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 №3

«Проведення двофакторного експерименту з використанням лінійного рівняння регресії»

Виконав:
студент ІІ курсу ФІОТ
групи ІВ-93
Ровишин Андрій

Перевірив:
Регіда П.Г.

Київ – 2021

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

Завдання на лабораторну роботу

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.

319	-20	30	20	60	-20	-5

Код програми

```
from random import randint
from numpy.linalg import det
from functools import reduce


def naturalize(matrix_of_plan, min_max_arr):
    result = []
    for i in matrix_of_plan:
        result.append(min_max_arr[1]) if i == 1 else result.append(min_max_arr[0])
    return result


M = 4
x1 = [-20, 30]
x2 = [20, 60]
x3 = [-20, -25]
print(f'x1_min = {x1[0]}, x1_max = {x1[1]}')
print(f'x2_min = {x2[0]}, x2_max = {x2[1]}')
print(f'x3_min = {x3[0]}, x3_max = {x3[1]}')

# ----------------------------------Матриця планування експерименту з +1,-1----------
---------------------------------
x0_mx1 = [1, 1, 1, 1]
x1_mx1 = [-1, -1, 1, 1]
x2_mx1 = [-1, 1, -1, 1]
x3_mx1 = [-1 * (x1_mx1[i] * x2_mx1[i]) for i in range(len(x1_p1))]
print('x0:', x0_mx1)
```

```
print('x1:', x1_mx1)
print('x2:', x2_mx1)
print('x3:', x3_mx1)

# ------------------------------Матриця планування з натуралізованими значеннями
факторів-----------------------------
x1_mx2 = naturalize(x1_mx1, x1)
x2_mx2 = naturalize(x2_mx1, x2)
x3_mx2 = naturalize(x3_mx1, x3)
print('\nx1:', x1_mx2)
print('x2:', x2_mx2)
print('x3:', x3_mx2)

x_avg_max = (max(x1_mx2) + max(x2_mx2) + max(x3_mx2)) / 3
x_avg_min = (min(x1_mx2) + min(x2_mx2) + min(x3_mx2)) / 3
print(f'\nx_avg_max = {x_avg_max}')
print(f'x_avg_min = {x_avg_min}')

# ------------------------------------------------Діапазон y-------------------------
---------------------------------
y_Max = int(200 + x_avg_max)
y_Min = int(200 + x_avg_min)
print(f'\ny_max = {y_Max}')
print(f'y_min = {y_Min}')

y1 = [randint(y_Min, y_Max) for _ in range(4)]
y2 = [randint(y_Min, y_Max) for _ in range(4)]
y3 = [randint(y_Min, y_Max) for _ in range(4)]
print('y1:', y1)
print('y2:', y2)
print('y3:', y3)

y_avg = [(y1[i] + y2[i] + y3[i]) / 3 for i in range(4)]
print('y average:', y_avg)

# ----------------------------------------Математичне очікування--------------------
---------------------------------
mx1 = reduce(lambda a, b: a + b, x1_mx2) / 4
mx2 = reduce(lambda a, b: a + b, x2_mx2) / 4
mx3 = reduce(lambda a, b: a + b, x3_mx2) / 4
my = reduce(lambda a, b: a + b, y_avg) / 4
print(f'\nmx1 = {mx1}')
print(f'mx2 = {mx2}')
print(f'mx3 = {mx3}')
print(f'my = {my}')

a1 = sum([x1_mx2[i] * y_avg[i] for i in range(4)]) / 4
a2 = sum([x2_mx2[i] * y_avg[i] for i in range(4)]) / 4
a3 = sum([x3_mx2[i] * y_avg[i] for i in range(4)]) / 4
print(f'\na1 = {a1}')
print(f'a2 = {a2}')
print(f'a3 = {a3}')

a11 = sum([i * i for i in x1_mx2]) / 4
a22 = sum([i * i for i in x2_mx2]) / 4
a33 = sum([i * i for i in x3_mx2]) / 4
print(f'\na11 = {a11}')
print(f'a22 = {a22}')
print(f'a33 = {a33}')

a12 = sum([x1_mx2[i] * x2_mx2[i] for i in range(4)]) / 4
a13 = sum([x1_mx2[i] * x3_mx2[i] for i in range(4)]) / 4
a23 = sum([x2_mx2[i] * x3_mx2[i] for i in range(4)]) / 4
a21 = a12
```

```
a31 = a13
a32 = a23
print(f'\na12 = {a12}')
print(f'a13 = {a13}')
print(f'a23 = {a23}')
print(f'a21 = {a21}')
print(f'a31 = {a31}')
print(f'a32 = {a32}')

