Міністерство освіти та науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА №3

з дисципліни «Методи оптимізації та планування експерименту» на тему:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконав:
студент ІІ курсу ФІОТ
групи ІВ-81 :
Бухтій О. В.
Перевірив:
Регіда П. Г.

Київ-2020

ЛАБОРАТОРНА РОБОТА №3.

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

Завдання

	x_1		x_2		x_3	
	min	max	min	max	min	max
107	-5	15	-15	35	15	30

$Y_{max} = 226$
$Y_{min} = 199$

Лістинг програми

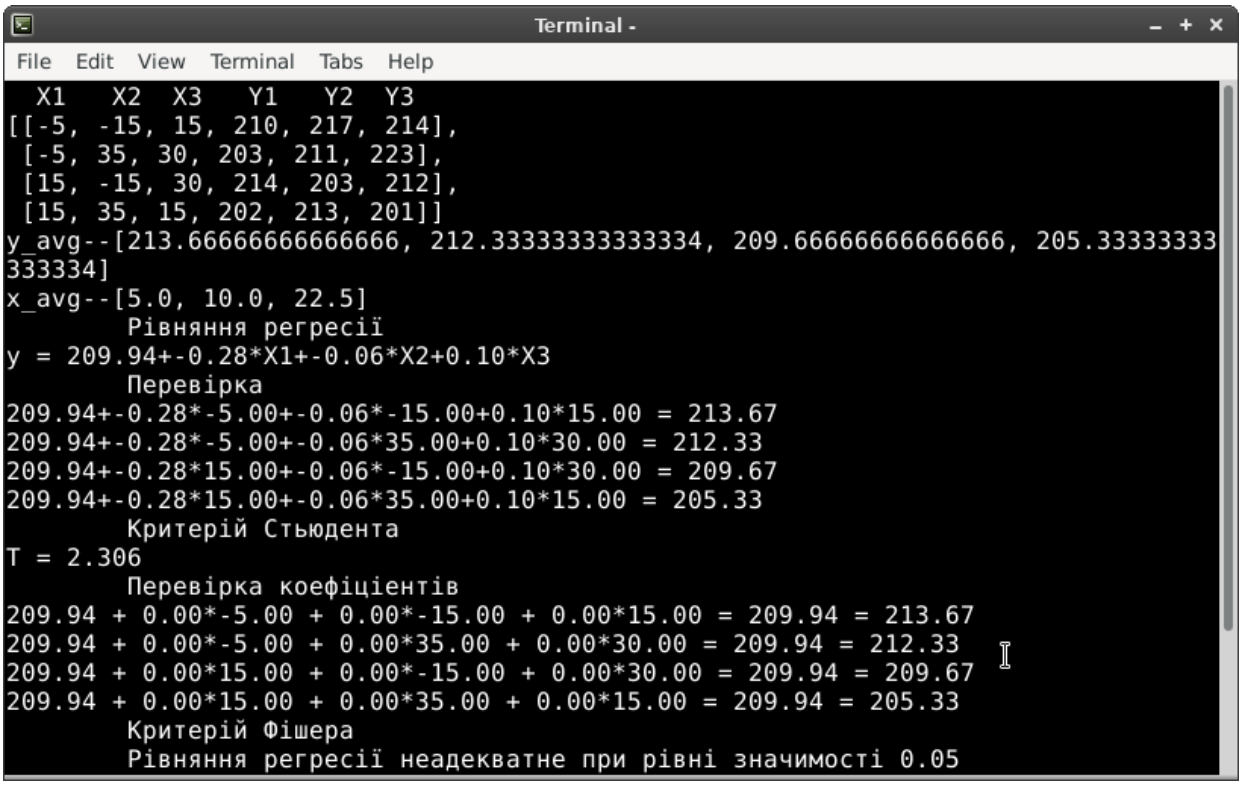
```
from math import *
from scipy.stats import f
import numpy as np
from _pydecimal import Decimal
import random as rnd
import pprint
x1_min = -5
x1_max = 15
x2_min = -15
x2_max = 35
x3_min = 15
x3_max = 30
y_max = 200 + int((x1_max + x2_max + x3_max) / 3)
y_min = 200 + int((x1_min + x2_min + x3_min) / 3)
p = 0.95
q = 1 - p
m = 3
N = 4
matrix = []
flag = True
while flag:
        x_matrix = [[x1_min, x2_min, x3_min],
                                [x1_min, x2_max, x3_max],
                                [x1_max, x2_min, x3_max],
                                [x1_max, x2_max, x3_min]]
        y_matrix = [[rnd.randrange(y_min, y_max) for j in range(m)] for i in range(len(x_matrix))]
        y_avg =    [sum(y_matrix[i]) / len(y_matrix[i]) for i in range(len(y_matrix))]
        tmp_avg = 0
        x_avg = []
        for j in range(len(x_matrix[0])):
                tmp_avg = 0
                for i in range(len(x_matrix)):
```

```
                tmp_avg += x_matrix[i][j] / len(x_matrix)
            x_avg.append(tmp_avg)
        my = sum(y_avg) / len(y_avg)
        a1, a2, a3, a11, a22, a33, a12, a13, a23 = 0, 0, 0, 0, 0, 0, 0, 0, 0
        for i in range(len(x_matrix)):
            a1 += x_matrix[i][0] * y_avg[i] / len(x_matrix)
            a2 += x_matrix[i][1] * y_avg[i] / len(x_matrix)
            a3 += x_matrix[i][2] * y_avg[i] / len(x_matrix)
            a11 += x_matrix[i][0]**2 / len(x_matrix)
            a22 += x_matrix[i][1]**2 / len(x_matrix)
            a33 += x_matrix[i][2]**2 / len(x_matrix)
            a12 += x_matrix[i][0] * x_matrix[i][1] / len(x_matrix)
            a13 += x_matrix[i][0] * x_matrix[i][2] / len(x_matrix)
            a23 += x_matrix[i][1] * x_matrix[i][2] / len(x_matrix)
        a21 = a12
        a31 = a13
        a32 = a23
        b0_mat = [[my, x_avg[0], x_avg[1], x_avg[2]],
                  [a1, a11, a12, a13],
                  [a2, a21, a22, a23],
                  [a3, a31, a32, a33]]
        b1_mat = [[1, my, x_avg[1], x_avg[2]],
                  [x_avg[0], a1, a12, a13],
                  [x_avg[1], a2, a22, a23],
                  [x_avg[2], a3, a32, a33]]
        b2_mat = [[1, x_avg[0], my, x_avg[2]],
                  [x_avg[0], a11, a1, a13],
                  [x_avg[1], a21, a2, a23],
                  [x_avg[2], a31, a3, a33]]
        b3_mat = [[1, x_avg[0], x_avg[1], my],
                  [x_avg[0], a11, a12, a1],
                  [x_avg[1], a21, a22, a2],
                  [x_avg[2], a31, a32, a3]]
        denom_mat = [[1, x_avg[0], x_avg[1], x_avg[2]],
                     [x_avg[0], a11, a12, a13],
                     [x_avg[1], a21, a22, a23],
                     [x_avg[2], a31, a32, a33]]
