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 (центральний ортогональний композиційний план)

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІО-91
Павлюк Владислав Володимирович
Номер заліковки: 9120
Номер у списку: 17

ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ 2021 р.

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

Завдання

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$

$$y_{i\min} = 200 + x_{cp\min}$$

$$\text{где } x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3},\ x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Виконання роботи

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
117	-7	10	-4	8	-5	4

Код програми

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


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


def skv(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(round(s, 3))
    return res
```

```
def planMatrix5(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(minY, maxY)

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

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

    l = 1.215

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

    def addSqNums(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

    normX = addSqNums(normX)

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

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

    dx = [rangeX[i][1] - (rangeX[i][0] + rangeX[i][1]) / 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 = addSqNums(x)

    print('\nX:\n', x)
    print('\nX нормоване:\n')
    for i in normX:
        print([round(x, 2) for x in i])
```

```
    print('\nY:\n', y)

    return x, y, normX


def findCoef(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 kriteriaCochrana(y, avgY, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    Skv = skv(y, avgY, n, m)
    Gp = max(Skv) / sum(Skv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp


def cohren(f1, f2, q=0.05):
    q1 = q / f1
    fisherValue = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fisherValue / (fisherValue + f1 - 1)


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

    for i in range(len(x[0])):
        b = sum(j[0] * j[1] for j in zip(x[:, i], avgY)) / n
        res.append(b)
    return res


def studentKriteria(x, y, avgY, n, m):
    Skv = skv(y, avgY, n, m)
    avgSkv = sum(Skv) / n

    s_Bs = (avgSkv / n / m) ** 0.5
    Bs = bs(x, avgY, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts


def kriteriaFisher(y, avgY, newY, n, m, d):
    Sad = m / (n - d) * sum([(newY[i] - avgY[i]) ** 2 for i in range(len(y))])
    Skv = skv(y, avgY, n, m)
    avgSkv = sum(Skv) / n

    return Sad / avgSkv


def check(X, Y, B, n, m):
    print('\n\tПеревірка рівняння:')
    f1 = m - 1
    f2 = n
```

```
    f3 = f1 * f2
    q = 0.05

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

    Gkr = cohren(f1, f2)

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

    disp = skv(Y, avgY, n, m)
    print('Дисперсія y:', disp)

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

    ts = studentKriteria(X[:, 1:], Y, avgY, n, m)
    print('\nКритерій Стьюдента:\n', ts)
    res = [t for t in ts if t > Tstudent]
    finalK = [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 finalK]))

    newY = []
    for j in range(n):
        newY.append(regression([X[j][i] for i in range(len(ts)) if ts[i] in res], finalK))

    print(f'\nЗначення "y" з коефіцієнтами {finalK}')
    print(newY)

    timeStart = process_time()

    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        print('')
        return
    f4 = n - d

    Fp = kriteriaFisher(Y, avgY, newY, n, m, d)
    fisher = partial(f.ppf, q=0.95)
    Ft = fisher(dfn=f4, dfd=f3)
    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', Fp)
    print('Ft =', Ft)
    if Fp < Ft:
        print('Математична модель адекватна експериментальним даним')
    else:
        print('Математична модель не адекватна експериментальним даним')

    timeEnd = process_time()
    print(f"Час перевірки адекватності системи за критерієм Фішера: {timeEnd - timeStart}")


def main(n, m):
    X5, Y5, normX5 = planMatrix5(n, m)

    averY5 = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = findCoef(X5, averY5)

    check(normX5, Y5, B5, n, m)
```

```python
if __name__ == '__main__':
    # Варіант - 117
    rangeX = ((-7, 10), (-4, 8), (-5, 4))

    avgMaxX = sum([x[1] for x in rangeX]) / 3
    avgMinX = sum([x[0] for x in rangeX]) / 3

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

    n = 15
    m = 3
    main(n, m
```

Результат роботи програми

C:\Anac\envs\Lab_1\python.exe "C:/Users/Владислав/Desktop/Лаби/Лаб 5/main.py"

Генеруємо матрицю планування для n = 15, m = 3

X:
```
[[   1   -7   -4   -5   28   35   20 -140   49   16   25]
 [   1   10   -4   -5  -40  -50   20  200  100   16   25]
 [   1   -7    8   -5  -56   35  -40  280   49   64   25]
 [   1   10    8   -5   80  -50  -40 -400  100   64   25]
 [   1   -7   -4    4   28  -28  -16  112   49   16   16]
 [   1   10   -4    4  -40   40  -16 -160  100   16   16]
 [   1   -7    8    4  -56  -28   32 -224   49   64   16]
 [   1   10    8    4   80   40   32  320  100   64   16]
 [   1   11    2    1   22   11    2   22  121    4    1]
 [   1   -9    2    1  -18   -9    2  -18   81    4    1]
 [   1    1    9    1    9    1    9    9    1   81    1]
 [   1    1   -5    1   -5    1   -5   -5    1   25    1]
 [   1    1    2    6    2    6   12   12    1    4   36]
 [   1    1    2   -4    2   -4   -8   -8    1    4   16]
 [   1    1    2    1    2    1    2    2    1    4    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:
```
[[204. 202. 197.]
 [198. 196. 195.]
 [206. 195. 207.]
 [204. 195. 204.]
 [198. 203. 202.]
 [203. 200. 196.]
 [205. 196. 200.]
 [196. 206. 200.]
 [196. 206. 196.]
 [196. 207. 195.]
 [203. 201. 201.]
 [196. 199. 198.]
 [203. 201. 199.]
 [202. 205. 200.]
 [203. 205. 200.]]
```

Коефіцієнти рівняння регресії:
[201.027, -0.06, 0.239, -0.005, 0.011, 0.021, -0.023, -0.001, -0.014, -0.02, 0.023]

Результат рівняння зі знайденими коефіцієнтами:
[200.808 196.201 202.752 201.409 199.809 199.027 200.025 200.671 199.494
200.434 201.406 199.362 202.009 201.869 201.364]

Перевірка рівняння:

Середнє значення y: [201.0, 196.333, 202.667, 201.0, 201.0, 199.667, 200.333, 200.667, 199.333, 199.333, 201.667, 197.667, 201.0, 202.333, 202.667]
Дисперсія y: [8.667, 1.556, 29.556, 18.0, 4.667, 8.222, 13.556, 16.889, 22.222, 29.556, 0.889, 1.556, 2.667, 4.222, 4.222]

Перевірка за критерієм Кохрена
Gp = 0.17757003730917345
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
[403.653, 0.984, 0.243, 0.307, 0.627, 0.716, 0.806, 0.179, 294.173, 294.305, 295.098]

Коефіцієнти [-0.06, 0.239, -0.005, 0.011, 0.021, -0.023, -0.001] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [201.027, -0.014, -0.02, 0.023]
[201.01599999999996, 201.01599999999996, 201.01599999999996, 201.01599999999996, 201.01599999999996, 201.01599999999996, 201.01599999999996, 201.01599999999996, 201.00633284999998, 201.00633284999998, 200.99747549999998, 200.99747549999998, 201.06095317499998, 201.06095317499998, 201.027]

Перевірка адекватності за критерієм Фішера
Fp = 1.1924430452735435
Ft = 2.125558760875511
Математична модель адекватна експериментальним даним
Час перевірки адекватності системи за критерієм Фішера: 0.0

Process finished with exit code 0

Висновок

У ході виконання лабораторної роботи я провів трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план.
Знайшов рівняння регресії, яке буде адекватним для опису об'єкту.
Кінцева мета досягнута.