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 наукових досліджень»

Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-91
Микитенко В.О.
Залікова
№ ІВ-9119

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
118	-3	10	-8	2	-6	1

Лістинг програми:

```
import random
from sklearn.linear_model import LinearRegression
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *
from prettytable import PrettyTable
from time import time

x_range = ((-3, 10), (-8, 2), (-6, 1))
y_min = 200 + int(sum([x[0] for x in x_range]) / 3)
y_max = 200 + int(sum([x[1] for x in x_range]) / 3)


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 = ccdesign(3, center=(0, 1))
    permutation = [2, 1, 0]
    idx = np.empty_like(permutation)
    idx[permutation] = np.arange(len(permutation))
    x_norm = x_norm[:, idx]

    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
    x_norm[8][1] = -l
    x_norm[9][1] = l
    x_norm[8][3] = 0
    x_norm[9][3] = 0
    x_norm[10][2] = -l
    x_norm[11][2] = l
    x_norm[12][3] = -l
    x_norm[13][3] = l
    x_norm[12][1] = 0
    x_norm[13][1] = 0

    def fill_tabs(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] = round(x[i][1] ** 2, 3)
            x[i][9] = round(x[i][2] ** 2, 3)
            x[i][10] = round(x[i][3] ** 2, 3)
        return x

    x_norm = fill_tabs(x_norm)
```

```
    x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.float64)
    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, 4):
            x[i][j] = (x_range[j - 1][0] + x_range[j - 1][1]) / 2
    dx = [(x_range[i][1] - x_range[i][0]) / 2 for i in range(3)]
    x[8][1] = -l * dx[0] + x[9][1]
    x[9][1] = l * dx[0] + x[9][1]
    x[10][2] = -l * dx[1] + x[9][2]
    x[11][2] = l * dx[1] + x[9][2]
    x[12][3] = -l * dx[2] + x[9][3]
    x[13][3] = l * dx[2] + x[9][3]
    x = np.around(fill_tabs(x), 3)

    x_table = PrettyTable()
    for i in range(n):
        x_table.add_row([*x[i]])
    print('Матриця планування:')
    print(x_table)

    x_norm_table = PrettyTable()
    for i in range(n):
        x_norm_table.add_row([*x_norm[i]])
    print('Нормована матриця планування:')
    print(x_norm_table)

    return x, y, x_norm


def coef_finding(x, y, norm=False):
    skm = LinearRegression(fit_intercept=False)
    skm.fit(x, y)
    b = skm.coef_
    if norm == 1:
        print('\nКоефіцієнти з нормаваними значеннями:')
    else:
        print('\nКоефіцієнти:')
    b = [round(i, 3) for i in b]
    print(b)
    print(y)
    print('\nЗначення рівняння зі знайденими
коефіцієнтами:\n{}'.format(np.dot(x, b)))
    return b


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


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 checkFull(x, y, b, n, m):
    print('\nСтатистичні перевірки:')
    f1 = m - 1
    f2 = n
```

```
    f3 = f1 * f2
    q = 0.05

    kohren = {2: 3346, 3: 2758, 4: 2419, 5: 2159, 6: 2034, 7: 1911, 8: 1815, 9:
1736, 10: 1671}
    g_kr = kohren.get(f1) / 10000

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

    y_aver = [round(sum(i) / len(i), 3) for i in y]
    print('\nСередні значення Y:', y_aver)
    disp = s_kv(y, y_aver, n, m)
    print('Дисперсія Y:', disp)
    ck = time()

    f1 = m - 1
    f2 = n
    q = 0.05
    skv = s_kv(y, y_aver, n, m)
    gp = max(skv) / sum(skv)

    ck = time() - ck
    print(f'\nКритерій Кохрена:\ngp = {gp}')
    if gp < g_kr:
        print('Дисперсія однорідна')
    else:
        print("Дисперсія неонорідна")
        m += 1
        start(n, m)
    cs = time()
    skv = s_kv(y, y_aver, n, m)
    skv_aver = sum(skv) / n
    sbs_tmp = (skv_aver / n / m) ** 0.5

    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

    bs_tmp = bs(x[:, 1:], y_aver, n)
    ts = [round(abs(b) / sbs_tmp, 3) for b in bs_tmp]

    cs = time() - cs
    print('\nКритерій Стьюдента:\n{}:'.format(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 {} незначимі'.format([round(i, 3) for i in b if i not
in final_k]))
    y_new = []
    for j in range(n):
        y_new.append(round(regression([x[j][i] for i in range(len(ts)) if ts[i]
in res], final_k), 3))
    print('Значення функції відгуку зі значимими коефіцієнтами {}:
'.format(final_k))
    print(y_new)
    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        return
    f4 = n - d
    cr = time()
    S_ad = m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in
range(len(y))])
    skv = s_kv(y, y_aver, n, m)
    skv_aver = sum(skv) / n
```

```
    f_p = S_ad / skv_aver

    fisher = partial(f.ppf, q=1 - q)
    f_t = fisher(dfn=f4, dfd=f3)
    cr = time() - cr
    print('\nКритерій Фішера:')
    print('fp =', f_p)
    print('ft =', f_t)
    if f_p < f_t:
        print('Математична модель адекватна')
    else:
        print('Математична модель неадекватна')
    print("час проведення перевірки за критерієм Кохрена ", ck * 1000, " мс")
    print("час проведення перевірки за критерієм Стьюдента ", cs * 1000, " мс")
    print("час проведення перевірки за критерієм Фішера ", cr * 1000, " мс")


def start(n, m):
    x, y, x_norm = plan_matrix(n, m)
    y_aver = [round(sum(i) / len(i), 3) for i in y]
    b = coef_finding(x, y_aver)
    checkFull(x_norm, y, b, n, m)


start(15, 3)
```