        b0 = np.linalg.det(b0_mat) / np.linalg.det(denom_mat)
        b1 = np.linalg.det(b1_mat) / np.linalg.det(denom_mat)
        b2 = np.linalg.det(b2_mat) / np.linalg.det(denom_mat)
        b3 = np.linalg.det(b3_mat) / np.linalg.det(denom_mat)
        b_list = [b0, b1, b2, b3]
        f1 = m - 1
        f2 = N
        q = 1 - p
        def cohren_value(size_of_selections, qty_of_selections, significance):
            size_of_selections += 1
            partResult1 = significance / (size_of_selections - 1)
            params = [partResult1, qty_of_selections, (size_of_selections - 1 - 1) * qty_of_selections]
            fisher = f.isf(*params)
            result = fisher / (fisher + (size_of_selections - 1 - 1))
            return Decimal(result).quantize(Decimal('.0001')).__float__()
        y_disp = []
        for i in range(len(x_matrix)):
            tmp_disp = 0
            for j in range(m):
                tmp_disp += ((y_matrix[i][j] - y_avg[i]) ** 2) / m
            y_disp.append(tmp_disp)
# ~ Критерій Кохрена
        Gp = max(y_disp) / sum(y_disp)
        Gt = cohren_value(f2, f1, q)
        if Gt > Gp:
            flag = False
```

```
        else:
                m += 1
# ~ Рівняння регресії
for i in range(4):
   matrix.append(x_matrix[i] + y_matrix[i])
print("  X1   X2  X3   Y1   Y2  Y3 ")
pprint.pprint(matrix)
print("y_avg--"+str(y_avg))
print("x_avg--"+str(x_avg))
print("    Рівняння регресії")
print("y = {:.2f}+{:.2f}*X1+{:.2f}*X2+{:.2f}*X3".format(b0, b1, b2, b3))
print("    Перевірка")
print("{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f} = ".format(b0, b1, x1_min, b2, x2_min, b3, x3_min)
    + str(round(b0 + b1 * x1_min + b2 * x2_min + b3 * x3_min,2)))
print("{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f} = ".format(b0, b1, x1_min, b2, x2_max, b3, x3_max)
    + str(round(b0 + b1 * x1_min + b2 * x2_max + b3 * x3_max,2)))
print("{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f} = ".format(b0, b1, x1_max, b2, x2_min, b3, x3_max)
    + str(round(b0 + b1 * x1_max + b2 * x2_min + b3 * x3_max,2)))
print("{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f}+{:.2f}*{:.2f} = ".format(b0, b1, x1_max, b2, x2_max, b3, x3_min)
    + str(round(b0 + b1 * x1_max + b2 * x2_max + b3 * x3_min,2)))
# ~ Критерій Стьюдента
print("    Критерій Стьюдента")
f3 = f1 * f2
S2b = sum(y_disp) / (N * N * m)
Sb = sqrt(S2b)
beta0 = ( y_avg[0] + y_avg[1] + y_avg[2] + y_avg[3]) / N
beta1 = (-y_avg[0] - y_avg[1] + y_avg[2] + y_avg[3]) / N
beta2 = (-y_avg[0] + y_avg[1] - y_avg[2] + y_avg[3]) / N
beta3 = (-y_avg[0] + y_avg[1] + y_avg[2] - y_avg[3]) / N
t0 = abs(beta0) / Sb
t1 = abs(beta1) / Sb
t2 = abs(beta2) / Sb
t3 = abs(beta3) / Sb
T_list = [t0, t1, t2, t3]
student_table = [12.71, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365,
                                        2.306, 2.262, 2.228, 2.201, 2.179, 2.160, 2.145,
                                        2.131, 2.12,  2.11,  2.101, 2.093, 2.086, 2.08,
                                        2.074, 2.069, 2.064, 2.06, 2.056, 2.052,  2.048,
                                        2.045, 2.042];
try:
        T = student_table[f3-1]
except:
        T = 1.960
print("T = "+str(T))
for i in range(len(T_list)):
        if T_list[i] < T :
