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»

Виконала:
студентка групи ІО-92
Шолотюк Ганна Сергіївна
Номер залікової книжки № 9229
Номер у списку – 21

Перевірив:
Регіда Павло Геннадійович

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).

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.

3. Провести 3 статистичні перевірки.

4. Написати комп'ютерну програму, яка усе це виконує.

$$y_{max} = 200 + x_{cp\ max};$$

$$y_{min} = 200 + x_{cp\ min}$$

де $x_{cp\,max} = \frac{x_{1max} + x_{2max} + x_{3max}}{3}$, $x_{cp\,min} = \frac{x_{1min} + x_{2min} + x_{3min}}{3}$

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
221	10	40	10	60	10	15

Код програми:

```
import xlrd
import random
import numpy as np
import math
import itertools
from prettytable import PrettyTable

class Lab3:
    def __init__(self):
        self.N = 4
        self.m = 3
        self.x_avg_min = round((10 + 10 + 10) / 3)
        self.x_avg_max = round((40 + 60 + 15) / 3)
        self.y_min = 200 + self.x_avg_min
        self.y_max = 200 + self.x_avg_max

        self.factors = [[1, -1, -1, -1],
                        [1, -1, +1, +1],
                        [1, +1, -1, +1],
                        [1, +1, +1, -1]]

        self.matrix = [[random.randint(self.y_min, self.y_max) for i in
range(self.m)] for j in range(4)]
        self.natur_factors = [[10, 10, 10],
                              [10, 60, 15],
                              [40, 10, 60],
                              [40, 60, 10]]

        table0 = PrettyTable()
        table0.field_names = (["N", "X0", "X1", "X2", "X3"] +
                              ["Y{}".format(i+1) for i in range(self.m)])
        for i in range(self.N):
            table0.add_row([i+1] + self.factors[i] + self.matrix[i])
        print(table0)

        table1 = PrettyTable()
        table1.field_names = (["X1", "X2", "X3"]
                              + ["Y{}".format(i + 1) for i in range(self.m)])
        for i in range(self.N):
            table1.add_row(self.natur_factors[i] + self.matrix[i])
        print(table1)

        self.calculate()

    def calculate(self):
        self.avg_Y1 = sum(self.matrix[0][j] for j in range(self.m)) / self.m
        self.avg_Y2 = sum(self.matrix[1][j] for j in range(self.m)) / self.m
        self.avg_Y3 = sum(self.matrix[2][j] for j in range(self.m)) / self.m
        self.avg_Y4 = sum(self.matrix[3][j] for j in range(self.m)) / self.m
        self.avg_Y = [self.avg_Y1, self.avg_Y2, self.avg_Y3, self.avg_Y4]


        self.mx1 = sum(self.natur_factors[i][0] for i in range(self.N)) / self.N
        self.mx2 = sum(self.natur_factors[i][1] for i in range(self.N)) / self.N
        self.mx3 = sum(self.natur_factors[i][2] for i in range(self.N)) / self.N

        self.my = sum(self.avg_Y) / self.N

        self.a1 = sum(self.natur_factors[i][0] * self.avg_Y[i] for i in
range(self.N)) / self.N
```

```
        self.a2 = sum(self.natur_factors[i][1] * self.avg_Y[i] for i in
range(self.N)) / self.N
        self.a3 = sum(self.natur_factors[i][2] * self.avg_Y[i] for i in
range(self.N)) / self.N


        self.a11 = sum((self.natur_factors[i][0]) ** 2 for i in range(self.N)) /
self.N
        self.a22 = sum((self.natur_factors[i][1]) ** 2 for i in range(self.N)) /
self.N
        self.a33 = sum((self.natur_factors[i][2]) ** 2 for i in range(self.N)) /
self.N

        self.a12 = sum(self.natur_factors[i][0] * self.natur_factors[i][1] for i in
range(self.N)) / self.N
        self.a13 = sum(self.natur_factors[i][0] * self.natur_factors[i][2] for i in
range(self.N)) / self.N
        self.a23 = sum(self.natur_factors[i][1] * self.natur_factors[i][2] for i in
range(self.N)) / self.N

        equations_sys_coefs = [[1, self.mx1, self.mx2, self.mx3],
                               [self.mx1, self.a11, self.a12, self.a13],
                               [self.mx2, self.a12, self.a22, self.a23],
                               [self.mx3, self.a13, self.a23, self.a33]]
        equations_sys_free_members = [self.my, self.a1, self.a2, self.a3]
        self.b_coefficients = np.linalg.solve(equations_sys_coefs,
equations_sys_free_members)
        b_normalized_coefficients = np.array([np.average(self.avg_Y),
                              np.average(self.avg_Y * np.array([i[1] for i in
self.factors])),
                              np.average(self.avg_Y * np.array([i[2] for i in
self.factors])),
                              np.average(self.avg_Y * np.array([i[3] for i in
