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»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО-93
Прокопчук Д.І.
варіант – 23

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Варіант:

323	-5	15	-25	10	15	45

Код Програми:

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

variant = 323
rangeX = [(-5, 15), (-25, 10), (15, 45)]  # (x1,x2,x3)
avgMaxX = (15 + 10 + 45) / 3
avgMinX = (-5 - 25 + 15) / 3

maxY = 200 + int(avgMaxX)
minY = 200 + int(avgMinX)


def matrixPlan(n, m):
    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(minY, maxY)
    NormalX = 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]])
    xNorm = NormalX[:len(y)]

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

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

    return x, y, xNorm


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


def coef_of_regress(x, avgY, n):
    mx1 = sum(x[:, 1]) / n
    mx2 = sum(x[:, 2]) / n
```

```
    mx3 = sum(x[:, 3]) / n
    my = sum(avgY) / n

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

    a11 = sum([i ** 2 for i in x[:, 1]]) / 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
    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'({B[0]}) + ({B[1]})x1 + ({B[2]})x2 + ({B[3]})x3')

    return B


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


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


def kohren(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)


# оцінки коефіцієнтів
def bs(x, avgY, n):
    res = [sum(1 * y for y in avgY) / n]
    for i in range(3):  # 4 - кість факторів
        b = sum(j[0] * j[1] for j in zip(x[:, i], avgY)) / n
        res.append(b)
    return res


def test_student(x, y, avgY, n, m):
    Skv = dispersion(y, avgY, n, m)
    avg_skv = sum(Skv) / n

    s_Bs = (avg_skv / n / m) ** 0.5
    Bs = bs(x, avgY, n)
    ts = [abs(B) / s_Bs for B in Bs]
```

```
    return ts


def test_fisher(y, avgY, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - avgY[i]) ** 2 for i in
range(len(y))])
    S_kv = dispersion(y, avgY, n, m)
    avg_S_kv = sum(S_kv) / n

    return S_ad / avg_S_kv


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

    student = partial(scipy.stats.t.ppf, q=1 - 0.025)
    t_student = student(df=f3)

    G_kr = kohren(f1, f2)

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

    B = coef_of_regress(x, avgY, n)

    Gp = test_kohren(y, avgY, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'З ймовірністю {1 - q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити кількість дослідів")
        m += 1
        main(n, m)

    ts = test_student(x_norm[:, 1:], y, avgY, n, m)
    print('\nКритерій Стюдента =>\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(regress([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 = test_fisher(y, avgY, 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 variant == 323:
    n = 4  # кількість експериментів (рядків матриці планування)
    m = 4  # кількість вимірів y за однією й тією ж самою комбінації
факторів
    main(n, m)
```

Результати:

```
C:\Users\proko\AppData\Local\Programs\Python\Python39\python.exe B:/MOPE/lab3/main.py

Матриця планування =>
[[  1.  -5. -25.  15. 223. 222. 200. 196.]
 [  1.  -5.  10.  45. 196. 214. 213. 213.]
 [  1.  15. -25.  45. 216. 195. 222. 200.]
 [  1.  15.  10.  15. 209. 220. 213. 215.]]

Рівняння регресії =>
(214.17) + (0.08)x1 + (0.07)x2 + (-0.12)x3

Перевірка за критерієм Кохрена = >
Gp = 0.43787088653119943
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стюдента =>
 [90.30185349012015, 0.3486558049811589, 0.5095738688186168, 0.7777706418810467]
Коефіцієнти [0.08, 0.07, -0.12] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами - [214.17]
[214.17, 214.17, 214.17, 214.17]

Перевірка адекватності за критерієм Фішера =>
Fp = 1.1838346580351196
F_t = 3.490294819497605
Математична модель адекватна експериментальним даним

Process finished with exit code 0
```