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 №3

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

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

Київ
2021 р.

Лабораторна робота № 3

Тема: ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ.

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

Завдання:

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.

Завдання відповідно за номером варіанту:

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
215	10	50	-20	60	10	15

Роздруківка тексту програми:

```
import numpy as np
import random as ra
import math as ma
from prettytable import PrettyTable


def cochrane(eq, n, list_y, list_av_y):
    i = 0
    f1 = eq - 1
    f2 = n
    list_g = [9065, 7679, 6841, 6287, 5892, 5598, 5365, 5175, 5017, 4884]
    list_sig = []
    for i in range(len(list_y)):
        tem = 0
        for j in range(len(list_y[i])):
            tem += pow(list_y[i][j] - list_av_y[i], 2)
        list_sig.append(tem / len(list_y[i]))
    gp = max(list_sig) / sum(list_sig)
```

```
    print("F1 = ", f1)
    print("F2 = ", f2)
    print("q = 0.05")
    print("Значення дисперсій по рядках")
    print(list_sig)
    print("\nGp = ", gp)
    for i in range(len(list_g)):
        if i == f1 - 1:
            if gp < list_g[i] / 10000:
                print("\nGp = {0} < Gt = {1}".format(gp, list_g[i] / 10000))
                print("Дисперсія однорідна\n")
                print("Оцінимо значимість коефіцієнтів регресії згідно критерію
Стьюдента")
                student(eq, n, list_sig, list_av_y, array_xp.transpose(), res,
array_xn)
            else:
                print("Дисперсія не однорідна")


def student(eq, n, list_sig, list_av_y, list_x, koef, list_xn):
    list_t_prover = [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, 2.120, 2.110, 2.101, 2.093, 2.086, 2.080, 2.074, 2.069,
2.064, 2.060, 2.056, 2.052, 2.048,
                     2.045, 2.042]
    sv = sum(list_sig) / n
    s_sq_beta = sv / (n * eq)
    s_beta = ma.sqrt(s_sq_beta)
    list_beta = []
    new_koef = []
    for i in range(len(list_x)):
        pol = 0
        for j in range(len(list_x[i])):
            pol += list_x[i][j] * list_av_y[j]
        list_beta.append(pol / n)
    print("m = ", m)
    print("N = ", n)
    print("Отримані значення βi")
    print(list_beta)
    list_t = [abs(list_beta[i]) / s_beta for i in range(len(list_beta))]
    print("Отримані значення ti")
    print(list_t)
    print("\nf3 = ", (eq - 1) * n)
    print("q = 0.05")
    for i in range(len(list_t_prover)):
        if i == (eq - 1) * n - 1:
            for j in range(len(list_t)):
                if list_t[j] < list_t_prover[i]:
                    print("\nt{0} = {1} < ttабл = {2}".format(j, list_t[j],
list_t_prover[i]))
                    print("b{0} - виключається з рівняння".format(j))
                else:
                    print("\nt{0} = {1} > ttабл = {2}".format(j, list_t[j],
list_t_prover[i]))
                    new_koef.append([j, koef[j]])
    print("\nПерепишемо рівняння враховуючи вилучених коефіцієнтів")
    rivn = "y = "
    for i in range(len(new_koef)):
        if i == 0:
            rivn += str(new_koef[i][1])
        else:
            rivn += " + " + str(new_koef[i][1]) + " * x{0}".format(new_koef[i][0])
```

```
    print(rivn)
    list_res_y = []
    print("\nПідставимо необхідні значення X")
    for i in range(len(list_xn)):
        pol = 0
        for j in range(len(new_koef)):
            if new_koef[j][0] == 0:
                pol += new_koef[j][1]
            else:
                pol += list_xn[i][new_koef[j - 1][0]] * new_koef[j][1]
        list_res_y.append(pol)
        print("y{0} = {1}".format(i, pol))
    print("\nКритерій Фішера")
    fisher(len(new_koef), n, eq, list_res_y, list_av_y, sv)


def fisher(d, n, eq, list_res_y, list_av_y, sv):
    fish_table = [[8, [5.3, 4.5, 4.1, 3.8, 3.7, 3.6]],
                  [12, [4.8, 3.9, 3.5, 3.3, 3.1, 3.0]],
                  [16, [4.5, 3.6, 3.2, 3.0, 2.9, 2.7]],
                  [20, [4.4, 3.5, 3.1, 2.9, 2.7, 2.6]],
                  [24, [4.3, 3.4, 3.0, 2.8, 2.6, 2.5]],
                  [28, [4.2, 3.3, 3.0, 2.7, 2.6, 2.4]]]
    f4 = n - d
    f3 = (m - 1) * n
    sad = 0
    print("d = ", d)
    print("f3 = ", f3)
