

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»
За темою:
«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З
ВИКОРИСТАННЯМ
ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:
Студент II курсу ФІОТ
Групи ІВ-91
Гришин О.С.
Номер у списку - 07

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

Мета: Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)

Завдання:

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.

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

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

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

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

107	-9	7	-4	7	-10	5
-----	----	---	----	---	-----	---

Програмний код

Main.pt

```
from lab05.src.Experiment import Experiment

if __name__ == '__main__':
    experiment = Experiment((-9, 7), (-4, 7), (-10, 5))

    experiment.run_experiment(15, 3)
```

Experiment.py

```
import random
import sklearn.linear_model as lm
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *
from numpy import average
from lab05.src.Criterium import Criteria
import pandas as pd
from tabulate import tabulate

class Experiment:

    def __init__(self, x1, x2, x3):

        self.n, self.m = 15, 6

        self.x, self.y, self.x_norm = None, None, None
        self.y_average = None

        self.b = None

        self.x_range = (x1, x2, x3)
```

```

        self.x_aver_max = average([x[1] for x in self.x_range])
        self.x_aver_min = average([x[0] for x in self.x_range])

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

    @staticmethod
    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

    def count_y_average(self):
        self.y_average = [round(sum(i) / len(i), 3) for i in self.y]

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

    def count_b_coefficient(self):
        # X5, y5_aver
        skm = lm.LinearRegression(fit_intercept=False)
        skm.fit(self.x, self.y_average)
        self.b = skm.coef_

        print('\nКоефіцієнти рівняння регресії:')
        self.b = [round(i, 3) for i in self.b]
        print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1*x2 + {}*x1*x3 + {}*x2*x3 + {}*x1*x2*x3 + {}*x1^2 + {}*x2^2 + {}*x3^2\n".format(*self.b))
        print('\nРезультат рівняння зі знайденими коефіцієнтами:\n',
              np.dot(self.x, self.b))

    def run_check(self):
        criteria = Criteria(self.x, self.y, self.n, self.m)

        print('\n\tПеревірка рівняння:')
        f1 = self.m - 1
        f2 = self.n
        f3 = f1 * f2
        q = 0.05

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

        G_kr = criteria.cohren()

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

        disp = criteria.s_kv(y_aver)
        print('Дисперсія y:', disp)

        Gp = criteria.criteria_cochrana(y_aver)
        print(f'Gp = {Gp}')
        if Gp < G_kr:
            print(f'З ймовірністю {1 - q} дисперсії однорідні.')
        else:
            print("Необхідно збільшити кількість дослідів")

```

```

        self.m += 1
        new_exp = Experiment(*self.x_range)
        new_exp.run_experiment(self.n, self.m)

        ts = criteria.criteria_studenta(self.x_norm[:, 1:], y_aver)
        print('\nКритерій Стьюдента:\n', ts)
        res = [t for t in ts if t > t_student]
        final_k = [self.b[i] for i in range(len(ts)) if ts[i] in res]
        print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх
з порівняння.'.format(
            [round(i, 3) for i in self.b if i not in final_k]))

        y_new = []
        for j in range(self.n):
            y_new.append(self.regression([self.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 >= self.n:
            print('\nF4 <= 0')

        f4 = self.n - d

        F_p = criteria.criteria_fishera(y_aver, y_new, d)

        fisher = partial(f.ppf, q=0.95)
        f_t = fisher(dfn=f4, dfd=f3)
        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')

    def fill_y(self):
        self.y = np.zeros(shape=(self.n, self.m))
        for i in range(self.n):
            for j in range(self.m):
                self.y[i][j] = random.randint(self.y_min, self.y_max)

    def make_matrix(self):
        self.fill_y()
        if self.n > 14:
            no = self.n - 14
        else:
            no = 1

        self.x_norm = ccdesign(3, center=(0, no))
        self.x_norm = np.insert(self.x_norm, 0, 1, axis=1)

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

        l = 1.215

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

