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

Лабораторна робота №5

З дисципліни «Методи оптимізації та планування»

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІО-92
Кушенко Сергій

ПЕРЕВІРИВ:
Регіда П.Г.

Київ 2021 р.

Лабораторна робота № 5

Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)

Мета роботи: Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

$$y_{imax} = 200 + x_{cpmax}$$
$$y_{imin} = 200 + x_{cpmin}$$

где $x_{cpmax} = \frac{x_{1max} + x_{2max} + x_{3max}}{3}$, $x_{cpmin} = \frac{x_{1min} + x_{2min} + x_{3min}}{3}$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант:

№213

$x_{1min} = -2$; $x_{2min} = -10$; $x_{3min} = -3$;

$x_{1max} = 4$; $x_{2max} = 8$; $x_{3max} = 6$;

$y_{imax} = 206$

$y_{imin} = 195$

Код програми:

```
import math
from _pydecimal import Decimal
from scipy.stats import f, t
from functools import reduce
from itertools import compress
import numpy as np
import random

m = 3
N = 15
ymin = 195
ymax = 206

raw_naturalized_factors_table = [[-2, -10, -3],
                                 [-2, 10, 6],
                                 [-2, 8, -3],
                                 [-2, 8, 6],

                                 [4, -10, -3],
                                 [4, -10, 6],
```

```
                                [4, -8, -3],
                                [4, 8, 6],

                                [-2.645, -1, 1.5],
                                [4.645, -1, 1.5],
                                [1, -11.935, 1.5],
                                [1, 9.935, 1.5],
                                [1, -1, -3.967],
                                [1, -1, 6.967],
                                [1, -1, 1.5]]

raw_factors_table = [[-1, -1, -1],
                     [-1, +1, +1],
                     [+1, -1, +1],
                     [+1, +1, -1],

                     [-1, -1, +1],
                     [-1, +1, -1],
                     [+1, -1, -1],
                     [+1, +1, +1],

                     [-1.215, 0, 0],
                     [+1.215, 0, 0],
                     [0, -1.215, 0],
                     [0, +1.215, 0],
                     [0, 0, -1.215],
                     [0, 0, +1.215],
                     [0, 0, 0]]


def generate_factors_table(raw_array):
    return [row + [row[0] * row[1], row[0] * row[2], row[1] * row[2], row[0] * row[1]
* row[2]]
     + list(map(lambda x: round(x ** 2, 5), row))
     for row in raw_array]


def x_i(i):
    try:
        assert i <= 10
    except:
        raise AssertionError(" i повинно бути <=10")
    with_null_factor = list(map(lambda x: [1] + x,
generate_factors_table(raw_factors_table)))
    res = [row[i] for row in with_null_factor]
    return np.array(res)


def cochran_criteria(m, N, y_table):
    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N = {}:
".format(m, N))
    y_variations = [np.var(i) for i in y_table]
    max_y_variation = max(y_variations)
    gp = max_y_variation/sum(y_variations)
    f1 = m - 1
    f2 = N
    p = 0.95
    q = 1-p
    gt = get_cochran_value(f1,f2, q)
    print("Gp = {}; Gt = {}; f1 = {}; f2 = {}; q = {:.2f}".format(gp, gt, f1, f2, q))
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні")
```

```
        return True
    else:
        print("Gp > Gt => дисперсії нерівномірні - потрібно додати експерименти")
        return False


def student_criteria(m, N, y_table, beta_coefficients):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: m =
{}, N = {}: ".format(m, N))
    average_variation = np.average(list(map(np.var, y_table)))

    variation_beta_s = average_variation/N/m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    t_i = np.array([abs(beta_coefficients[i])/standard_deviation_beta_s for i in
range(len(beta_coefficients))])
    f3 = (m-1)*N
    q = 0.05

    t = get_student_value(f3, q)
    importance = [True if el > t else False for el in list(t_i)]

