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 №3

з дисципліни « Методи оптимізації та планування » на тему

«Проведення трьохфакторного експерименту
з використанням лінійного рівняння регресії»

Виконала:
студентка ІІ курсу ФІОТ
групи ІВ – 92
Гайдукевич Марія
Номер залікової книжки: ІВ - 9206

Перевірив:
ас. Регіда П.Г.

Київ – 2021

Мета роботи: 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).

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

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

де $x_{cp\ max} = (1/3)(x_{1max}+x_{2max}+x_{3max})$, $x_{cp\ min} = (1/3)(x_{1min}+x_{2min}+x_{3min})$

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

3. Провести 3 статистичні перевірки.

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

Варіант завдання:

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
204	15	45	15	50	15	30

Роздруківка тексту програми:

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

x_range = [(15, 45), (15, 50), (15, 30)]
x_aver_max = (45 + 50 + 30) / 3
x_aver_min = (15 + 15 + 15) / 3

y_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)


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


def plan_matrix(n, m):
    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_norm = 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]])
    x_norm = x_norm[:len(y)]

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


def find_coefficient(x, y_aver, n):
    mx1 = sum(x[:, 1]) / n
    mx2 = sum(x[:, 2]) / n
    mx3 = sum(x[:, 3]) / n
    my = sum(y_aver) / 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_aver[i] * x[i][1] for i in range(len(x))]) / n
    a2 = sum([y_aver[i] * x[i][2] for i in range(len(x))]) / n
    a3 = sum([y_aver[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 s_kv(y, y_aver, n, m):
    res = []
    for i in range(n):
        s = sum([(y_aver[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(s)
    return res


def kriteriy_cochrena(y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp


def bs(x, y, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]
    for i in range(3):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)
    return res


def kriteriy_studenta(x, y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n

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

    return ts


def kriteriy_fishera(y, y_aver, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in range(len(y))])
    S_kv = s_kv(y, y_aver, n, m)
    S_kv_aver = sum(S_kv) / n

    return S_ad / S_kv_aver


def cohren(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 = cohren(f1, f2)

    x, y, x_norm = plan_matrix(n, m)
    y_aver = [round(sum(i) / len(i), 2) for i in y]

    B = find_coefficient(x, y_aver, n)

    Gp = kriteriy_cochrena(y, y_aver, 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 = kriteriy_studenta(x_norm[:, 1:], y, y_aver, 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
    F_p = kriteriy_fishera(y, y_aver, y_new, n, m, d)

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

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


if __name__ == '__main__':
    main(4, 4)
```

Результати роботи програми:

```
Матриця планування
[[  1.  15.  15.  15. 225. 226. 239. 218.]
 [  1.  15.  50.  30. 221. 225. 217. 226.]
 [  1.  45.  15.  30. 224. 235. 233. 228.]
 [  1.  45.  50.  15. 218. 229. 240. 218.]]

Рівняння регресії
227.57 + 0.12*x1 + -0.12*x2 + -0.03*x3

Перевірка за критерієм Кохрена
Gp = 0.484
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [138.13765992990142, 1.0678780999550634, 1.2967091213740054, 0.15255401427929477]
Коефіцієнти [0.12, -0.12, -0.03] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [227.57]
[227.57, 227.57, 227.57, 227.57]

Перевірка адекватності за критерієм Фішера
Fp = 1.1256118303030282
F_t = 3.490294819497605
Математична модель адекватна експериментальним даним

Process finished with exit code 0
```

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.

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент з використанням лінійного рівняння регресії, скаладено 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). Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості $q = 0.05$.