Результати виконання:

Матриця планування:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10	Field 11
1.0	-3.0	-8.0	-6.0	24.0	18.0	48.0	-144.0	9.0	64.0	36.0
1.0	-3.0	-8.0	1.0	24.0	-3.0	-8.0	24.0	9.0	64.0	1.0
1.0	-3.0	2.0	-6.0	-6.0	18.0	-12.0	36.0	9.0	4.0	36.0
1.0	-3.0	2.0	1.0	-6.0	-3.0	2.0	-6.0	9.0	4.0	1.0
1.0	10.0	-8.0	-6.0	-80.0	-60.0	48.0	480.0	100.0	64.0	36.0
1.0	10.0	-8.0	1.0	-80.0	10.0	-8.0	-80.0	100.0	64.0	1.0
1.0	10.0	2.0	-6.0	20.0	-60.0	-12.0	-120.0	100.0	4.0	36.0
1.0	10.0	2.0	1.0	20.0	10.0	2.0	20.0	100.0	4.0	1.0
1.0	-4.398	-3.0	-2.5	13.193	10.994	7.5	-32.981	19.338	9.0	6.25
1.0	11.398	-3.0	-2.5	-34.192	-28.494	7.5	85.481	129.903	9.0	6.25
1.0	3.5	-9.075	-2.5	-31.762	-8.75	22.688	79.406	12.25	82.356	6.25
1.0	3.5	3.075	-2.5	10.763	-8.75	-7.688	-26.906	12.25	9.456	6.25
1.0	3.5	-3.0	-6.752	-10.5	-23.634	20.258	70.901	12.25	9.0	45.596
1.0	3.5	-3.0	1.753	-10.5	6.134	-5.258	-18.401	12.25	9.0	3.071
1.0	3.5	-3.0	-2.5	-10.5	-8.75	7.5	26.25	12.25	9.0	6.25

Нормована матриця планування:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10	Field 11
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	-1.215	0.0	0.0	-0.0	-0.0	0.0	-0.0	1.476	0.0	0.0
1.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476	0.0	0.0
1.0	0.0	-1.215	0.0	-0.0	0.0	-0.0	-0.0	0.0	1.476	0.0
1.0	0.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476	0.0
1.0	0.0	0.0	-1.215	0.0	-0.0	-0.0	-0.0	0.0	0.0	1.476
1.0	0.0	0.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
Коефіцієнти:
[199.996, 0.178, 0.009, -0.115, -0.016, 0.001, -0.018, -0.004, -0.035, 0.027, -0.028]
[200.0, 200.667, 199.0, 199.333, 198.0, 202.0, 198.333, 198.0, 198.333, 197.333, 201.333, 200.667, 199.333, 199.667, 201.667]

Значення рівняння зі знайденими коефіцієнтами:
[199.831    200.321    199.141    199.211    198.05     201.543
 198.4      197.833    198.661656 197.848393 202.218121 200.650787
 199.40216  200.461049 200.438   ]

Статистичні перевірки:

Середні значення Y: [200.0, 200.667, 199.0, 199.333, 198.0, 202.0, 198.333, 198.0, 198.333, 197.333, 201.333, 200.667, 199.333, 199.667, 201.667]
Дисперсія Y: [8.667, 8.222, 4.667, 6.222, 2.0, 4.667, 16.222, 4.667, 5.556, 3.556, 9.556, 16.222, 2.889, 6.889, 2.889]

Критерій Кохрена:
gp = 0.15766199181658264
Дисперсія однорідна

Критерій Стьюдента:
[511.182, 0.663, 1.163, 0.866, 0.228, 0.455, 0.797, 0.683, 372.131, 373.728, 372.972]:
Коефіцієнти [0.178, 0.009, -0.115, -0.016, 0.001, -0.018, -0.004] незначимі
Значення функції відгуку зі значимими коефіцієнтами [199.996, -0.035, 0.027, -0.028]:
[199.96, 199.96, 199.96, 199.96, 199.96, 199.96, 199.96, 199.96, 199.944, 199.944, 200.036, 200.036, 199.955, 199.955, 199.996]

Критерій Фішера:
fp = 1.2350611635791093
ft = 2.125558760875511
Математична модель адекватна
час проведення перевірки за критерієм Кохрена  2.7124881744384766  мс
час проведення перевірки за критерієм Стьюдента  1.0421276092529297  мс
час проведення перевірки за критерієм Фішера  0.0  мс
```

Висновок:

В даній лабораторній роботі я провів трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайшов рівняння регресії, яке буде адекватним для опису об'єкту.