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 №5

З дисципліни «Методи оптимізації та планування»

Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

ВИКОНАЛА:
Студентка II курсу ФІОТ
Групи ІО-92
Бондар Х.В. - 9201

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

Мета: Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Варіант завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
201	-4	4	-10	4	-5	6

Лістинг програми:

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



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


x_range = ((-4, 4), (-10, 4), (-5, 6)) #значення за варіантом

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)    #значення функції відгуку
y_min = 200 + int(x_aver_min)


def s_kv(y, y_aver, n, m):        #кв. дисперсія
    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('\n-----------------------------------------------------------
---')
        print('\nКоефіцієнти рівняння регресії з нормованими X:')
```

```
    else:
        print('\n----------------------------------------------------------
---')
        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)


def bs(x, y_aver, n):   #оцінка коефіцієнтів
    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

    s_Bs = (s_kv_aver / n / m) ** 0.5
    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 perevirka(X, Y, B, n, m):
    print('\n----------------------------------------------------------------
')
    print('\nПеревірка рівняння:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05
```

```
    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('\nСр знач 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("Збільшіть 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, 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')
    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)
    print('\n----------------------------------------------------------------
')
    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('\n----------------------------------------------------------------
---')
        print('\n\tМатематична модель адекватна експериментальним даним')
    else:
        print('Математична модель не адекватна експериментальним даним')

def main(n, m):
    X5, Y5, X5_norm = plan_matrix5(n, m)

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

```
    B5 = find_coef(X5, y5_aver)

    perevirka(X5_norm, Y5, B5, n, m)


if __name__ == '__main__':
    main(15, 6)
```

Скріншоти виконання:

```
Генеруємо матрицю планування для n = 15, m = 6

X:
 [[   1   -4  -10   -5   40   20   50 -200   16  100   25]
 [   1    4  -10   -5  -40  -20   50  200   16  100   25]
 [   1   -4    4   -5  -16   20  -20   80   16   16   25]
 [   1    4    4   -5   16  -20  -20  -80   16   16   25]
 [   1   -4  -10    6   40  -24  -60  240   16  100   36]
 [   1    4  -10    6  -40   24  -60 -240   16  100   36]
 [   1   -4    4    6  -16  -24   24  -96   16   16   36]
 [   1    4    4    6   16   24   24   96   16   16   36]
 [   1    4   -3    1  -12    4   -3  -12   16    9    1]
 [   1   -4   -3    1   12   -4   -3   12   16    9    1]
 [   1    0    5    1    0    0    5    0    0   25    1]
 [   1    0  -11    1    0    0  -11    0    0  121    1]
 [   1    0   -3    7    0    0  -21    0    0    9   49]
 [   1    0   -3   -5    0    0   15    0    0    9   25]
 [   1    0   -3    1    0    0   -3    0    0    9    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, 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:
 [[194. 196. 195. 199. 204. 201.]
 [199. 200. 194. 196. 204. 200.]
 [204. 202. 201. 194. 202. 195.]
 [199. 198. 195. 198. 197. 203.]
 [197. 203. 200. 204. 201. 200.]
 [198. 197. 195. 198. 197. 204.]
 [194. 202. 194. 203. 199. 195.]
 [197. 199. 197. 194. 203. 198.]
 [197. 199. 199. 203. 198. 201.]
 [195. 194. 202. 198. 202. 194.]
 [194. 204. 194. 204. 201. 200.]
 [201. 203. 195. 196. 202. 194.]
 [204. 197. 195. 199. 196. 202.]
 [198. 197. 196. 203. 195. 194.]
 [204. 199. 194. 199. 203. 200.]]

----------------------------------------------------------------

Коефіцієнти рівняння регресії:
[198.796, -0.018, 0.032, 0.004, 0.002, 0.003, -0.013, 0.004, -0.002, 0.006, -0.013]

Результат рівняння зі знайденими коефіцієнтами:
 [197.461 198.637 199.323 198.483 200.42  198.34  197.816 198.648 198.62
 198.884 199.032 199.304 198.418 198.214 198.784]
```

```
----------------------------------------------------------------

Перевірка рівняння:

Ср знач y: [198.167, 198.833, 199.667, 198.333, 200.833, 198.167, 197.833, 198.0, 199.5, 197.5, 199.5, 198.5, 198.833, 197.167, 199.833]
Дисперсія y: [12.472, 10.139, 14.222, 5.889, 5.139, 7.806, 13.806, 7.333, 3.917, 11.917, 17.25, 12.917, 10.472, 8.472, 10.472]

Перевірка за критерієм Кохрена
Gp = 0.11332058887290355
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [591.764, 1.111, 0.671, 0.435, 0.165, 0.364, 0.827, 0.96, 431.99, 432.283, 431.697]

Коефіцієнти [-0.018, 0.032, 0.004, 0.002, 0.003, 0.004] статистично незначущі, виключаємо їх з рівняння.

Значення y з коефіцієнтами
[198.78699999999998, 198.78699999999998, 198.78699999999998, 198.78699999999998, 198.78699999999998, 198.78699999999998, 198.78699999999998, 198
 .79304754999998, 198.79304754999998, 198.80485735, 198.80485735, 198.776809075, 198.776809075, 198.796]

----------------------------------------------------------------

Перевірка адекватності за критерієм Фішера
Fp = 0.7463012546152221
F_t = 1.9187589455788492

----------------------------------------------------------------

    Математична модель адекватна експериментальним даним
```

Висновок:

В даній лабораторній роботі я провела трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайшла рівняння регресії, яке буде адекватним для опису об'єкту. Було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера) для кожної форми рівняння регресії . При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння регресії було застосовано рівняння регресії з квадратичними членами. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05