self.factors]))])

        print("\nРівняння регресії для нормованих факторів: \ny =
{0:.2f}{1:+.2f}x1{2:+.2f}x2{3:+.2f}x3".format(*b_normalized_coefficients))
        print("\nРівняння регресії для натуралізованих факторів: \ny =
{0:.2f}{1:+.3f}x1{2:+.2f}x2{3:+.2f}x3".format(*self.b_coefficients))

        self.cochran_criteria(self.m, self.N, self.matrix)



    def cochran_criteria(self, m, N, y_table):
        print("\nПеревірка за критерієм Кохрена:".format(m, N))
        cochran_table = xlrd.open_workbook("Cochran.xls").sheet_by_index(0)
        y_variations = [np.var(i) for i in y_table]
        max_y_variation = max(y_variations)
        gp = max_y_variation/sum(y_variations)
        gt = cochran_table.row_values(N-2)[m-2]/math.pow(10,4)

        if gp < gt:
            print("Gp = {},Gt = {}".format(round(gp, 2), round(gt, 2)))
            print("Gp < Gt => дисперсії рівномірні")
            self.student_criteria(self.m, self.N, self.matrix, self.factors)
        else:
            print("Gp > Gt => дисперсії нерівномірні ")
            self.m = self.m + 1
            self.matrix = [[random.randint(self.y_min, self.y_max) for i in
range(self.m)] for j in range(4)]
```

```
    def student_criteria(self, m, N, y_table, factors_table):
        print("\nПеревірка за критерієм Стьюдента:".format(m, N))
        student_table = xlrd.open_workbook("Student.xls").sheet_by_index(0)

        average_variation = np.average(list(map(np.var, y_table)))
        standard_deviation_beta_s = math.sqrt(average_variation / N / m)

        y_averages = np.array(list(map(np.average, y_table)))
        x_i = np.array([[el[i] for el in factors_table] for i in
range(len(factors_table))])
        coefficients_beta_s = np.array([round(np.average(self.avg_Y*x_i[i]), 2) for
i in range(len(x_i))])
        print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(str,
coefficients_beta_s))))
        t_i = np.array(
            [abs(coefficients_beta_s[i]) / standard_deviation_beta_s for i in
range(len(coefficients_beta_s))])
        print("Коефіцієнти ts: " + ", ".join(list(map(lambda i: "{:.2f}".format(i),
t_i))))
        p = 0.95
        q = 0.05
        t = float(student_table.col_values(3)[(m-1)*N].replace(",", "."))
        self.importance = [True if el > t else False for el in list(t_i)]
        # print result data
        beta_i = ["β{}".format(i) for i in range(N)]
        importance_to_print = ["- важливий" if i else "- неважливий" for i in
self.importance]
        to_print = list(zip(beta_i, importance_to_print))
        x_i_names = [""] + list(itertools.compress(["x{}".format(i) for i in
range(N)], self.importance))[1:]
        betas_to_print = list(itertools.compress(coefficients_beta_s,
self.importance))
        print("{0[0]} {0[1]}\n{1[0]} {1[1]}\n{2[0]} {2[1]}\n{3[0]}
{3[1]}\n".format(*to_print))
        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)
        self.d = len(betas_to_print)
        self.factors_table2 = [np.array([1] + list(i)) for i in self.natur_factors]
        self.fisher_criteria(self.m, self.N, 1, self.factors_table2, self.matrix,
self.b_coefficients, self.importance)

    def calculate_theoretical_y(self, x_table, b_coefficients, importance):
        x_table = [list(itertools.compress(row, importance)) for row in x_table]
        b_coefficients = list(itertools.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(self, m, N, d, factors_table, matrix, b_coefficients,
importance):
        print("\nПеревірка за критерієм Фішера:".format(m, N))
        fisher_table = xlrd.open_workbook("Fisher.xls").sheet_by_index(0)

        f3 = (m - 1) * N
        f4 = N - d

        theoretical_y = self.calculate_theoretical_y(factors_table, b_coefficients,
importance)
        theoretical_values_to_print = list(
```

```
            zip(map(lambda x: "x1 = {0[1]}, x2 = {0[2]}, x3 = {0[3]}".format(x),
factors_table), theoretical_y))

        y_averages = np.array(list(map(np.average, matrix)))
        s_ad = m / (N - d) * (sum((theoretical_y - y_averages) ** 2))
        y_variations = np.array(list(map(np.var, matrix)))
        s_v = np.average(y_variations)
        f_p = float(s_ad / s_v)
        f_t = float((fisher_table.row_values(f3) if f3 <= 30 else
fisher_table.row_values(30))[f4].replace(",", "."))
        print("Fp = {}, Ft = {}".format(round(f_p, 2), round(f_t, 2)))
        print("Fp < Ft => модель адекватна" if f_p < f_t else "Fp > Ft => модель
неадекватна")
Lab3()
```

