m\_ij(x1\*\*2\*x2\*\*2\*x3)],

                [m\_ij(x1\*x3), m\_ij(x1\*\*2\*x3), m\_ij(x1\*x2\*x3), m\_ij(x1\*x3\*\*2), m\_ij(x1\*\*2\*x2\*x3), m\_ij(x1\*\*2\*x3\*\*2), m\_ij(x1\*x2\*x3\*\*2), m\_ij(x1\*\*2\*x2\*x3\*\*2)],

                [m\_ij(x2\*x3), m\_ij(x1\*x2\*x3), m\_ij(x2\*\*2\*x3), m\_ij(x2\*x3\*\*2), m\_ij(x1\*x2\*\*2\*x3), m\_ij(x1\*x2\*x3\*\*2), m\_ij(x2\*\*2\*x3\*\*2), m\_ij(x1\*x2\*\*2\*x3\*\*2)],

                [m\_ij(x1\*x2\*x3), m\_ij(x1\*\*2\*x2\*x3), m\_ij(x1\*x2\*\*2\*x3), m\_ij(x1\*x2\*x3\*\*2), m\_ij(x1\*\*2\*x2\*\*2\*x3), m\_ij(x1\*\*2\*x2\*x3\*\*2), m\_ij(x1\*x2\*\*2\*x3\*\*2), m\_ij(x1\*\*2\*x2\*\*2\*x3\*\*2)]]

        free\_vector = [m\_ij(av\_y), m\_ij(av\_y\*x1), m\_ij(av\_y\*x2), m\_ij(av\_y\*x3), m\_ij(av\_y\*x1\*x2), m\_ij(av\_y\*x1\*x3), m\_ij(av\_y\*x2\*x3), m\_ij(av\_y\*x1\*x2\*x3)]

        natural\_bi = np.linalg.solve(koef, free\_vector)

        return natural\_bi

    # -------------------------------------------------------

    # Перевірка однорідності дисперсії за критерієм Кохрена:

    # -------------------------------------------------------

    def kohrenCriteriy(self, m, n, y, p, q, f1, f2):

        # Знайдемо дисперсії по рядках:

        S = [np.var(i) for i in y]

        # Знайдемо критерій Кохрена:

        Gp = max(S)/sum(S)

        # Табличне значення критерію Кохрена:

        q\_ = q / f2

        khr = f.ppf(q=1-q\_, dfn=f1, dfd=(f2 - 1) \* f1)

        Gt = khr / (khr + f2 - 1)

        print("Критерій Кохрена: Gr = " + str(round(Gp,3)))

        # Рівень значимості приймемо 0.05.

        # Перевірка рівняння на однорідність:

        if Gp < Gt:

            print("Дисперсії однорідні з вірогідністю 95%.")

            pass

        else:

            print("\nДисперсії не однорідні.\nПроводимо експеремент для m+=1\n")

            Experiment(self.n, self.m+1)

    # ---------------------------------------------------------

    # Перевірка однорідності дисперсії за критерієм Стьюдента:

    # ---------------------------------------------------------

    def studentCriteriy(self, m, n, y, av\_y, norm\_xt, f3, q):

        av\_S = np.average(list(map(np.var, y)))

        s2\_beta = av\_S/n/m

        s\_beta = math.sqrt(s2\_beta)

        xi = np.array([[el[i] for el in norm\_xt] for i in range(len(norm\_xt))])

        k\_beta = np.array([round(np.average(av\_y\*xi[i]),3) for i in range(len(xi))])

        # Значення критерію Стьюдента:

        T = np.array([abs(k\_beta[i])/s\_beta for i in range(len(k\_beta))])

        # Табличне значення критерію Стьюдента:

        T\_tabl = t.ppf(q=1-q, df=f3)

        # Вивід:

        print("\nКритерій Стьюдента:")

        T\_ = list(map(lambda i: "{:.2f}".format(i), T))

        for i in T\_: print(str(i))

        imp = [i if i > T\_tabl else 0 for i in T]

        index\_list = []

        b = ["b0", "b1", "b2", "b3", "b4", "b12", "b13", "b23", "b123"]

        index\_list = [i for i, x in enumerate(imp) if x == 0]

        index\_list = [b[i] for i in index\_list]

        deleted\_koef = ', '.join(index\_list) + " - коефіцієнти рівняння регресії приймаємо незначними, виключаємо їх з рівняння. "

        print(deleted\_koef)

        self.y\_x = [True if i > T\_tabl else False for i in T]

        x\_i = list(compress(["", "\*x1", "\*x2", "\*x3", "\*x12", "\*x13", "\*x23", "\*x123"], self.y\_x))

        p = list(compress(k\_beta, self.y\_x))

        y = " ".join(["".join(i) for i in zip(list(map(lambda x: "{:+.2f}".format(x), p)),x\_i)])

        print("Рівняння регресії: y = " + y)

