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

Методи оптимізації та планування експерименту

Лабораторна робота №2

«ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконав:
Студент групи ІО-92
Педенко Данило Денисович
Перевірив:
ас. Регіда П. Г.

Київ
2021 р.

Лабораторна робота № 2

Тема: ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ.

Мета: провести двофакторний експеримент, перевірити однорідність дисперсії за критерієм Романовського, отримати коефіцієнти рівняння регресії, провести натуралізацію рівняння регресії.

Завдання:

1. Записати лінійне рівняння регресії.
2. Обрати тип двофакторного експерименту і скласти матрицю планування для нього з використанням додаткового нульового фактору (xo=1).
3. Провести експеримент в усіх точках повного факторного простору (знайти значення функції відгуку y). Значення функції відгуку задати випадковим чином у відповідності до варіанту у діапазоні ymin ÷ ymax.
4. Перевірити однорідності дисперсії за критерієм Романовського
5. Знайти коефіцієнти нормованих рівнянь регресії і виконати перевірку (підставити значення нормованих факторів і коефіцієнтів у рівняння).
6. Провести натуралізацію рівняння регресії й виконати перевірку натуралізованого рівняння.
7. Написати комп'ютерну програму, яка все це виконує.

Завдання відповідно за номером варіанту:

215	10	50	-20	60

Роздруківка тексту програми:

```
import random as ran
import math as ma
import numpy
from prettytable import PrettyTable


def inf(count, mean):
    array_x1 = [10, 50]
    m = 5 + count
    array_x2 = [-20, 60]
    array_y = [(30 - 15) * 10, (20 - 15) * 10]
    teta0 = ma.sqrt(2 * (2 * m - 2) / (m * (m - 4)))
    matr_x = []
    matr_x2 = []
    matr_y = [[ran.randint(array_y[1], array_y[0]) for j in range(m)] for i in
range(3)]
    avey = []
    sigma = []
    array_fuv = []
    array_a = []
    array_aij = []
    for i in range(3):
        if i == 0:
            matr_x.append([-1, -1])
            matr_x2.append([min(array_x1), min(array_x2)])
        elif i == 1:
            matr_x.append([-1, 1])
            matr_x2.append([min(array_x1), max(array_x2)])
        elif i == 2:
```

```
            matr_x.append([1, -1])
            matr_x2.append([max(array_x1), min(array_x2)])
    for i in range(len(matr_y)):
        sumer = 0
        temp = sum(matr_y[i]) / len(matr_y[i])
        avey.append(temp)
        for j in range(len(matr_y[i])):
            sumer += (matr_y[i][j] - temp) ** 2
        sigma.append(sumer / len(matr_y[i]))
    array_fuv.append(sigma[0] / sigma[1])
    array_fuv.append(sigma[2] / sigma[0])
    array_fuv.append(sigma[2] / sigma[1])
    array_tetas = [(m - 2) / m * array_fuv[i] for i in range(len(array_fuv))]
    array_ruv = [ma.fabs(i - 1) / teta0 for i in array_tetas]
    for i in range(len(array_ruv)):
        if array_ruv[i] < 2:
            mean = True
            print("R{0}uv > 2".format(i))
        else:
            mean = False
            print("R{0}uv < 2".format(i))
    if mean:
        trans = numpy.array(matr_x).transpose()
        array_mx = [sum(trans[i]) / len(trans[i]) for i in range(2)]
        my = sum(avey) / len(avey)
        for i in range(2):
            temp = 0
            if i == 1:
                for j in matr_x:
                    temp += numpy.array(j).prod()
                array_a.append(temp / 3)
                temp = 0
            for j in range(len(trans[i])):
                temp += (trans[i][j] ** 2)
            array_a.append(temp / 3)
        for i in range(2):
            temp = 0
            for j in range(len(trans[i])):
                temp += trans[i][j] * avey[j]
            array_aij.append(temp / 3)
        first = numpy.array(
            [[1, array_mx[0], array_mx[1]], [array_mx[0], array_a[0], array_a[1]],
             [array_mx[1], array_a[1], array_a[2]]])
        second = numpy.array([my, array_aij[0], array_aij[1]])
        res = numpy.linalg.solve(first, second)
        array_delx = [(max(array_x1) - min(array_x1)) / 2, (max(array_x2) -
min(array_x2)) / 2]
        array_zerx = [sum(array_x1) / 2, sum(array_x2) / 2]
        a0 = res[0] - res[1] * (array_zerx[0] / array_delx[0]) - res[2] *
(array_zerx[1] / array_delx[1])
        a1 = res[1] / array_delx[0]
        a2 = res[2] / array_delx[1]
        ta = PrettyTable()
        ta.field_names = ["X1", "X2", "Y1", "Y2", "Y3", "Y4", "Y5"]
        ta.add_rows(
            [
                [matr_x[0][0], matr_x[0][1], matr_y[0][0], matr_y[0][1],
matr_y[0][2], matr_y[0][3], matr_y[0][4]],
                [matr_x[1][0], matr_x[1][1], matr_y[1][0], matr_y[1][1],
matr_y[1][2], matr_y[1][3], matr_y[1][4]],
                [matr_x[2][0], matr_x[2][1], matr_y[2][0], matr_y[2][1],
matr_y[2][2], matr_y[2][3], matr_y[2][4]],
```

```
            ]
        )
        print("m = ", m)
        print("Матриця планування для m = 5")
        print(ta)
        print("Нормованні значення X1 та X2:\n", matr_x)
        print("Значення функції відгуку при m = {0}:\n".format(m),
numpy.array(matr_y))
        print("Середнє значення функції відгуку:\n", avey)
        print("Значення дисперсій:\n", sigma)
        print("Значення Fuv:\n", array_fuv)
        print("Значення θuv:\n", array_tetas)
        print("Значення Ruv:\n", array_ruv)
        print("Матиматичне очікування X1 та X2:\n", array_mx)
        print("Значення a:\n", array_a)
        print("Значення aij:\n", array_aij, "\n")
        print("Нормоване рівняння регресії")
        print("y = {0} + {1}*x1 + {2}*x2\n".format(res[0], res[1], res[2]))
        print("Зробимо перевірку:")
        for i in range(len(matr_x)):
            check = res[0] + res[1] * matr_x[i][0] + res[2] * matr_x[i][1]
            print("y{0} = {1}".format(i, check))
        print("\n")
        print("Натуралізоване рівняння регресії")
        print("y = {0} + {1}*x1 + {2}*x2\n".format(a0, a1, a2))
        print("Зробимо перевірку:")
        for i in range(len(matr_x2)):
            check = a0 + a1 * matr_x2[i][0] + a2 * matr_x2[i][1]
            print("y{0} = {1}".format(i, check))
        return mean
    else:
        return mean


a = 0
while True:
    b = False
    if inf(a, b):
        n = input("Введіть \"stop\" щоб зупинити програму: ")
        if n == "stop":
            break
    else:
        n = input("Введіть \"stop\" щоб зупинити програму:")
        if n == "stop":
            break
        print("Збільшуємо m на 1")
        a += 1
```