                T_list[i] = 0
                b_list[i] = 0
print("    Перевірка коефіцієнтів")
y_1 = b_list[0] + b_list[1] * x1_min + b_list[2] * x2_min + b_list[3] * x3_min
y_2 = b_list[0] + b_list[1] * x1_min + b_list[2] * x2_max + b_list[3] * x3_max
y_3 = b_list[0] + b_list[1] * x1_max + b_list[2] * x2_min + b_list[3] * x3_max
y_4 = b_list[0] + b_list[1] * x1_max + b_list[2] * x2_max + b_list[3] * x3_min
print("{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} = "\
.format(b_list[0],b_list[1],x1_min,b_list[2],x2_min,b_list[3],x3_min)\
 + str(round(y_1,2))+" = "+str(round(y_avg[0],2)))
print("{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} = "\
.format(b_list[0],b_list[1],x1_min,b_list[2],x2_max,b_list[3],x3_max)\
 + str(round(y_2,2))+" = "+str(round(y_avg[1],2)))
print("{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} = "\
.format(b_list[0],b_list[1],x1_max,b_list[2],x2_min,b_list[3],x3_max)\
 + str(round(y_3,2))+" = "+str(round(y_avg[2],2)))
print("{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} + {:.2f}*{:.2f} = "\
```

```
.format(b_list[0],b_list[1],x1_max,b_list[2],x2_max,b_list[3],x3_min)\
 + str(round(y_4,2))+" = "+str(round(y_avg[3],2)))
# ~ Критерій Фішера
print("    Критерій Фішера")
fisher_table = [[164.4, 199.5, 215.7, 224.6, 230.2, 234],
                                        [18.5, 19.2, 19.2, 19.3, 19.3, 19.3],
                                        [10.1, 9.6, 9.3, 9.1, 9, 8.9],
                                        [7.7, 6.9, 6.6, 6.4, 6.3, 6.2],
                                        [6.6, 5.8, 5.4, 5.2, 5.1, 5],
                                        [6, 5.1, 4.8, 4.5, 4.4, 4.3],
                                        [5.5, 4.7, 4.4, 4.1, 4, 3.9],
                                        [5.3, 4.5, 4.1, 3.8, 3.7, 3.6],
                                        [5.1, 4.3, 3.9, 3.6, 3.5, 3.4],
                                        [5, 4.1, 3.7, 3.5, 3.3, 3.2],
                                        [4.8, 4, 3.6, 3.4, 3.2, 3.1],
                                        [4.8, 3.9, 3.5, 3.3, 3.1, 3],
                                        [4.7, 3.8, 3.4, 3.2, 3, 2.9],
                                        [4.6, 3.7, 3.3, 3.1, 3, 2.9],
                                        [4.5, 3.7, 3.3, 3.1, 2.9, 2.8],
                                        [4.5, 3.6, 3.2, 3, 2.9, 2.7],
                                        [4.5, 3.6, 3.2, 3, 2.8, 2.7],
                                        [4.4, 3.6, 3.2, 2.9, 2.8, 2.7],
                                        [4.4, 3.5, 3.1, 2.9, 2.7, 2.6],
                                        [4.4, 3.5, 3.1, 2.9, 2.7, 2.6]]
b_list = list(filter(lambda i : (i != 0), b_list))
d = len(b_list)
f4 = N - d # [f3][f4]
S2ad = m * ((y_1-y_avg[0])**2 + (y_2-y_avg[1])**2 + (y_3-y_avg[2])**2 + (y_4-y_avg[3])**2)/f4
Fp = S2ad / S2b
Ft = fisher_table[f3-1][f4-1]
if Fp > Ft: print(" Рівняння регресії неадекватне при рівні значимості {:.2f}".format(q))
else: print("         Рівняння регресії адекватне при рівні значимості {:.2f}".format(q))
```

Результат виконання



Висновки

Підчас виконання лабораторної роботи було реалізовано завдання . Отримані результати збігаються , отже, експеримент було поставлено правильно.