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

Лабораторна робота №4
з дисципліни «Методи оптимізації та планування експерименту»

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконав:
студент групи ІВ-92
Коптюх Н.Є
Залікова книжка № ІВ-9214
Варіант: 212
Перевірив:
Регіда П.Г.

Київ 2021

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

Завдання:

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.
3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Варіант:

212	10	60	-35	15	10	15

Код програми:

```
import numpy as np
from itertools import product, combinations
from scipy.stats import t, f

np.set_printoptions(formatter={'float_kind': lambda x: "%.2f"%(x)})



def table_student(prob, n, m):
    x_vec = [i*0.0001 for i in range(int(5/0.0001))]
    par = 0.5 + prob/0.1*0.05
    f3 = (m - 1) * n
    for i in x_vec:
        if abs(t.cdf(i, f3) - par) < 0.000005:
            return i


def table_fisher(prob, n, m, d):
    x_vec = [i*0.001 for i in range(int(10/0.001))]
    f3 = (m - 1) * n
    for i in x_vec:
        if abs(f.cdf(i, n-d, f3)-prob) < 0.0001:
            return i


def make_norm_plan_matrix(plan_matrix, matrix_of_min_and_max_x):
    X0 = np.mean(matrix_with_min_max_x, axis=1)
    interval_of_change = np.array([(matrix_of_min_and_max_x[i, 1] - X0[i]) for i in
range(len(plan_matrix[0]))])
    X_norm = np.array(
```

```
        [[round((plan_matrix[i, j] - X0[j]) / interval_of_change[j], 3) for j in
range(len(plan_matrix[i]))]
         for i in range(len(plan_matrix))])
    return X_norm


def cochran_check(Y_matrix):
    mean_Y = np.mean(Y_matrix, axis=1)
    dispersion_Y = np.mean((Y_matrix.T - mean_Y) ** 2, axis=0)
    Gp = np.max(dispersion_Y) / (np.sum(dispersion_Y))
    fisher = table_fisher(0.95, N, m, 1)
    Gt = fisher / (fisher + (m - 1) - 2)
    return Gp < Gt


def students_t_test(norm_matrix, Y_matrix):
    mean_Y = np.mean(Y_matrix, axis=1)
    dispersion_Y = np.mean((Y_matrix.T - mean_Y) ** 2, axis=0)
    mean_dispersion = np.mean(dispersion_Y)
    sigma = np.sqrt(mean_dispersion / (N * m))
    betta = np.mean(norm_matrix.T * mean_Y, axis=1)
    t = np.abs(betta) / sigma
    if (m - 1) * N > 32:
        return np.where(t > table_student(0.95, N, m))
    return np.where(t > table_student(0.95, N, m))


def phisher_criterion(Y_matrix, d):
    if d == N:
        return False
    Sad = m / (N - d) * np.mean(check1 - mean_Y)
    mean_dispersion = np.mean(np.mean((Y_matrix.T - mean_Y) ** 2, axis=0))
    Fp = Sad / mean_dispersion
    if (m-1)*N > 32:
        if N-d > 6:
            return table_fisher(0.95, N, m, d)
        return Fp < table_fisher(0.95, N, m, d)
    if N - d > 6:
        return Fp < table_fisher(0.95, N, m, d)
    return Fp < table_fisher(0.95, N, m, d)


matrix_with_min_max_x = np.array([[10, 60], [-35, 15], [10, 15]])
m = 6
N = 8
norm_matrix = np.array(list(product("01", repeat=3)), dtype=np.int)
norm_matrix[norm_matrix == 0] = -1
norm_matrix = np.insert(norm_matrix, 0, 1, axis=1)
plan_matrix = np.empty((8, 3))
for i in range(len(norm_matrix)):
    for j in range(1, len(norm_matrix[i])):
        if j == 1:
            if norm_matrix[i, j] == -1:
                plan_matrix[i, j-1] = 10
            elif norm_matrix[i, j] == 1:
                plan_matrix[i, j-1] = 60
        elif j == 2:
            if norm_matrix[i, j] == -1:
                plan_matrix[i, j-1] = -35
            elif norm_matrix[i, j] == 1:
                plan_matrix[i, j-1] = 15
        elif j == 3:
            if norm_matrix[i, j] == -1:
                plan_matrix[i, j-1] = 10
            elif norm_matrix[i, j] == 1:
                plan_matrix[i, j-1] = 15
plan_matr = np.insert(plan_matrix, 0, 1, axis=1)
```

```
Y_matrix = np.random.randint(200 + np.mean(matrix_with_min_max_x, axis=0)[0],
                             200 + np.mean(matrix_with_min_max_x, axis=0)[1], size=(N,
m))
mean_Y = np.mean(Y_matrix, axis=1)
combination = list(combinations(range(1, 4), 2))
for i in combination:
    plan_matr = np.append(plan_matr, np.reshape(plan_matr[:, i[0]]*plan_matr[:, i[1]],
(8, 1)), axis=1)
    norm_matrix = np.append(norm_matrix, np.reshape(norm_matrix[:, i[0]]*norm_matrix[:,
i[1]], (8, 1)), axis=1)
plan_matr = np.append(plan_matr, np.reshape(plan_matr[:, 1]*plan_matr[:,
2]*plan_matr[:, 3], (8, 1)), axis=1)
norm_matrix = np.append(norm_matrix, np.reshape(norm_matrix[:, 1]*norm_matrix[:,
2]*norm_matrix[:, 3], (8, 1)), axis=1)

if cochran_check(Y_matrix):
    b_natura = np.linalg.lstsq(plan_matr, mean_Y, rcond=None)[0]
    b_norm = np.linalg.lstsq(norm_matrix, mean_Y, rcond=None)[0]
    check1 = np.sum(b_natura * plan_matr, axis=1)
    check2 = np.sum(b_norm * norm_matrix, axis=1)
    indexes = students_t_test(norm_matrix, Y_matrix)
    print("Матриця плану експерименту: \n", plan_matr)
    print("Нормована матриця: \n", norm_matrix)
    print("Матриця відгуків: \n", Y_matrix)
    print("Середні значення У: ", mean_Y)
    print("Натуралізовані коефіцієнти: ", b_natura)
    print("Нормовані коефіцієнти: ", b_norm)
    print("Перевірка 1: ", check1)
    print("Перевірка 2: ", check2)
    print("Індекси коефіцієнтів, які задовольняють критерію Стьюдента: ",
np.array(indexes)[0])
    print("Критерій Стьюдента: ", np.sum(b_natura[indexes] * np.reshape(plan_matr[:,
indexes], (N, np.size(indexes))), axis=1))
    if phisher_criterion(Y_matrix, np.size(indexes)):
        print("Рівняння регресії адекватно оригіналу.")
    else:
        print("Рівняння регресії неадекватно оригіналу.")
else:
    print("Дисперсія неоднорідна!")
```

Результати виконання:

```
Матриця плану експерименту:
 [[1.00 10.00 -35.00 10.00 -350.00 100.00 -350.00 -3500.00]
 [1.00 10.00 -35.00 15.00 -350.00 150.00 -525.00 -5250.00]
 [1.00 10.00 15.00 10.00 150.00 100.00 150.00 1500.00]
 [1.00 10.00 15.00 15.00 150.00 150.00 225.00 2250.00]
 [1.00 60.00 -35.00 10.00 -2100.00 600.00 -350.00 -21000.00]
 [1.00 60.00 -35.00 15.00 -2100.00 900.00 -525.00 -31500.00]
 [1.00 60.00 15.00 10.00 900.00 600.00 150.00 9000.00]
 [1.00 60.00 15.00 15.00 900.00 900.00 225.00 13500.00]]
```

```
Нормована матриця:
 [[ 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]]
Матриця відгуків:
 [[214 217 224 205 210 209]
 [208 228 198 223 211 200]
 [195 220 200 227 228 217]
 [220 199 202 223 212 214]
 [201 213 201 209 223 204]
 [205 212 227 210 201 207]
 [204 219 204 203 203 198]
 [212 224 220 205 228 202]]
Середні значення У:  [213.17 211.33 214.50 211.67 208.50 210.33 205.17 215.17]
Натуралізовані коефіцієнти:  [224.79 -0.56 0.16 -0.91 -0.01 0.04 -0.01 0.00]
Нормовані коефіцієнти:  [211.23 -1.44 0.40 0.90 -0.02 2.06 0.90 1.15]
Перевірка 1:  [213.17 211.33 214.50 211.67 208.50 210.33 205.17 215.17]
Перевірка 2:  [213.17 211.33 214.50 211.67 208.50 210.33 205.17 215.17]
Індекси коефіцієнтів, які задовольняють критерію Стьюдента:  [0]
Критерій Стьюдента:  [224.79 224.79 224.79 224.79 224.79 224.79 224.79 224.79]
Рівняння регресії адекватно оригіналу.
```

Висновок: Під час виконання роботи проблем не виникало. Отримані результати збігаються з очікуваними.