Роздруківка результатів роботи програми:

```
D:\Anaconda\envs\Lab2MOPE\python.exe "D:/Педенко ІО-92/MOPE/Lab2MOPE/lab2.py"
R0uv > 2
R1uv > 2
R2uv > 2
m =  5
Матриця планування для m = 5
+----+----+-----+-----+----+-----+-----+
| X1 | X2 |  Y1 |  Y2 | Y3 |  Y4 |  Y5 |
+----+----+-----+-----+----+-----+-----+
| -1 | -1 | 116 | 110 | 82 | 134 | 125 |
| -1 | 1  | 111 | 131 | 90 |  85 |  67 |
| 1  | -1 | 108 | 109 | 98 | 145 |  59 |
+----+----+-----+-----+----+-----+-----+
Нормованні значення X1 та X2:
 [[-1, -1], [-1, 1], [1, -1]]
Значення функції відгуку при m = 5:
 [[116 110  82 134 125]
 [111 131  90  85  67]
 [108 109  98 145  59]]
Середнє значення функції відгуку:
 [113.4, 96.8, 103.8]
Значення дисперсій:
 [312.64000000000004, 488.96000000000004, 756.5600000000001]
Значення Fuv:
 [0.6393979057591623, 2.419907881269191, 1.5472840314136125]
Значення θuv:
 [0.38363874345549737, 1.4519447287615146, 0.9283704188481674]
Значення Ruv:
 [0.3445564170826738, 0.25264478389586276, 0.040042153163833846]
Матиматичне очікування X1 та X2:
 [-0.3333333333333333, -0.3333333333333333]
Значення a:
 [1.0, -0.3333333333333333, 1.0]
Значення aij:
 [-35.46666666666666, -40.13333333333333]

Нормоване рівняння регресії
y = 100.30000000000001 + -4.799999999999998*x1 + -8.3*x2

Зробимо перевірку:
y0 = 113.4
y1 = 96.80000000000001
y2 = 103.80000000000001
Натуралізоване рівняння регресії
y = 111.65000000000002 + -0.2399999999999999*x1 + -0.20750000000000002*x2

Зробимо перевірку:
y0 = 113.40000000000002
y1 = 96.80000000000001
y2 = 103.80000000000003
Введіть "stop" щоб зупинити програму: stop

Process finished with exit code 0
```

Висновок:

В ході лабораторної роботи, було проведено двофакторний експеримент, перевірено однорідність дисперсії за критерієм Романовського, отримано коефіцієнти рівняння регресії, проведено натуралізацію рівняння регресії. Закріплено отримані знання практичним їх використанням при написанні програми, що реалізує завдання на лабораторну роботу.

Контрольні запитання

1) **Що таке регресійні поліноми і де вони застосовуються?**

В теорії планування експерименту найважливішою частиною є оцінка результатів вимірів. При цьому використовують апроксимуючі поліноми, за допомогою яких ми можемо описати нашу функцію. В ТПЕ ці поліноми отримали спеціальну назву - регресійні поліноми

2) **Визначення однорідності дисперсії.**

Для цього необхідно спочатку знайти середньоарифметичне значення дослідів $\bar{y}_j$ ($j=\overline{1,m}$) (математичне сподівання m_{yj}) в кожній точці факторного простору: $\bar{y}_j=(1/m)\sum_1^m y_{js}$($i=\overline{1,N}$).

Оскільки теоретичні значення дисперсії σ^2_j ($j=\overline{1,N}$) невідомі, то перевірка однорідності дисперсії виконується на основі аналізу статистичних оцінок дисперсії S^2_j ($i=\overline{1,N}$) для усіх точок факторного простору.

Статистичні оцінки дисперсії S^2_j ($j=\overline{1,N}$) для кожної точки факторного простору розрахо-вують за формулою: $S^2_j=\{1/(m-1)\}\{\sum_1^m(y_{js}-\bar{y}_j)^2\}$ ($j=\overline{1,N}$).

Отже, перевірка однорідності дисперсії – це перевірка гіпотези стосовно належності **N** значень статистичних оцінок дисперсії S^2_j ($i=\overline{1,N}$) одній генеральній сукупності.

3) **Що називається повним факторним експериментом?**

ПФЕ – повний факторний експеримент,- це коли використовуються усі можливі комбінації рівнів факторів; при ПФЕ кількість комбінацій $N_п=r^k$.