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 №4

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)»

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

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Варіант:

323	-6	2	-9	5	-3	9

Код Програми:

```
import random
import sklearn.linear_model as lm
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *
import numpy as np

variant = 323


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


x_range = ((-6, 2), (-9, 5), (-3, 9))

x_aver_max = sum([x[1] for x in x_range]) / 3
x_aver_min = sum([x[0] for x in x_range]) / 3

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


# квадратна дисперсія
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(round(s, 3))
    return res


def plan_matrix5(n, m):
    print(f'\nГенеруємо матрицю планування для n = {n}, m = {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)

    if n > 14:
        no = n - 14
    else:
        no = 1
    x_norm = ccdesign(3, center=(0, no))
    x_norm = np.insert(x_norm, 0, 1, axis=1)

    for i in range(4, 11):
        x_norm = np.insert(x_norm, i, 0, axis=1)

    l = 1.215
```

```
    for i in range(len(x_norm)):
        for j in range(len(x_norm[i])):
            if x_norm[i][j] < -1 or x_norm[i][j] > 1:
                if x_norm[i][j] < 0:
                    x_norm[i][j] = -1
                else:
                    x_norm[i][j] = 1

    def add_sq_nums(x):
        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]
            x[i][8] = x[i][1] ** 2
            x[i][9] = x[i][2] ** 2
            x[i][10] = x[i][3] ** 2
        return x

    x_norm = add_sq_nums(x_norm)

    x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.int64)
    for i in range(8):
        for j in range(1, 4):
            if x_norm[i][j] == -1:
                x[i][j] = x_range[j - 1][0]
            else:
                x[i][j] = x_range[j - 1][1]

    for i in range(8, len(x)):
        for j in range(1, 3):
            x[i][j] = (x_range[j - 1][0] + x_range[j - 1][1]) / 2

    dx = [x_range[i][1] - (x_range[i][0] + x_range[i][1]) / 2 for i in
range(3)]

    x[8][1] = 1 * dx[0] + x[9][1]
    x[9][1] = -1 * dx[0] + x[9][1]
    x[10][2] = 1 * dx[1] + x[9][2]
    x[11][2] = -1 * dx[1] + x[9][2]
    x[12][3] = 1 * dx[2] + x[9][3]
    x[13][3] = -1 * dx[2] + x[9][3]

    x = add_sq_nums(x)

    print('\nX:\n', x)
    print('\nX нормоване:\n')
    for i in x_norm:
        print([round(x, 2) for x in i])
    print('\nY:\n', y)

    return x, y, x_norm


def find_coef(X, Y, norm=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(X, Y)
    B = skm.coef_

    if norm == 1:
        print('\nКоефіцієнти рівняння регресії з нормованими X:')
    else:
```

```
        print('\nКоефіцієнти рівняння регресії:')
    B = [round(i, 3) for i in B]
    print(B)
    print('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(X,
B))
    return B


def kriteriy_cochrana(y, y_aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp


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)


# оцінки коефіцієнтів
def bs(x, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]

    for i in range(len(x[0])):
        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

    # статистична оцінка дисперсії
    s_Bs = (s_kv_aver / n / m) ** 0.5  # статистична оцінка дисперсії
    Bs = bs(x, y_aver, n)
    ts = [round(abs(B) / s_Bs, 3) 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 check(X, Y, B, n, m):
    print('\n\tПеревірка рівняння:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05
```

```
    ### табличні значення
    student = partial(t.ppf, q=1 - q)
    t_student = student(df=f3)

    G_kr = cohren(f1, f2)
    ###

    y_aver = [round(sum(i) / len(i), 3) for i in Y]
    print('\nСереднє значення y:', y_aver)

    disp = s_kv(Y, y_aver, n, m)
    print('Дисперсія y:', disp)

    Gp = kriteriy_cochrana(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 кількість дослідів")
        m += 1
        main(n, m)

    ts = kriteriy_studenta(X[:, 1:], Y, y_aver, n, m)
    print('\nКритерій Стьюдента:\n', ts)
    res = [t for t in ts if t > t_student]
    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(regression([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

    F_p = kriteriy_fishera(Y, y_aver, y_new, n, m, d)

    fisher = partial(f.ppf, q=0.95)
    f_t = fisher(dfn=f4, dfd=f3)  # табличне знач
    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 variant == 323:
    n = 15
    m = 3
    X5, Y5, X5_norm = plan_matrix5(n, m)
```

```python
y5_aver = [round(sum(i) / len(i), 3) for i in Y5]
B5 = find_coef(X5, y5_aver)

check(X5_norm, Y5, B5, n, m)
```