```

```

        else:
            self.x_norm[i][j] = 1

    self.x_norm = self.add_sq_nums(self.x_norm)

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

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

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

    self.x[8][1] = 1 * dx[0] + self.x[9][1]
    self.x[9][1] = -1 * dx[0] + self.x[9][1]
    self.x[10][2] = 1 * dx[1] + self.x[9][2]
    self.x[11][2] = -1 * dx[1] + self.x[9][2]
    self.x[12][3] = 1 * dx[2] + self.x[9][3]
    self.x[13][3] = -1 * dx[2] + self.x[9][3]
    self.x = self.add_sq_nums(self.x)

    show_arr = pd.DataFrame(self.x)
    print('\nX:\n', tabulate(show_arr, headers='keys', tablefmt='psql'))

    show_arr = pd.DataFrame(self.x_norm)
    print('\nX нормоване:\n', tabulate(show_arr.round(0), headers='keys',
tablefmt='psql'))

    show_arr = pd.DataFrame(self.y)
    print('\nY:\n', tabulate(show_arr, headers='keys', tablefmt='psql'))

def run_experiment(self, n=None, m=None):
    if n is None and m is None:
        pass
    else:
        self.n = n
        self.m = m

    self.make_matrix()
    self.count_y_average()

    self.count_b_coeficient()
    self.run_check()

```

Criteria.py

```

from scipy.stats import f

class Criteria:
    def __init__(self, x, y, n, m):
        self.x = x
        self.y = y

```

```

        self.n = n
        self.m = m
        self.f1 = self.m - 1
        self.f2 = self.n
        self.q = 0.05
        self.q1 = self.q / self.f1

    def s_kv(self, y_aver):
        res = []
        for i in range(self.n):
            s = sum([(y_aver[i] - self.y[i][j]) ** 2 for j in range(self.m)])
/ self.m
            res.append(round(s, 3))
        return res

    def criteria_cochrana(self, y_aver):
        S_kv = self.s_kv(y_aver)
        Gp = max(S_kv) / sum(S_kv)
        print('\nПеревірка за критерієм Кохрена')
        return Gp

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

    def bs(self, x, y_aver):
        res = [sum(y_aver) / self.n]

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

        return res

    def criteria_studenta(self, x, y_aver):
        S_kv = self.s_kv(y_aver)
        s_kv_aver = sum(S_kv) / self.n

        s_Bs = (s_kv_aver / self.n / self.m) ** 0.5
        Bs = self.bs(x, y_aver)
        ts = [round(abs(B) / s_Bs, 3) for B in Bs]

        return ts

    def criteria_fishera(self, y_aver, y_new, d):
        S_ad = self.m / (self.n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i
in range(len(self.y))])
        S_kv = self.s_kv(y_aver)
        S_kv_aver = sum(S_kv) / self.n

        return S_ad / S_kv_aver

```

Результати роботи програми

$X:$

X нормоване:

Y:

	0	1	2
0	193	195	193
1	203	195	198
2	201	205	193
3	202	199	193
4	198	204	193
5	197	206	203
6	205	206	198
7	201	194	201
8	193	200	203
9	200	195	198
10	194	194	204
11	206	197	199
12	195	199	199
13	204	195	197
14	201	197	194

Коефіцієнти рівняння регресії:

$y = 198.022 + 0.104 \cdot x_1 + -0.14 \cdot x_2 + 0.118 \cdot x_3 + -0.043 \cdot x_1 \cdot x_2 + -0.007 \cdot x_1 \cdot x_3 + -0.016 \cdot x_2 \cdot x_3 + b - 0.001 \cdot x_1 \cdot x_2 \cdot x_3 + 0.006 \cdot x_1^2 + 0.045 \cdot x_2^2 + -0.008 \cdot x_3^2$

Результат рівняння зі знайденими коефіцієнтами:

[194.414 199.118 199.386 198.282 198.149 202.133 201.966 197.502 198.829
198.091 199.462 199.726 198.172 196.48 197.974]

Перевірка рівняння:

Середнє значення y: [193.667, 198.667, 199.667, 198.0, 198.333, 202.0, 203.0, 198.667, 198.667, 197.667, 197.333, 200.667, 197.667, 198.667, 197.333]
Дисперсія y: [0.889, 10.889, 24.889, 14.0, 20.222, 14.0, 12.667, 10.889, 17.556, 4.222, 22.222, 14.889, 3.556, 14.889, 8.222]

Перевірка за критерієм Кохрена

$G_p = 0.12829315312807663$

З ймовірністю 0.95 дисперсії однорідні.

Критерій Стюдента:

[370.574, 0.181, 1.333, 1.643, 1.824, 0.497, 0.497, 0.166, 270.728, 271.034, 270.728]

Коефіцієнти [0.104, -0.14, 0.118, -0.007, -0.016, -0.001] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [198.022, -0.043, 0.006, 0.045, -0.008]

[196.87999999999997, 199.44, 202.62199999999999, 197.614, 197.48, 200.04000000000002,

Перевірка адекватності за критерієм Фішера

$F_p = 0.9750694017041039$

$F_t = 2.164579917125473$

Математична модель адекватна експериментальним даним