Міністерство освіти і науки України
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 » на тему

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

Виконав:
студент ІІ курсу ФІОТ
групи ІО - 93
Ільків Максим
Номер залікової книжки: ІО - 9313

Перевірив:
ас. Регіда П.Г.

Київ – 2021

Мета роботи: провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання на лабораторну роботу:

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.

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

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

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 завдання:

№варіанта	x_1		x_2		x_3	
	min	max	min	Max	min	max
311	-25	-5	10	60	-5	60

Роздруківка тексту програми:

```
import numpy as np
import random
from scipy.stats import f, t
from numpy.linalg import solve
import sklearn.linear_model as lm


# Варіант - 311 (-25, 5, 10, 60, -5, 60)
```

```
def calc_dispersion(y: np.ndarray, y_average: list, n: int, m: int) -> list:
    result = []
    for i in range(n):
        s = sum([(y_average[i] - y[i][j]) ** 2 for j in range(m)]) / m
        result.append(round(s, 3))
    return result


def calc_regression(x: list, b: list) -> list:
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y


def planning_matrix_interaction(n: int, m: int) -> tuple:
    x_normalized = [[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]]
    y = np.zeros(shape=(n, m), dtype=np.int64)
    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)
    for x in x_normalized:
        x.append(x[1] * x[2])
        x.append(x[1] * x[3])
        x.append(x[2] * x[3])
        x.append(x[1] * x[2] * x[3])
    x_normalized = np.array(x_normalized[:len(y)])
    x = np.ones(shape=(len(x_normalized), len(x_normalized)), dtype=np.int64)
    for i in range(len(x_normalized)):
        for j in range(1, 4):
            if x_normalized[i][j] == -1:
                x[i][j] = range_x[j - 1][0]
            else:
                x[i][j] = range_x[j - 1][1]
    for i in range(len(x)):
        x[i][4] = x[i][1] * x[i][2]
        x[i][5] = x[i][1] * x[i][3]
        x[i][6] = x[i][2] * x[i][3]
        x[i][7] = x[i][1] * x[i][3] * x[i][2]
    print(f'Матриця планування при n = {n} та m = {m}:')
    print('З кодованими значеннями:')
    print('\n     X0    X1    X2    X3  X1X2  X1X3  X2X3 X1X2X3   Y1    Y2     Y3')
```

```
    print(np.concatenate((x, y), axis=1))
    print('Нормовані значення:')
    print(x_normalized)
    return x, y, x_normalized


def find_coefficient(X: list, Y: list, is_normalized=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(X, Y)
    coefficients_b = skm.coef_
    if is_normalized == 1:
        print('Коефіцієнти з нормованими X:')
    else:
        print('Коефіцієнти рівняння регресії')
    coefficients_b = [round(i, 3) for i in coefficients_b]
    print(coefficients_b)
    return coefficients_b


def bs(x: list, y, y_average: list, n: int) -> list:
    result = [sum(1 * y for y in y_average) / n]
    for i in range(7):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_average)) / n
        result.append(b)
    return result


def student_criteria_2(x: list, y, y_average: list, n: int, m: int) -> list:
    student_squared = calc_dispersion(y, y_average, n, m)
    student_squared_average = sum(student_squared) / n
    students_bs = (student_squared_average / n / m) ** 0.5
    Bs = bs(x, y, y_average, n)
    ts = [round(abs(B) / students_bs, 3) for B in Bs]
    return ts


def student_criteria(x: list, y_average: list, n: int, m: int, dispersion: list) -> list:
    dispersion_average = sum(dispersion) / n
    students_beta = (dispersion_average / n / m) ** 0.5
    beta = [sum(1 * y for y in y_average) / n]
    for i in range(3):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_average)) / n
        beta.append(b)
    t = [round(abs(b) / students_beta, 3) for b in beta]
    return t


def fisher_criteria(y: list, y_average: list, y_new: list, n: int, m: int, d: int, dispersion: list) -> float:
    S_ad = m / (n - d) * sum([(y_new[i] - y_average[i]) ** 2 for i in range(len(y))])
```

```
    dispersion_average = sum(dispersion) / n

    return S_ad / dispersion_average


def check(X: list, Y: np.ndarray, B: list, n: int, m: int, is_Normalized=False):
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05
    y_average = [round(sum(i) / len(i), 3) for i in Y]
    print('Середнє знач. y: ', y_average)
    dispersion_array = calc_dispersion(Y, y_average, n, m)
    qq = (1 + 0.95) / 2
    students_criteria_table = t.ppf(df=f3, q=qq)
    ts = student_criteria_2(X[:, 1:], Y, y_average, n, m)
    temp_cohren = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)
    cohren_criteria_table = temp_cohren / (temp_cohren + f1 - 1)
    Gp = max(dispersion_array) / sum(dispersion_array)
    print('Дисперсія: ', dispersion_array)
    print(f'Gp = {Gp}')
    if Gp < cohren_criteria_table:
        print(f'Дисперсії однорідні з ймовірністю {1 - q}')
    else:
        print(f'Дисперсія неоднорідна. Збільшуємо к-сть дослідів з {m} до {m + 1}')
        m += 1
        with_interaction_effect(n, m)

    print('\nКритерій Стьюдента:\n', ts)
    res = [t for t in ts if t > students_criteria_table]
    final_k = [B[i] for i in range(len(ts)) if ts[i] in res]
    print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з рівняння.'.format(
        [round(i, 3) for i in B if i not in final_k]))

    y_new = []
    for j in range(n):
        y_new.append(calc_dispersion([X[j][i] for i in range(len(ts)) if ts[i] in res], final_k))

    print(f'\nЗначення "y" з коефіцієнтами {final_k}')
    print(y_new)

    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        print('')
        return
    f4 = n - d

    Fp = fisher_criteria(Y, y_average, y_new, n, m, d, dispersion_array)
```

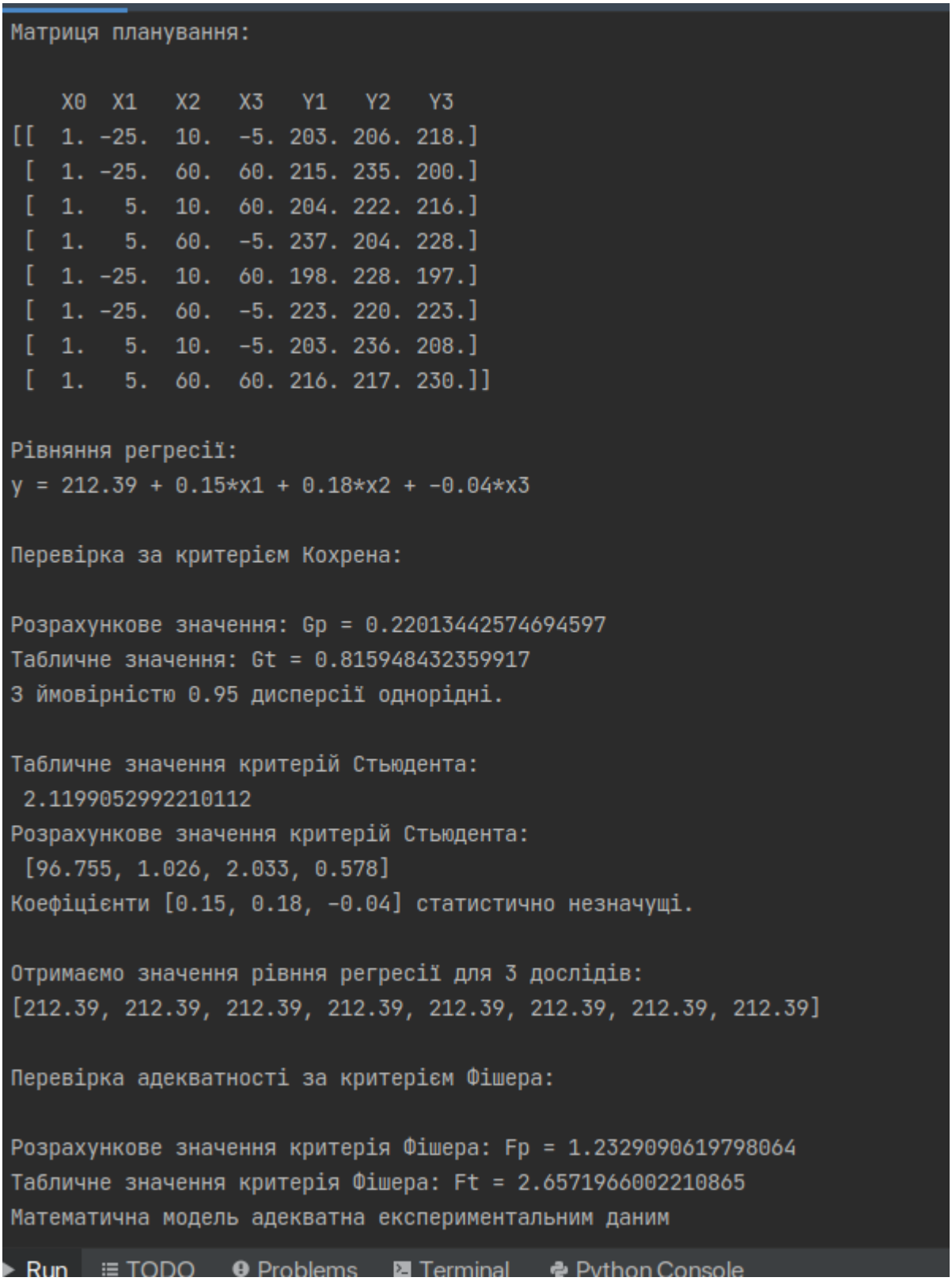
```
    Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', Fp)
    print('Ft =', Ft)
    if Fp < Ft:
        print('Математична модель адекватна експериментальним даним')
        return True
    else:
        print('Математична модель не адекватна експериментальним даним')
        return False


def with_interaction_effect(n: int, m: int):
    X, Y, X_normalized = planning_matrix_interaction(n, m)

    y_aver = [round(sum(i) / len(i), 3) for i in Y]

    B_normalized = find_coefficient(X_normalized, y_aver, norm=True)

    return check(X_normalized, Y, B_normalized, n, m, norm=True)


def planning_matrix_linear(n, m, range_x):
    x_normalized = np.array([[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]])
    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)

    x_normalized = x_normalized[:len(y)]

    x = np.ones(shape=(len(x_normalized), len(x_normalized[0])))
    for i in range(len(x_normalized)):
        for j in range(1, len(x_normalized[i])):
            if x_normalized[i][j] == -1:
                x[i][j] = range_x[j - 1][0]
            else:
                x[i][j] = range_x[j - 1][1]

    print('\nМатриця планування:')
```

```
    print('\n    X0  X1   X2   X3   Y1   Y2   Y3  ')
    print(np.concatenate((x, y), axis=1))

    return x, y, x_normalized


def regression_equation(x: np.ndarray, y: np.ndarray, n: int) -> tuple:
    y_average = [round(sum(i) / len(i), 2) for i in y]

    mx1 = sum(x[:, 1]) / n
    mx2 = sum(x[:, 2]) / n
    mx3 = sum(x[:, 3]) / n

    my = sum(y_average) / n

    a1 = sum([y_average[i] * x[i][1] for i in range(len(x))]) / n
    a2 = sum([y_average[i] * x[i][2] for i in range(len(x))]) / n
    a3 = sum([y_average[i] * x[i][3] for i in range(len(x))]) / n

    a12 = sum([x[i][1] * x[i][2] for i in range(len(x))]) / n
    a13 = sum([x[i][1] * x[i][3] for i in range(len(x))]) / n
    a23 = sum([x[i][2] * x[i][3] for i in range(len(x))]) / n

    a11 = sum([i ** 2 for i in x[:, 1]]) / n
    a22 = sum([i ** 2 for i in x[:, 2]]) / n
    a33 = sum([i ** 2 for i in x[:, 3]]) / n

    X = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a12, a22, a23], [mx3, a13, a23, a33]]
    Y = [my, a1, a2, a3]
    B = [round(i, 2) for i in solve(X, Y)]

    print('\nРівняння регресії:')
    print(f'y = {B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

    return y_average, B


def linear(n: int, m: int):
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    x, y, x_norm = planning_matrix_linear(n, m, range_x)

    y_average, B = regression_equation(x, y, n)

    dispersion_arr = calc_dispersion(y, y_average, n, m)
```

```
    temp_cohren = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)
    cohren_cr_table = temp_cohren / (temp_cohren + f1 - 1)
    Gp = max(dispersion_arr) / sum(dispersion_arr)

    print('\nПеревірка за критерієм Кохрена:\n')
    print(f'Розрахункове значення: Gp = {Gp}'
          f'\nТабличне значення: Gt = {cohren_cr_table}')
    if Gp < cohren_cr_table:
        print(f'З ймовірністю {1 - q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити ксть дослідів")
        m += 1
        linear(n, m)

    qq = (1 + 0.95) / 2
    student_cr_table = t.ppf(df=f3, q=qq)
    student_t = student_criteria(x_norm[:, 1:], y_average, n, m, dispersion_arr)

    print('\nТабличне значення критерій Стьюдента:\n', student_cr_table)
    print('Розрахункове значення критерій Стьюдента:\n', student_t)
    res_student_t = [temp for temp in student_t if temp > student_cr_table]
    final_coefficients = [B[student_t.index(i)] for i in student_t if i in res_student_t]
    print('Коефіцієнти {} статистично незначущі.'.
          format([i for i in B if i not in final_coefficients]))

    y_new = []
    for j in range(n):
        y_new.append(
            calc_regression([x[j][student_t.index(i)] for i in student_t if i in res_student_t],
final_coefficients))

    print(f'\nОтримаємо значення рівння регресії для {m} дослідів: ')
    print(y_new)

    d = len(res_student_t)
    f4 = n - d
    Fp = fisher_criteria(y, y_average, y_new, n, m, d, dispersion_arr)
    Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

    print('\nПеревірка адекватності за критерієм Фішера:\n')
    print('Розрахункове значення критерія Фішера: Fp =', Fp)
    print('Табличне значення критерія Фішера: Ft =', Ft)
    if Fp < Ft:
        print('Математична модель адекватна експериментальним даним')
        return True
    else:
        print('Математична модель не адекватна експериментальним даним')
        return False
```

```
def main(n: int, m: int):
    main_1 = linear(n, m)
    if not main_1:
        interaction_effect = with_interaction_effect(n, m)
        if not interaction_effect:
            main(n, m)


if __name__ == '__main__':
    range_x = ((-25, 5), (10, 60), (-5, 60))
    y_max = 200 + int(sum([x[1] for x in range_x]) / 3)
    y_min = 200 + int(sum([x[0] for x in range_x]) / 3)
    main(8, 3)
```

Результати роботи програми:


Матриця планування:
X0 X1 X2 X3 Y1 Y2 Y3
[[1. -25. 10. -5. 203. 206. 218.]
[1. -25. 60. 60. 215. 235. 200.]
[1. 5. 10. 60. 204. 222. 216.]
[1. 5. 60. -5. 237. 204. 228.]
[1. -25. 10. 60. 198. 228. 197.]
[1. -25. 60. -5. 223. 220. 223.]
[1. 5. 10. -5. 203. 236. 208.]
[1. 5. 60. 60. 216. 217. 230.]]
Рівняння регресії:
y = 212.39 + 0.15*x1 + 0.18*x2 + -0.04*x3
Перевірка за критерієм Кохрена:
Розрахункове значення: Gp = 0.22013442574694597
Табличне значення: Gt = 0.815948432359917
З ймовірністю 0.95 дисперсії однорідні.
Табличне значення критерій Стьюдента:
2.1199052992210112
Розрахункове значення критерій Стьюдента:
[96.755, 1.026, 2.033, 0.578]
Коефіцієнти [0.15, 0.18, -0.04] статистично незначущі.
Отримаємо значення рівння регресії для 3 дослідів:
[212.39, 212.39, 212.39, 212.39, 212.39, 212.39, 212.39, 212.39]
Перевірка адекватності за критерієм Фішера:
Розрахункове значення критерія Фішера: Fp = 1.2329090619798064
Табличне значення критерія Фішера: Ft = 2.6571966002210865
Математична модель адекватна експериментальним даним
Run
TODO
Problems
Terminal
Python Console

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння регресії з ефектом взаємодії, складено матрицю планування експерименту, було визначено коефіцієнти рівняння регресії(натуралізовані та нормовані), виконано перевірку правильності розрахунку коефіцієнтів рівняння регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера). При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05.