Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №4

з дисципліни «Методи оптимізації та планування експерименту»

на тему: «*ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ****»***

Виконав:
студент 2-го курсу ФІОТ
групи ІО-82
Скібінський А.О.
Номер у списку: 19
Перевірив:
Регіда П.Г.

Київ – 2020

Варіант:

219	-20	30	-35	15	-20	5

Лістинг коду програми:

```
import numpy as np

def lab4(m, N):
    A = np.random.randint(Ymin, Ymax, (N, m))
    print("Згенерована матриця значень Y: ")
    for row in A:
        for i in row:
            print("{:4d}".format(int(i)), end=" |")
        print()

    Y = np.sum(A, axis=1) / m

    b = []
    for i in range(N):
        S = 0
        for j in range(N):
            S += (x[j][i] * Y[j]) / N
        b.append(round(S, 3))
    print("Рівняння регресії:")
    print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 + {}*x2x3 + {}*x1x2x3
\n".format(b[0], b[1], b[2],

b[3], b[4], b[5],

b[6], b[7]))
    solve_y(b)

    print("Перевірка однорідності дисперсії за критерієм Кохрена:")
    D = []
    for i in range(N):
        Di = sum([(j - Y[i]) ** 2 for j in A[i]]) / m
        D.append(round(Di, 3))

    Dmax = max(D)
    Dsum = sum(D)
    Gp = Dmax / Dsum
    print("Коефіцієнт Gp = ", round(Gp, 5))

    f1 = m - 1
    f2 = N
    print("f1 = ", f1)
    print("f2 = ", f2)

    Gtable = {3: 0.4377, 4: 0.3910, 5: 0.3595, 6: 0.3362, 7: 0.3185, 8: 0.3043, 9:
0.2926,
              10: 0.2829, range(11, 17): 0.2462, range(17, 37): 0.2022, range(37,
145): 0.1616}
    Gt = Gtable.get(m)
    print("За таблицею Gt = ", Gt)

    while(Gp < Gt):
        print("Gp < Gt, отже дисперсія однорідна. Критерій Кохрена виконується")

        print("Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента:")
```

```
        mD = Dsum / N
        Db = mD / (m * N)
        sD = Db ** 0.5
        print("Дисперсія відносності Db = ", round(Db, 3))
        print("sD = ", round(sD, 3))
        print("Оцінка за t-критерієм Стьюдента:")
        t = []
        for i in range(N):
            T = abs(b[i]) / sD
            t.append(round(T, 3))
        f3 = f1 * f2
        print("f3 = ", f3)
        Ttabl = 2.120
        print("За таблицею в 16 рядку Ttabl = ", Ttabl)
        d = 0
        for i in range(N):
            if (t[i] < Ttabl):
                print("Коефіцієнт b{} є статистично незначущим, виключаємо його з
рівняння регресії".format(i))
                b[i] = 0
            else:
                d += 1
                print(
                    "Гіпотеза не підтверджується, тобто b{} - значимий коефіцієнт і
він залишається в рівнянні регресії."
                    .format(i))
        y = solve_y(b)

        print("Перевірка адекватності за критерієм Фішера:")
        print("Кількість значущих коефіцієнтів d = ", d)
        Dad = 0
        for i in range(N):
            Dad += (m / (N - d)) * ((y[i] - Y[i]) ** 2)
        print("Дисперсія адекватності Dad = ", round(Dad, 3))
        Fp = Dad / Db
        print("Перевірка адекватності Fp = ", round(Fp, 3))
        f4 = N - d
        print("f4 = ", f4)
        Ftable = {1: 4.5, 2: 3.6, 3: 3.2, 4: 3.0, 5: 2.9, 6: 2.7, 7: 2.4}
        Ft = Ftable.get(f4)
        print("За таблицею Ft = ", Ft)
        if (Fp < Ft):
            print("Fp < Ft, отримана математична модель адекватна експериментальним
даним.")
            break
        else:
            print("Fp > Ft, отже, рівняння регресії неадекватно оригіналу")
            break

    else:
        print("Gp > Gt, отже дисперсія неоднорідна. Збільшуємо кількість дослідів на
1 ")
        m = m + 1
        lab4(m, N)
        return



def solve_y(b):
    print("Значення y:")
    y1 = b[0] + b[1] * X[0][0] + b[2] * X[0][1] + b[3] * X[0][2] + b[4] * X[0][0] *
X[0][1] + b[5] * X[0][0] * X[0][2] + \
```

```
        b[6] * X[0][1] * X[0][2] + b[7] * X[0][0] * X[0][1] * X[0][2]
    y2 = b[0] + b[1] * X[1][0] + b[2] * X[1][1] + b[3] * X[1][2] + b[4] * X[1][0] *
X[1][1] + b[5] * X[1][0] * X[1][2] + \
        b[6] * X[1][1] * X[1][2] + b[7] * X[1][0] * X[1][1] * X[1][2]
    y3 = b[0] + b[1] * X[2][0] + b[2] * X[2][1] + b[3] * X[2][2] + b[4] * X[2][0] *
X[2][1] + b[5] * X[2][0] * X[2][2] + \
        b[6] * X[2][1] * X[2][2] + b[7] * X[2][0] * X[2][1] * X[2][2]
    y4 = b[0] + b[1] * X[3][0] + b[2] * X[3][1] + b[3] * X[3][2] + b[4] * X[3][0] *
X[3][1] + b[5] * X[3][0] * X[3][2] + \
        b[6] * X[3][1] * X[3][2] + b[7] * X[3][0] * X[3][1] * X[3][2]
    y5 = b[0] + b[1] * X[4][0] + b[2] * X[4][1] + b[3] * X[4][2] + b[4] * X[4][0] *
X[4][1] + b[5] * X[4][0] * X[4][2] + \
        b[6] * X[4][1] * X[4][2] + b[7] * X[4][0] * X[4][1] * X[4][2]
    y6 = b[0] + b[1] * X[5][0] + b[2] * X[5][1] + b[3] * X[5][2] + b[4] * X[5][0] *
X[5][1] + b[5] * X[5][0] * X[5][2] + \
        b[6] * X[5][1] * X[5][2] + b[7] * X[5][0] * X[5][1] * X[5][2]
    y7 = b[0] + b[1] * X[6][0] + b[2] * X[6][1] + b[3] * X[6][2] + b[4] * X[6][0] *
X[6][1] + b[5] * X[6][0] * X[6][2] + \
        b[6] * X[6][1] * X[6][2] + b[7] * X[6][0] * X[6][1] * X[6][2]
    y8 = b[0] + b[1] * X[7][0] + b[2] * X[7][1] + b[3] * X[7][2] + b[4] * X[7][0] *
X[7][1] + b[5] * X[7][0] * X[7][2] + \
        b[6] * X[7][1] * X[7][2] + b[7] * X[7][0] * X[7][1] * X[7][2]
    y = [y1, y2, y3, y4, y5, y6, y7, y8]
    print("y1 = {}, y2 = {}, y3 = {}, y4 = {}, y5 = {}, y6 = {}, y7 = {}, y8 =
{}\n".format(y1, y2, y3, y4, y5, y6, y7, y8))
    return y


X1max = 30
X1min = -20
X2max = 15
X2min = -35
X3max = 5
X3min = -20
Ymax = 200 + (X1max + X2max + X3max)/3
Ymin = 200 + (X1min + X2min + X3min)/3
X = [[X1min, X2min, X3min],
     [X1min, X2min, X3max],
     [X1min, X2max, X3min],
     [X1min, X2max, X3max],
     [X1max, X2min, X3min],
     [X1max, X2min, X3max],
     [X1max, X2max, X3min],
     [X1max, X2max, X3max]]

nX1min = nX2min = nX3min = -1
nX1max = nX2max = nX3max = 1
x = [[1, nX1min, nX2min, nX3min, nX1min * nX2min, nX1min * nX3min, nX2min * nX3min,
nX1min * nX2min * nX3min],
     [1, nX1min, nX2min, nX3max, nX1min * nX2min, nX1min * nX3max, nX2min * nX3max,
nX1min * nX2min * nX3max],
     [1, nX1min, nX2max, nX3min, nX1min * nX2max, nX1min * nX3min, nX2max * nX3min,
nX1min * nX2max * nX3min],
     [1, nX1min, nX2max, nX3max, nX1min * nX2max, nX1min * nX3max, nX2max * nX3max,
nX1min * nX2max * nX3max],
     [1, nX1max, nX2min, nX3min, nX1max * nX2min, nX1max * nX3min, nX2min * nX3min,
nX1max * nX2min * nX3min],
     [1, nX1max, nX2min, nX3max, nX1max * nX2min, nX1max * nX3max, nX2min * nX3max,
nX1max * nX2min * nX3max],
     [1, nX1max, nX2max, nX3min, nX1max * nX2max, nX1max * nX3min, nX2max * nX3min,
nX1max * nX2max * nX3min],
     [1, nX1max, nX2max, nX3max, nX1max * nX2max, nX1max * nX3max, nX2max * nX3max,
```

```
nX1max * nX2max * nX3max]]

m = 3
N = 8
print("Кодовані значення факторів: ")
for row in x:
      for i in row:
          print("{:4d}".format(int(i)), end = " |")
      print()
print("Натуральні значення факторів: ")
for row in X:
      for i in row:
          print("{:4d}".format(int(i)), end = " |")
      print()

lab4(m, N)
```