Результати тестування:

```
+----+----+----+-----+-----+-----+
| X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+-----+-----+-----+
| 10 | 10 | 10 | 213 | 228 | 221 |
| 10 | 60 | 15 | 233 | 213 | 229 |
| 40 | 10 | 60 | 211 | 236 | 210 |
| 40 | 60 | 10 | 227 | 224 | 220 |
+----+----+----+-----+-----+-----+

Рівняння регресії для нормованих факторів:
y = 222.08-0.75x1+2.25x2-0.08x3

Рівняння регресії для натуралізованих факторів:
y = 220.31-0.045x1+0.09x2-0.01x3


Перевірка за критерієм Кохрена:
Gp = 0.55,Gt = 0.77
Gp < Gt => дисперсії рівномірні

Перевірка за критерієм Стьюдента:
Оцінки коефіцієнтів βs: 222.08, -0.75, 2.25, -0.08
Коефіцієнти ts: 94.50, 0.32, 0.96, 0.03
β0 - важливий
β1 - неважливий
β2 - неважливий
β3 - неважливий

Рівняння регресії без незначимих членів: y = +222.08

Перевірка за критерієм Фішера:
Fp = 0.53, Ft = 4.07
Fp < Ft => модель адекватна
```

Відповіді на контрольні питання:

1. ПФЕ – повний факторний експеримент,- це коли використовуються усі можливі комбінації рівнів факторів; при ПФЕ кількість комбінацій Nп=rk.

ДФЕ – дробовий факторний експеримент,- це коли використовуються частка ПФЕ, яка кратна степеню двійки, тобто Nд=2-1Nп (напівреплiка), Nд=2-2Nп (1/4 репліки) тощо.

2. Перевірка за критерієм Кохрена виконується для статичного визначення однорідності дисперсії. Перш, ніж розраховувати значення коефіцієнтів рівняння регресії по результатах досліджень , тобто по значенням yi (i=1,n), необхідно переконатися, що дисперсія однорідна. Якщо N≥3 (k≥2),-- тоді використовується критерій Кохрена, якщо N=2 (k=1),-- тоді використовується критерій Фішера або критерій Романовського.

3. Перевірка за критерієм Стьюдента:

Можливість встановлення нуль-гіпотези в теорії планування експерименту пропонується перевіряти з допомогою оцінок значень коефіцієнтів рівняння регресії, перевіряючи гіпотезу стосовно їх значущості. Тому перевірка значущості коефіцієнтів лінійної регресії виконується з допомогою t-критерія Стьюдента.

4. Критерій Фішера і його визначення:

Кількість значущих доданків рівняння регресії, що залишаться після нуль-гіпотези, позначається як d і її значення використовується при перевірці адекватності моделі оригіналу по критерію Фішера.

Спочатку знаходять значення статистичної оцінки дисперсії, розраховують значення критерія Фішера. Визначають число ступенів свободи f3 та f4. Обирають рівень значущості q. По спеціальним таблицям Фішера знаходять критичне (табличне) значення критерія Фішера F кр, яке відповідає значенням q, f3та f4. Нарешті, порівнюють значення F із значенням Fкр та роблять висновки щодо значущості.

Висновки:

В ході виконання лабораторної роботи було проведено трьохфакторний експеримент з використанням лінійного рівняння регресії. Також було складено матрицю планування, знайдено коефіцієнти рівняння регресії, проведено три статичні перевірки: перевірка однорідності дисперсії за критерієм Кохрана, перевірка значущості коефіцієнтів за критерієм Стьюдента та перевірка адекватності за критерієм Фішера. Отримані теоретичні знання закріплено практичними, а саме: написано програму, що реалізує дане завдання лабораторної роботи. Успішне виконання програми продемонстровано скріншотами тестування, тобто, кінцеву мету досягнуто.