Результати:

```
Генеруємо матрицю планування для n = 15, m = 3

X:
 [[   1  -10   -9   -8   90   80   72 -720  100   81   64]
 [   1    1   -9   -8   -9   -8   72   72    1   81   64]
 [   1  -10    7   -8  -70   80  -56  560  100   49   64]
 [   1    1    7   -8    7   -8  -56  -56    1   49   64]
 [   1  -10   -9    3   90  -30  -27  270  100   81    9]
 [   1    1   -9    3   -9    3  -27  -27    1   81    9]
 [   1  -10    7    3  -70  -30   21 -210  100   49    9]
 [   1    1    7    3    7    3   21   21    1   49    9]
 [   1    2   -1    1   -2    2   -1   -2    4    1    1]
 [   1  -10   -1    1   10  -10   -1   10  100    1    1]
 [   1   -4    8    1  -32   -4    8  -32   16   64    1]
 [   1   -4  -10    1   40   -4  -10   40   16  100    1]
 [   1   -4   -1    7    4  -28   -7   28   16    1   49]
 [   1   -4   -1   -5    4   20    5  -20   16    1   25]
 [   1   -4   -1    1    4   -4   -1    4   16    1    1]]

X нормоване:

[1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.22, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 1.48, 0.0, 0.0]
[1.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0, 0.0]
[1.0, 0.0, -1.22, 0.0, -0.0, 0.0, -0.0, -0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 0.0, -1.22, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
Y:
 [[193. 202. 199.]
 [192. 192. 192.]
 [196. 197. 199.]
 [196. 203. 201.]
 [202. 193. 203.]
 [194. 201. 201.]
 [195. 194. 193.]
 [201. 197. 200.]
 [203. 198. 191.]
 [193. 194. 196.]
 [195. 203. 201.]
 [197. 201. 195.]
 [201. 193. 200.]
 [196. 197. 191.]
 [202. 191. 199.]]

Коефіцієнти рівняння регресії:
[196.68, 0.044, 0.234, 0.295, 0.039, 0.035, -0.036, -0.001, -0.016, 0.027, 0.011]

Результат рівняння зі знайденими коефіцієнтами:
 [197.503 191.838 197.471 200.078 198.867 198.526 194.259 200.254 196.833
 194.805 198.51  198.654 198.045 195.537 196.395]

    Перевірка рівняння:

Середнє значення y: [198.0, 192.0, 197.333, 200.0, 199.333, 198.667, 194.0, 199.333, 197.333, 194.333, 199.667, 197.667, 198.0, 194.667, 197.333]
Дисперсія y: [14.0, 0.0, 1.556, 8.667, 20.222, 10.889, 0.667, 2.889, 24.222, 1.556, 11.556, 6.222, 12.667, 6.889, 21.556]

Перевірка за критерієм Кохрена
Gp = 0.168726229199376
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [427.559, 0.334, 0.034, 0.007, 2.12, 1.156, 1.735, 0.386, 311.794, 313.003, 312.807]

Коефіцієнти [0.044, 0.234, 0.295, 0.035, -0.001] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [196.68, 0.039, -0.036, -0.016, 0.027, 0.011]
[196.70499999999998, 196.627, 196.699, 196.777, 196.777, 196.699, 196.627, 196.70499999999998, 196.65638040000002, 196.65638040000002, 196.71985807500002, 196.71985807500002, 196.696238475, 196.696238475, 196.68]

Перевірка адекватності за критерієм Фішера
Fp = 2.7783947192383787
F_t = 2.2186969833835763
Математична модель не адекватна експериментальним даним
```