    print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x:
str(round(float(x), 3)), beta_coefficients))))
    print("Коефіцієнти ts:         " + ", ".join(list(map(lambda i:
"{:.2f}".format(i), t_i))))
    print("f3 = {}; q = {}; tтабл = {}".format(f3, q, t))
    beta_i = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123", "β11", "β22",
"β33"]
    importance_to_print = ["важливий" if i else "неважливий" for i in importance]
    to_print = map(lambda x: x[0] + " " + x[1], zip(beta_i, importance_to_print))
    x_i_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23", "x123",
"x1^2", "x2^2", "x3^2"], importance))
    betas_to_print = list(compress(beta_coefficients, importance))
    print(*to_print, sep="; ")
    equation = " ".join(["".join(i) for i in zip(list(map(lambda x:
"{:+.2f}".format(x), betas_to_print)),x_i_names)])
    print("Рівняння регресії без незначимих членів: y = " + equation)
    return importance


def calculate_theoretical_y(x_table, b_coefficients, importance):
    x_table = [list(compress(row, importance)) for row in x_table]
    b_coefficients = list(compress(b_coefficients, importance))
    y_vals = np.array([sum(map(lambda x, b: x*b, row, b_coefficients)) for row in
x_table])
    return y_vals


def fisher_criteria(m, N, d, naturalized_x_table, y_table, b_coefficients,
importance):
    f3 = (m - 1) * N
    f4 = N - d
    q = 0.05

    theoretical_y = calculate_theoretical_y(naturalized_x_table, b_coefficients,
importance)
    theoretical_values_to_print = list(zip(map(lambda x: "x1 = {0[1]}, x2 = {0[2]},
x3 = {0[3]}".format(x),naturalized_x_table),theoretical_y))

    y_averages = np.array(list(map(np.average, y_table)))
    s_ad = m/(N-d)*(sum((theoretical_y-y_averages)**2))
    y_variations = np.array(list(map(np.var, y_table)))
```

```
    s_v = np.average(y_variations)
    f_p = float(s_ad/s_v)
    f_t = get_fisher_value(f3, f4, q)

    print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N =
{}".format(m, N))
    print("Теоретичні значення y для різних комбінацій факторів:")
    print("\n".join(["{arr[0]}: y = {arr[1]}".format(arr=el) for el in
theoretical_values_to_print]))
    print("Fp = {}, Ft = {}".format(f_p, f_t))
    print("Fp < Ft => модель адекватна" if f_p < f_t else "Fp > Ft => модель
неадекватна")
    return True if f_p < f_t else False


def m_ij(*arrays):
    return np.average(reduce(lambda accum, el: accum*el, arrays))


def get_cochran_value(f1, f2, q):
    partResult1 = q / f2
    params = [partResult1, f1, (f2 - 1) * f1]
    fisher = f.isf(*params)
    result = fisher/(fisher + (f2 - 1))
    return Decimal(result).quantize(Decimal('.0001')).__float__()


def get_student_value(f3, q):
    return Decimal(abs(t.ppf(q/2,f3))).quantize(Decimal('.0001')).__float__()


def get_fisher_value(f3,f4, q):
    return Decimal(abs(f.isf(q,f4,f3))).quantize(Decimal('.0001')).__float__()

factors_table = generate_factors_table(raw_factors_table)
for row in factors_table:
    print(row)
naturalized_factors_table = generate_factors_table(raw_naturalized_factors_table)
with_null_factor = list(map(lambda x: [1] + x, naturalized_factors_table))

y_arr = [[random.randint(ymin, ymax) for _ in range(m)] for _ in range(N)]
while not cochran_criteria(m, N, y_arr):
    m+=1
    y_arr = [[random.randint(ymin, ymax) for _ in range(m)] for _ in range(N)]

y_i = np.array([np.average(row) for row in y_arr])

coefficients = [[m_ij(x_i(column)*x_i(row)) for column in range(11)] for row in
range(11)]

free_values = [m_ij(y_i, x_i(i)) for i in range(11)]

beta_coefficients = np.linalg.solve(coefficients, free_values)
print(list(map(int,beta_coefficients)))

importance = student_criteria(m, N, y_arr, beta_coefficients)
d = len(list(filter(None, importance)))
fisher_criteria(m, N, d, naturalized_factors_table, y_arr, beta_coefficients,
importance)
```