b0 = det([[my, mx1, mx2, mx3],
          [a1, a11, a12, a13],
          [a2, a21, a22, a23],
          [a3, a31, a32, a33]]) / det([[1, mx1, mx2, mx3],
                                       [mx1, a11, a12, a13],
                                       [mx2, a21, a22, a23],
                                       [mx3, a31, a32, a33]])
b1 = det([[1, my, mx2, mx3],
          [mx1, a1, a12, a13],
          [mx2, a2, a22, a23],
          [mx3, a3, a32, a33]]) / det([[1, mx1, mx2, mx3],
                                       [mx1, a11, a12, a13],
                                       [mx2, a21, a22, a23],
                                       [mx3, a31, a32, a33]])
b2 = det([[1, mx1, my, mx3],
          [mx1, a11, a1, a13],
          [mx2, a21, a2, a23],
          [mx3, a31, a3, a33]]) / det([[1, mx1, mx2, mx3],
                                       [mx1, a11, a12, a13],
                                       [mx2, a21, a22, a23],
                                       [mx3, a31, a32, a33]])
b3 = det([[1, mx1, mx2, my],
          [mx1, a11, a12, a1],
          [mx2, a21, a22, a2],
          [mx3, a31, a32, a3]]) / det([[1, mx1, mx2, mx3],
                                       [mx1, a11, a12, a13],
                                       [mx2, a21, a22, a23],
                                       [mx3, a31, a32, a33]])

print(f'y = {b0} + {b1}*x1 + {b2}*x2 + {b3}*x3')

for i in range(4):
    y = b0 + b1 * x1_mx2[i] + b2 * x2_mx2[i] + b3 * x3_mx2[i]
    print('y =', y)

# ----------------------------------Перевірка однорідності дисперсії за критерієм
Кохрена----------------------------
dspr = [((y1[i] - y_avg[i]) ** 2 + (y2[i] - y_avg[i]) ** 2 + (y3[i] - y_avg[i]) ** 2) /
3 for i in
              range(4)]
print('dispersion:', dspr)

gp = max(dspr) / sum(dspr)
print('Gp =', gp)

#  Gт = 0.7679
if gp < 0.7679:
    print('Дисперсія однорідна')
    # ----------------------Оцінка значимості коефіцієнтів регресії згідно критерію
Стьюдента-------------------------
    s2b = sum(dspr) / 4
    s2bs_avg = s2b / 4 * M
    sb = s2bs_avg ** (1 / 2)

    beta0 = sum([y_avg[i] * x0_mx1[i] for i in range(4)]) / 4
```

```
    beta1 = sum([y_avg[i] * x1_mx1[i] for i in range(4)]) / 4
    beta2 = sum([y_avg[i] * x2_mx1[i] for i in range(4)]) / 4
    beta3 = sum([y_avg[i] * x3_mx1[i] for i in range(4)]) / 4

    beta_arr = [beta0, beta1, beta2, beta3]
    print('beta:', beta_arr)
    t_arr = [abs(beta_arr[i]) / sb for i in range(4)]
    print('tetta:', t_arr)

    # ------------------------------------f3 = f1*f2-------------------------------
--------------------------------
    indexes = []
    for i, v in enumerate(t_arr):
        if t_arr[i] > 2.306:
            indexes.append(i)
        else:
            print(f'Коефіцієнт b{i} = {v} приймаємо не значним')

    b_list = [b0, b1, b2, b3]
    print(f'y = b{indexes[0]}')

    b_res = [b_list[indexes[0]] for _ in range(4)]
    for i in b_res:
        print(f'y = {i}')



    d = 1
    s2_ad = M * sum([(y_avg[i] - b_res[i]) ** 2 for i in range(4)]) / 4 - d
    fp = s2_ad / s2b
    print(f'Fp = {fp}')

    # Fт = 4.5
    if fp > 4.5:
        print('Рівняння регресії неадекватно оригіналу при рівні значимості 0.05')
    else:
        print('Рівняння регресії адекватно оригіналу при рівні значимості 0.05')
else:
    print('Дисперсія неоднорідна')
```

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