    print("f4 = ", f4)
    print("q = 0.05")
    for i in range(n):
        sad += pow(list_res_y[i] - list_av_y[i], 2)
    sad = sad * eq / (n - d)
    fp = sad / sv
    for i in fish_table:
        for j in range(len(i)):
            if i[j] == f3:
                if fp > i[1][f4 - 1]:
                    print("\nFp = {0} > Ft = {1}".format(fp, i[1][f4 - 1]))
                    print("Рівняння регресії неадекватно оригіналу")
                else:
                    print("\nFp = {0} < Ft = {1}".format(fp, i[1][f4 - 1]))
                    print("Рівняння регресії адекватно оригіналу")


m = 3
array_xd = [[10, 50], [-20, 60], [10, 15]]
array_yd = [round(200 + (max(array_xd[0]) + max(array_xd[1]) + max(array_xd[2])) /
len(array_xd)),
            200 + (min(array_xd[0]) + min(array_xd[1]) + min(array_xd[2])) /
len(array_xd)]
array_xp = np.array([[1, -1, -1, -1], [1, -1, 1, 1], [1, 1, -1, 1], [1, 1, 1, -1]])
array_y = np.array([[ra.randint(array_yd[1], array_yd[0]) for j in range(m)] for i in
range(4)])
array_aver_y = [sum(array_y[i]) / len(array_y[i]) for i in range(len(array_y))]
array_xn = []
array_a = []
array_aii = []
array_aij = []
for i in range(len(array_xp)):
    array_xn.append([])
    for j in range(len(array_xd)):
```

```
        if array_xp[i][j + 1] == -1:
            array_xn[i].append(min(array_xd[j]))
        else:
            array_xn[i].append(max(array_xd[j]))
trans = np.array(array_xn).transpose()
array_mx = np.array([sum(trans[i]) / len(trans[i]) for i in range(len(trans))])
my = sum(array_aver_y) / len(array_aver_y)
for i in range(len(trans)):
    summa = 0
    kvad = 0
    poly = 0
    for j in range(len(trans[i])):
        summa += trans[i][j] * array_aver_y[j]
        kvad += pow(trans[i][j], 2)
        if i == 0:
            poly += trans[i][j] * trans[i + 1][j]
        elif i == 1:
            poly += trans[0][j] * trans[2][j]
        else:
            poly += trans[1][j] * trans[2][j]
    array_a.append(summa / 4)
    array_aii.append(kvad / len(trans[i]))
    array_aij.append(poly / len(trans[i]))
ta = PrettyTable()
ta.field_names = ["X0", "X1", "X2", "X3", "Y1", "Y2", "Y3"]
ta.add_rows(
    [
        [array_xp[0][0], array_xp[0][1], array_xp[0][2], array_xp[0][3],
array_y[0][0], array_y[0][1], array_y[0][2]],
        [array_xp[1][0], array_xp[1][1], array_xp[1][2], array_xp[1][3],
array_y[1][0], array_y[1][1], array_y[1][2]],
        [array_xp[2][0], array_xp[2][1], array_xp[2][2], array_xp[2][3],
array_y[2][0], array_y[2][1], array_y[2][2]],
        [array_xp[3][0], array_xp[3][1], array_xp[3][2], array_xp[3][3],
array_y[3][0], array_y[3][1], array_y[3][2]]
    ]
)
ta1 = PrettyTable()
ta1.field_names = ["X1", "X2", "X3", "Y1", "Y2", "Y3"]
ta1.add_rows(
    [
        [array_xn[0][0], array_xn[0][1], array_xn[0][2], array_y[0][0],
array_y[0][1], array_y[0][2]],
        [array_xn[1][0], array_xn[1][1], array_xn[1][2], array_y[1][0],
array_y[1][1], array_y[1][2]],
        [array_xn[2][0], array_xn[2][1], array_xn[2][2], array_y[2][0],
array_y[2][1], array_y[2][2]],
        [array_xn[3][0], array_xn[3][1], array_xn[3][2], array_y[3][0],
array_y[3][1], array_y[3][2]]
    ]
)
print("Матриця планування експерименту")
print(ta)
print("Матриця планування експерименту для натуралізованих значень при m = 3")
print(ta1)
print("Середні значення функції відгуку за рядками")
print(array_aver_y)
res = np.linalg.solve(
    [[1, array_mx[0], array_mx[1], array_mx[2]], [array_mx[0], array_aii[0],
array_aij[0], array_aij[1]],
     [array_mx[1], array_aij[0], array_aii[1], array_aij[2]], [array_mx[2],
array_aij[1], array_aij[2], array_aii[2]]],
```

```
    [my, array_a[0], array_a[1], array_a[2]])
print("Значення коефіцієнтів")
print(res)
print("\nРівняння регресії")
print("{0} + ({1}) * x1 + ({2}) * x2 + ({3}) * x3\n".format(round(res[0], 3),
round(res[1], 3), round(res[2], 3),
                                                          round(res[3], 3)))
print("Зробимо перевірку підстановкою")
for i in range(len(array_xn)):
    temp = res[0] + res[1] * array_xn[i][0] + res[2] * array_xn[i][1] + res[3] *
array_xn[i][2]
    print("y{0} = {1}".format(i, temp))
print("\nПеревірка однорідності дисперсії за критерієм Кохрена")
cochrane(m, 4, array_y, array_aver_y)
```