Результат виконання:

```
C:\Users\HP\Anaconda3\python.exe C:/МОПЕ/Lab5/lab5.py
[-1, -1, -1, 1, 1, 1, -1, 1, 1, 1]
[-1, 1, 1, -1, -1, 1, -1, 1, 1, 1]
[1, -1, 1, -1, 1, -1, -1, 1, 1, 1]
[1, 1, -1, 1, -1, -1, -1, 1, 1, 1]
[-1, -1, 1, 1, -1, -1, 1, 1, 1, 1]
[-1, 1, -1, -1, 1, -1, 1, 1, 1, 1]
[1, -1, -1, -1, -1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[-1.215, 0, 0, -0.0, -0.0, 0, -0.0, 1.47623, 0, 0]
[1.215, 0, 0, 0.0, 0.0, 0, 0.0, 1.47623, 0, 0]
[0, -1.215, 0, -0.0, 0, -0.0, -0.0, 0, 1.47623, 0]
[0, 1.215, 0, 0.0, 0, 0.0, 0.0, 0, 1.47623, 0]
[0, 0, -1.215, 0, -0.0, -0.0, -0.0, 0, 0, 1.47623]
[0, 0, 1.215, 0, 0.0, 0.0, 0.0, 0, 0, 1.47623]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15:
Gp = 0.17177914110429449; Gt = 0.3346; f1 = 2; f2 = 15; q = 0.05
Gp < Gt => дисперсії рівномірні
[200, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15:
Оцінки коефіцієнтів βs: 200.001, 0.17, -0.822, -0.435, 1.625, -0.875, 0.875, 0.458, -0.159, 0.744, -0.159
Коефіцієнти ts:         498.47, 0.42, 2.05, 1.08, 4.05, 2.18, 2.18, 1.14, 0.40, 1.85, 0.40
f3 = 30; q = 0.05; tтабл = 2.0423
β0 важливий; β1 неважливий; β2 важливий; β3 неважливий; β12 важливий; β13 важливий; β23 важливий; β123 неважливий; β11 неважливий; β22 неважливий; β33 неважливий
Рівняння регресії без незначимих членів: y = +200.00 -0.82x2 +1.62x12 -0.88x13 +0.88x23
```

```
Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15
Теоретичні значення y для різних комбінацій факторів:
x1 = -10, x2 = -3, x3 = 20: y = -466.5360924412506
x1 = 10, x2 = 6, x3 = -20: y = -581.9316979906938
x1 = 8, x2 = -3, x3 = -16: y = -324.7860924412506
x1 = 8, x2 = 6, x3 = -16: y = -550.4316979906938
x1 = -10, x2 = -3, x3 = -40: y = 861.7177904319445
x1 = -10, x2 = 6, x3 = -40: y = 676.572184882501
x1 = -8, x2 = -3, x3 = -32: y = 845.9677904319445
x1 = 8, x2 = 6, x3 = 32: y = 960.072184882501
x1 = -1, x2 = 1.5, x3 = 2.645: y = -531.8974376248406
x1 = -1, x2 = 1.5, x3 = -4.645: y = 934.3085300660912
x1 = -11.935, x2 = 1.5, x3 = -11.935: y = 201.20554622062525
x1 = 9.935, x2 = 1.5, x3 = 9.935: y = 201.2055462206253
x1 = -1, x2 = -3.967, x3 = -1: y = 196.81409072493713
x1 = -1, x2 = 6.967, x3 = -1: y = 205.59700171631343
x1 = -1, x2 = 1.5, x3 = -1: y = 201.20554622062528
Fp = 191241.42377041045, Ft = 2.1646
Fp > Ft => модель неадекватна

Process finished with exit code 0
```

Висновок:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння регресії з ефектом взаємодії та квадратичних членів, знайдено рівняння регресії адекватне об’єкту, коефіцієнти рівняння регресії, складено матрицю планування, проведено 3 статистичні перевірки. Закріплено отримані знання практичним їх використанням при написанні програми, що реалізує завдання лабораторної роботи.