

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 II курсу ФІОТ
Групи ІВ-91
Богомол В.Ю. ІВ-9101

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
101	-1	4	-3	6	-1	9

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 = ((-1, 4), (-3, 6), (-1, 9))
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] = -1
    x_norm[9][1] = 1
    x_norm[8][3] = 0
    x_norm[9][3] = 0
    x_norm[10][2] = -1
    x_norm[11][2] = 1
    x_norm[12][3] = -1
    x_norm[13][3] = 1
    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] = -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 = 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 нормованими 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 неоднорідна")
        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 Стюдента:\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 Стюдента ", 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)

```

```

main x
Значення рівняння зі знайденими коефіцієнтами:
[203.4      203.07    203.076    204.096    202.62    202.34
 200.631    201.251    204.017865 201.95225    203.342595 202.620887
 202.837847 203.150709 203.40025 ]

Статистичні перевірки:

Середні значення Y: [203.667, 203.0, 203.333, 204.0, 202.667, 202.0, 200.667, 201.0, 203.667, 202.333, 203.333, 202.667, 202.333, 203.667, 203.333]
Дисперсія Y: [4.222, 0.667, 5.556, 2.667, 0.222, 4.667, 0.222, 2.0, 6.889, 4.222, 0.889, 1.556, 4.222, 10.889, 0.222]

Критерій Кохрена:
gr = 0.22171770646685127
Дисперсія однорідна

Критерій Стюдента:
[751.759, 2.295, 0.777, 0.318, 0.906, 0.083, 0.577, 0.083, 548.58, 548.58, 548.58]:

Коефіцієнти [-0.017, 0.098, -0.038, -0.002, 0.014, -0.001] незначимі
Значення функції відгуку зі значимими коефіцієнтами [203.566, -0.134, -0.045, -0.014, -0.011]:
[203.63, 203.63, 203.63, 203.63, 203.362, 203.362, 203.362, 203.362, 203.662, 203.337, 203.545, 203.545, 203.55, 203.55, 203.566]

Критерій Фішера:
fp = 1.75708653689525
ft = 2.164579917125473
Математична модель адекватна
час проведення перевірки за критерієм Кохрена 0.9970664978027344 мс
час проведення перевірки за критерієм Стюдента 0.0 мс
час проведення перевірки за критерієм Фішера 0.0 мс

Process finished with exit code 0
PyCharm 202
Update...

```

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.