    # -------------------------------------------------------

    # Перевірка однорідності дисперсії за критерієм Фішера:

    # -------------------------------------------------------

    def fisherCriteriy(self, m, n, d, f3, q, natural\_x, y\_t, b\_k, imp):

        f4 = n - d

        table = [list(compress(row, imp)) for row in natural\_x]

        b\_k = list(compress(b\_k, imp))

        y\_vals = np.array([sum(map(lambda x, b: x\*b, row, b\_k)) for row in natural\_x])

        y\_averages = np.array(list(map(np.average, y\_t)))

        s\_ad = m/(n-d)\*(sum((y\_vals-y\_averages)\*\*2))\*0.001

        y\_variations = np.array(list(map(np.var, y\_t)))

        s\_v = np.average(y\_variations)

        # Критерій Фішера:

        Fp = s\_ad/s\_v

        print("\nКритерій Фішера: Fp = "+ str(round(Fp,4)))

        # Табличне значення критерія Фішера:

        F\_tabl = f.isf(q,f4,f3)

        print("Табличне значення критерія Фішера: Ft = "+ str(round(F\_tabl,4)))

        if Fp < F\_tabl:

            print("\nРівняння регресії адекватно оригіналу.")

            pass

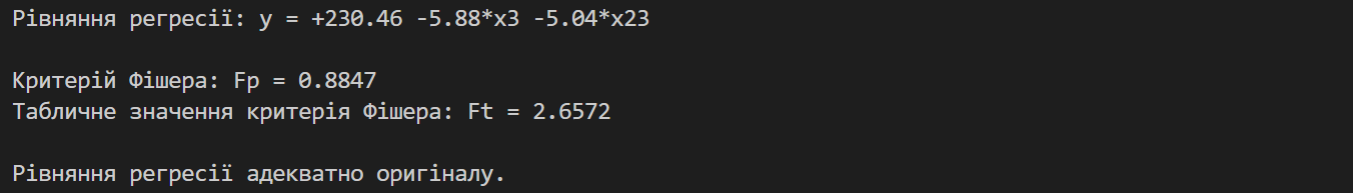
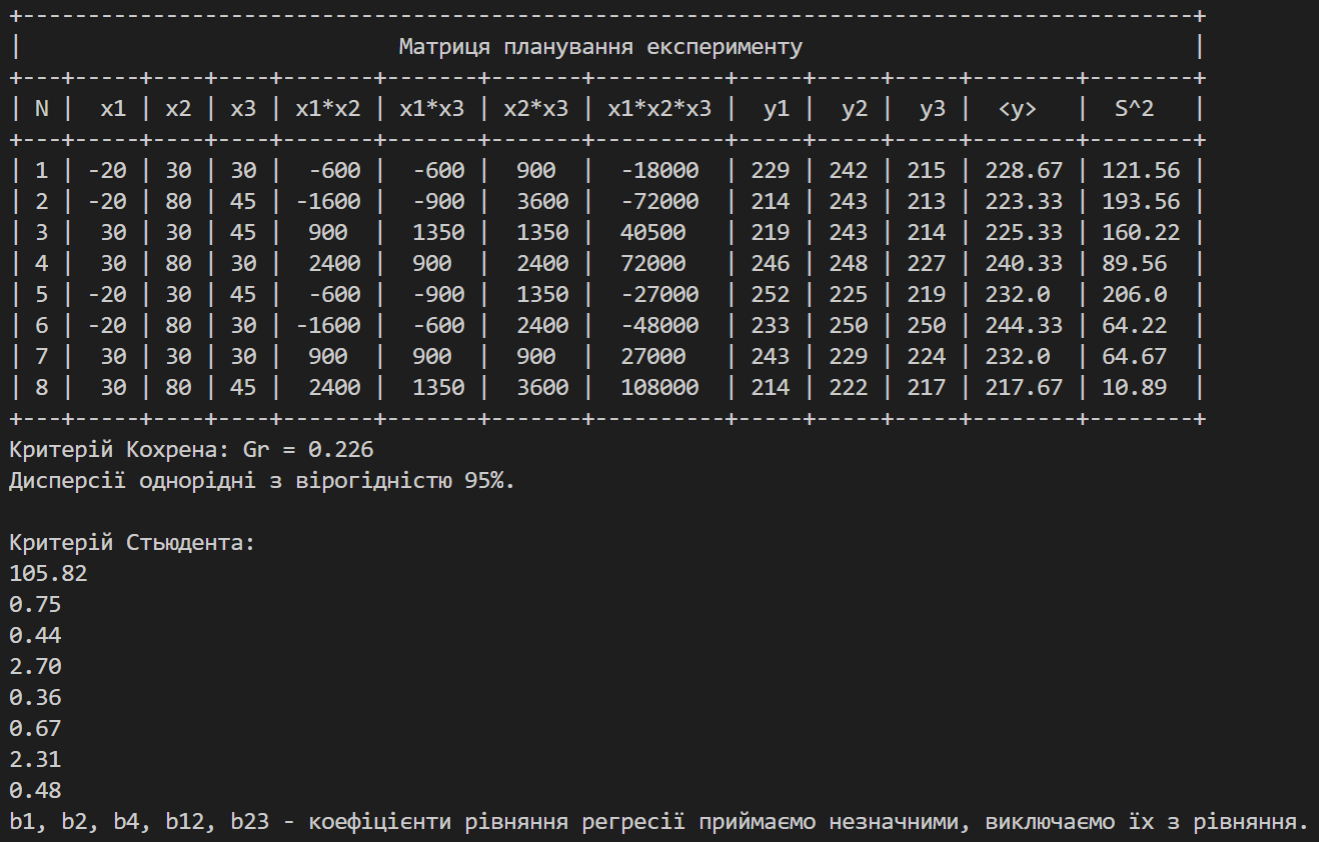
        else:

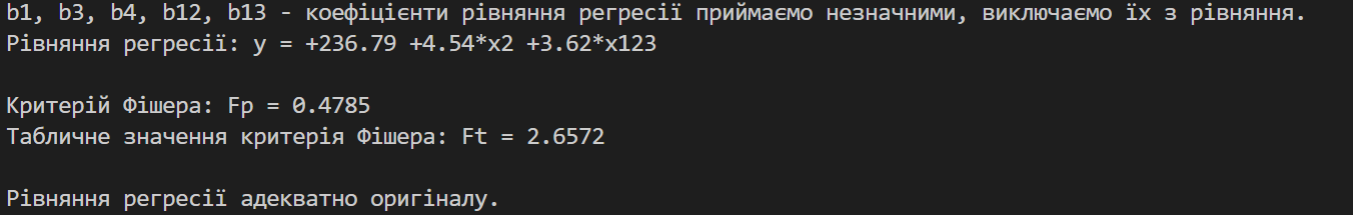
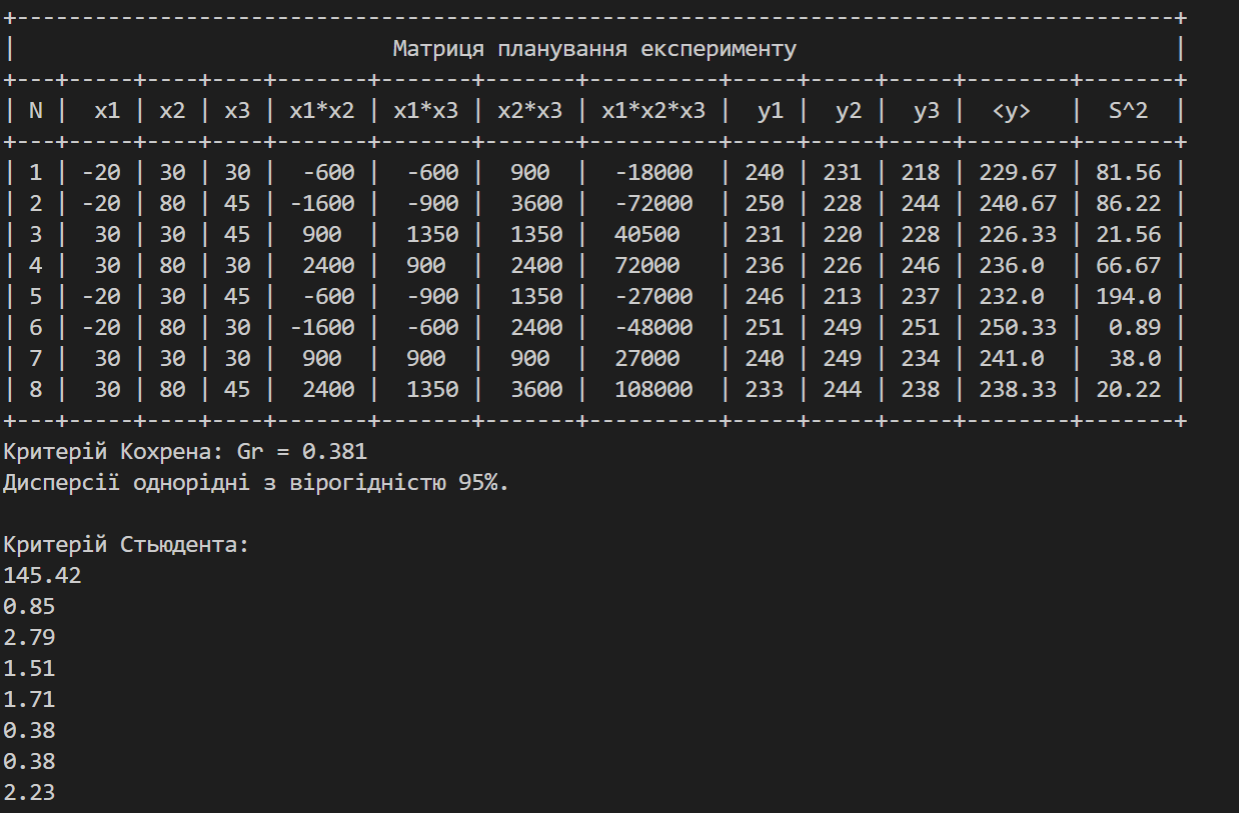
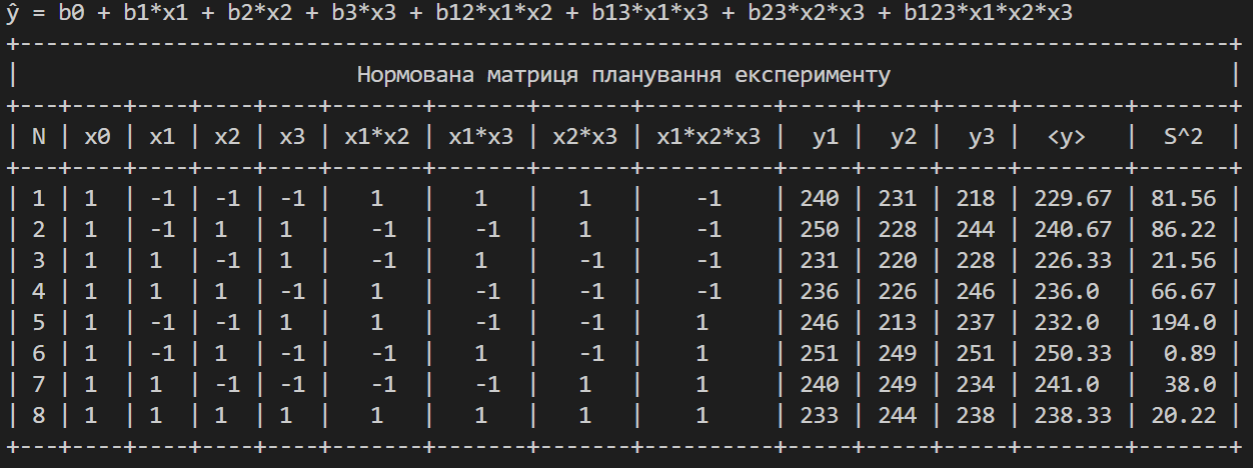
            print("\nРівняння регресії НЕ адекватно оригіналу. >>> m+=1\n")

            Experiment(self.n, self.m+1)

Experiment(8, 3)

**Результати роботи програми:**





**Висновок:**

У ході лабораторної роботи я змоделював трьохфакторний експеримент з використанням лінійного рівняння регресії, склав матрицю планування експерименту, визначив коефіцієнти рівняння регресії, натуралізовані та нормовані, виконав перевірку правильності розрахунку коефіцієнтів рівняння регресії. Також я реалізував 3 статистичні перевіркиза критерієм Кохрена, Стьюдента та Фішера. При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів.