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 № 3

з дисципліни «Методи наукових досліджень» на тему

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІВ-92
Злочевський Нікіта Вікторович
Варіант: 209

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Хід роботи

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)

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

3. Провести 3 статистичні перевірки.

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

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
209	-20	15	-30	45	-30	-15

Лістинг програми

```
import random
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial


def plan_matrix(n, m):
    x_range = [(-20, 15), (-30, 45), (-30, -15)]
    avg_x_min = (-20 - 30 - 30) / 3
    avg_x_max = (15 + 45 - 15) / 3
    y_max = round(200 + avg_x_max)
    y_min = round(200 + avg_x_min)
    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)
    factors_table = 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]])
    factors_table = factors_table[:len(y)]

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

    print('\nМатриця планування')
    print(np.concatenate((x, y), axis=1))

    return x, y, factors_table


def regression(x, b):
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y


def find_coefficient(x, y_average, n):
    mx1 = sum(x[:, 1]) / n
    mx2 = sum(x[:, 2]) / n
    mx3 = sum(x[:, 3]) / n
    my = sum(y_average) / 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
    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
```

```
    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'{B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

    return B


def dispersion(y, y_average, n, m):
    res = []
    for i in range(n):
        s = sum([(y_average[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(s)
    return res


def cochran_test(y, y_average, n, m):
    S_kv = dispersion(y, y_average, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp


def bs(x, y, y_average, n):
    res = [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
        res.append(b)
    return res


def student_test(x, y, y_average, n, m):
    S_kv = dispersion(y, y_average, n, m)
    s_kv_aver = sum(S_kv) / n

    s_Bs = (s_kv_aver / n / m) ** 0.5
    Bs = bs(x, y, y_average, n)
    ts = [abs(B) / s_Bs for B in Bs]

    return ts


def fisher_test(y, y_average, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_average[i]) ** 2 for i in range(len(y))])
    S_kv = dispersion(y, y_average, n, m)
    S_kv_average = sum(S_kv) / n
    Fp = S_ad / S_kv_average
    return Fp


def cochran(f1, f2, q=0.05):
    q1 = q / f1
    fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fisher_value / (fisher_value + f1 - 1)


def main(n, m):
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
```

```
    q = 0.05

    student = partial(t.ppf, q=1 - 0.025)
    t_student = student(df=f3)
    G_kr = cochran(f1, f2)
    x, y, x_norm = plan_matrix(n, m)
    y_average = [round(sum(i) / len(i), 2) for i in y]

    B = find_coefficient(x, y_average, n)
    Gp = cochran_test(y, y_average, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'З ймовірністю {1 - q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити ксть дослідів")
        m += 1
        main(n, m)

    ts = student_test(x_norm[:, 1:], y, y_average, n, m)
    print('\nКритерій Стьюдента:\n', ts)
    res = [t for t in ts if t > t_student]
    final_k = [B[ts.index(i)] for i in ts if i in res]
    print('Коефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.format(
        [i for i in B if i not in final_k]))

    y_new = []
    for j in range(n):
        y_new.append(regression([x[j][ts.index(i)] for i in ts if i in res], final_k))

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

    d = len(res)
    f4 = n - d
    Fp = fisher_test(y, y_average, y_new, n, m, d)

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

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


if __name__ == '__main__':
    main(6, 7)
```

Результат роботи програми

```
Матриця планування
[[  1. -20. -30. -30. 207. 179. 188. 214. 204. 212. 194.]
 [  1. -20.  45. -15. 194. 193. 197. 181. 180. 211. 193.]
 [  1.  15. -30. -15. 180. 210. 210. 200. 174. 178. 191.]
 [  1.  15.  45. -30. 195. 176. 185. 182. 195. 176. 199.]
 [  1. -20. -30. -15. 197. 207. 194. 184. 175. 181. 215.]
 [  1. -20.  45. -30. 215. 192. 215. 182. 174. 212. 212.]]

Рівняння регресії
186.72 + -0.2*x1 + -0.04*x2 + -0.27*x3

Перевірка за критерієм Кохрена
Gp = 0.27107785603975737
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [100.01336862781376, 34.97273574312795, 0.42934511568365685, 0.7728212082305823]
Коефіцієнти [-0.04, -0.27] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [186.72, -0.2]
[190.72, 190.72, 183.72, 183.72, 190.72, 190.72]

Перевірка адекватності за критерієм Фішера
Fp = 2.865587101582547
F_t = 2.6335320942137526
Математична модель не адекватна експериментальним даним
```

Відповіді на контрольні запитання

1. **Що називається дробовим факторним експериментом?**
   У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, 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?**
   Статистична перевірка за критерієм Кохрена використовується для перевірки гіпотези про однорідність дисперсії з довірчою ймовірністю p. Якщо експериментальне значення $G < G_{кр}$, яке обирається з таблиці, то гіпотеза підтверджується, якщо ні, то відповідно не підтверджується.

3. **Для чого перевіряється критерій Стьюдента?**
   Критерій Стьюдента використовується для перевірки значимості коефіцієнта рівняння регресії. Якщо з'ясувалось, що будь-який коефіцієнт рівняння регресії не значимий, то відповідний $b_i = 0$ і відповідний член рівняння регресії треба викреслити. Іноді ця статистична перевірка має назву «нуль-гіпотеза». Якщо експериментальне значення $t > t_{кр}$, тонуль-гіпотеза не підтверджується і даний коефіцієнт значимий, інакше нуль-гіпотеза підтверджується і даний коефіцієнт рівняння регресії не значимий.
4. **Чим визначається критерій Фішера і як його застосовувати?**
   Критерій Фішера застосовується для перевірки адекватності моделі (рівняння регресії) оригіналу (експериментальним даним). Обчислюється експериментальне значення F, яке порівнюється з $F_{кр}$, взятим з таблиці залежно від кількості значимих коефіцієнтів та ступенів вільності. Якщо $F < F_{кр}$, то модель адекватна оригіналу, інакше – ні.

Висновок:

На цій лабораторній роботі ми вивчили основні поняття, визначення, принципи теорії планування експерименту, на основі яких вивчили побудову формалізованих алгоритмів проведення експерименту і отримання формалізованої моделі об'єкта. Закріпили отримані знання їх практичним використанням при написанні програми у середовищі Pycharm на мові python, що реалізує завдання лабораторної роботи.