Роздруківка результатів роботи програми:

```
D:\Anaconda\envs\Lab3MOPE\python.exe "D:/Педенко ІО-92/MOPE/Lab3MOPE/lab3.py"
Матриця планування експерименту
+----+----+----+----+-----+-----+-----+
| X0 | X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+----+-----+-----+-----+
| 1  | -1 | -1 | -1 | 236 | 205 | 232 |
| 1  | -1 | 1  | 1  | 200 | 222 | 235 |
| 1  | 1  | -1 | 1  | 242 | 236 | 213 |
| 1  | 1  | 1  | -1 | 210 | 233 | 209 |
+----+----+----+----+-----+-----+-----+
Матриця планування експерименту для натуралізованих значень при m = 3
+----+-----+----+-----+-----+-----+
| X1 |  X2 | X3 |  Y1 |  Y2 |  Y3 |
+----+-----+----+-----+-----+-----+
| 10 | -20 | 10 | 236 | 205 | 232 |
| 10 |  60 | 15 | 200 | 222 | 235 |
| 50 | -20 | 15 | 242 | 236 | 213 |
| 50 |  60 | 10 | 210 | 233 | 209 |
+----+-----+----+-----+-----+-----+
Середні значення функції відгуку за рядками
[224.33333333333334, 219.0, 230.33333333333334, 217.33333333333334]
Значення коефіцієнтів
[ 2.13833333e+02  5.41666667e-02 -1.14583333e-01  7.66666667e-01]

Рівняння регресії
213.833 + (0.054) * x1 + (-0.115) * x2 + (0.767) * x3

Зробимо перевірку підстановкою
y0 = 224.33333333333348
y1 = 218.99999999999997
y2 = 230.3333333333334
y3 = 217.33333333333343

Перевірка однорідності дисперсії за критерієм Кохрена
F1 =  2
F2 =  4
q = 0.05
Значення дисперсій по рядках
[189.55555555555554, 208.66666666666666, 156.22222222222223, 122.88888888888887]
```

```
Gp =  0.3080708661417323

Gp = 0.3080708661417323 < Gt = 0.7679
Дисперсія однорідна

Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента
m =  3
N =  4
Отримані значення βi
[222.75000000000003, 1.0833333333333286, -4.583333333333336, 1.9166666666666643]
Отримані значення ti
[59.29759874173182, 0.28839086556023574, 1.2201152004471572, 0.5102299929142649]

f3 =  8
q = 0.05

t0 = 59.29759874173182 > ttабл = 2.306

t1 = 0.28839086556023574 < ttабл = 2.306
b1 - виключається з рівняння

t2 = 1.2201152004471572 < ttабл = 2.306
b2 - виключається з рівняння

t3 = 0.5102299929142649 < ttабл = 2.306
b3 - виключається з рівняння

Перепишемо рівняння враховуючи вилучених коефіцієнтів
y = 213.83333333333363

Підставимо необхідні значення X
y0 = 213.83333333333363
y1 = 213.83333333333363
y2 = 213.83333333333363
y3 = 213.83333333333363

Критерій Фішера
d =  1
f3 =  8
f4 =  3
q = 0.05

Fp = 2.4888451443568353 < Ft = 4.1
Рівняння регресії адекватно оригіналу

Process finished with exit code 0
```

Висновок:

В ході лабораторної роботи, було проведено дробовий трьохфакторний експеримент. Складено матрицю планування, знайдено коефіцієнти рівняння регресії, проведено 3 статистичні перевірки. Під час лабораторної роботи проблем не виникло, що підтверджують дані наведені вище.

Контрольні запитання:

1. **Що називається дробовим факторним експериментом?**

У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Тоді слід скоротити кількість дослідів, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

2. **Для чого потрібно розрахункове значення Кохрена?**

Це значення необхідне щоб перевірити однорідність дисперсії, щоб потім було можливо отримати більш точну статистичну оцінку дисперсії функції відгуку.

3. **Для чого перевіряється критерій Стьюдента?**

Після розрахунків значень коефіцієнтів здійснюється ще одна статистична перевірка – перевірка значущості коефіцієнтів рівняння регресії. Якщо буде встановлено, що якийсь коефіцієнт рівняння регресії незначущий (з обраною ймовірністю), це означає, що відповідний теоретичний коефіцієнт ряду Тейлора дорівнює нулю і необхідно вилучити з рівняння регресії відповідний доданок.

Саме тому ця перевірка має і іншу назву – **нуль-гіпотеза**.

4. **Чим визначається критерій Фішера і як його застосовувати?**

Критерії Фішера необхідні, що перевірити адекватність моделі. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору.