Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни **«Методи оптимізації та планування»**

Виконав:
студент групи ІВ-91
Вігор Дмитро
Залікова книжка: 9106

Перевірив:
ас. Регіда П.Г.

Київ – 2021

ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

Мета: провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Завдання на лабораторну роботу:

1. Скласти матрицю планування для дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку У. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

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

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

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

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.
3. Провести 3 статистичні перевірки.
4. Написати комп'ютерну програму, яка усе це виконує.

Варіант завдання:

106	10	40	25	45	40	45

Xcp max = 130/3 ≈ 43

Xcp min = 75/3 = 25

Ymax = 243

Ymin = 225

Роздруківка тексту програми:

```
from random import *
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial


class FractionalExperiment:
    """Проведення дробового трьохфакторного експерименту"""
```

```
    def __init__(self, n, m):
        self.n = n
        self.m = m
        self.x_min = (10 + 25 + 40) / 3
        self.x_max = (40 + 45 + 45) / 3
        self.y_max = round(200 + self.x_max)
        self.y_min = round(200 + self.x_min)
        self.x_norm = [[1, -1, -1, -1],
                       [1, -1, 1, 1],
                       [1, 1, -1, 1],
                       [1, 1, 1, -1],
                       [1, -1, -1, 1],
                       [1, -1, 1, -1],
                       [1, 1, -1, -1],
                       [1, 1, 1, 1]]
        self.x_range = [(10, 40), (25, 45), (40, 45)]
        self.y = np.zeros(shape=(self.n, self.m))
        self.y_new = []
        for i in range(self.n):
            for j in range(self.m):
                self.y[i][j] = randint(self.y_min, self.y_max)
        self.y_av = [round(sum(i) / len(i), 2) for i in self.y]
        self.x_norm = self.x_norm[:len(self.y)]
        self.x = np.ones(shape=(len(self.x_norm), len(self.x_norm[0])))
        for i in range(len(self.x_norm)):
            for j in range(1, len(self.x_norm[i])):
                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]
        self.f1 = m - 1
        self.f2 = n
        self.f3 = self.f1 * self.f2
        self.q = 0.05

    def regression(self, x, b):
        """Підстановка коефіцієнтів у рівняння регресії"""
        y = sum([x[i] * b[i] for i in range(len(x))])
        return y

    def count_koefs(self):
        """Розрахунок коефіцієнтів рівняння регресії"""
        mx1 = sum(self.x[:, 1]) / self.n
        mx2 = sum(self.x[:, 2]) / self.n
        mx3 = sum(self.x[:, 3]) / self.n
        my = sum(self.y_av) / self.n
        a12 = sum([self.x[i][1] * self.x[i][2] for i in range(len(self.x))]) / self.n
        a13 = sum([self.x[i][1] * self.x[i][3] for i in range(len(self.x))]) / self.n
        a23 = sum([self.x[i][2] * self.x[i][3] for i in range(len(self.x))]) / self.n
        a11 = sum([i ** 2 for i in self.x[:, 1]]) / self.n
        a22 = sum([i ** 2 for i in self.x[:, 2]]) / self.n
        a33 = sum([i ** 2 for i in self.x[:, 3]]) / self.n
        a1 = sum([self.y_av[i] * self.x[i][1] for i in range(len(self.x))]) / self.n
        a2 = sum([self.y_av[i] * self.x[i][2] for i in range(len(self.x))]) / self.n
        a3 = sum([self.y_av[i] * self.x[i][3] for i in range(len(self.x))]) / self.n

        X = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a12, a22, a23], [mx3, a13,
a23, a33]]
        Y = [my, a1, a2, a3]
        B = [round(i, 2) for i in solve(X, Y)]
        print('\nРівняння регресії')
```

```
        print(f'y = {B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

        return B

    def dispersion(self):
        """Розрахунок дисперсії"""
        res = []
        for i in range(self.n):
            s = sum([(self.y_av[i] - self.y[i][j]) ** 2 for j in range(self.m)]) /
self.m
            res.append(s)
        return res

    def kohren(self):
        """Перевірка однорідності дисперсій за критерієм Кохрена"""
        q1 = self.q / self.f1
        fisher_value = f.ppf(q=1 - q1, dfn=self.f2, dfd=(self.f1 - 1) * self.f2)
        G_cr = fisher_value / (fisher_value + self.f1 - 1)
        s = self.dispersion()
        Gp = max(s) / sum(s)
        return Gp, G_cr

    def student(self):
        """Перевірка знащущості коефіцієнтів за критерієм Стьюдента"""

        def bs():
            res = [sum(1 * y for y in self.y_av) / self.n]
            for i in range(3):  # 4 - ксть факторів
                b = sum(j[0] * j[1] for j in zip(self.x[:, i], self.y_av)) / self.n
                res.append(b)
            return res

        S_kv = self.dispersion()
        s_kv_aver = sum(S_kv) / self.n

        # статистична оцінка дисперсії
        s_Bs = (s_kv_aver / self.n / self.m) ** 0.5
        Bs = bs()
        ts = [abs(B) / s_Bs for B in Bs]
        return ts

    def fisher(self, d):
        """Перевірка адекватності за критерієм Фішера"""
        S_ad = self.m / (self.n - d) * sum([(self.y_new[i] - self.y_av[i]) ** 2 for i
in range(len(self.y))])
        S_kv = self.dispersion()
        S_kv_aver = sum(S_kv) / self.n
        F_p = S_ad / S_kv_aver
        return F_p

    def check(self):
        """Проведення статистичних перевірок"""
        student = partial(t.ppf, q=1 - 0.025)
        t_student = student(df=self.f3)
        ts = self.student()
        res = [t for t in ts if t > t_student]
        B = self.count_koefs()
        final_k = [B[ts.index(i)] for i in ts if i in res]
        for j in range(self.n):
            self.y_new.append(self.regression([self.x[j][ts.index(i)] for i in ts if i
in res], final_k))
        print(f'\nЗначення "y" з коефіцієнтами {final_k}')
```

```
        print(self.y_new)
        d = len(res)
        f4 = self.n - d
        F_p = self.fisher(d)
        print('\nКритерій Стьюдента:\n', ts)
        print('Коефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.format(
            [i for i in B if i not in final_k]))

        print('\nПеревірка за критерієм Кохрена')
        Gp, G_kr = self.kohren()
        print(f'Gp = {Gp}')
        if Gp < G_kr:
            print(f'З ймовірністю {1 - self.q} дисперсії однорідні.')
        else:
            print("Необхідно збільшити кількість дослідів")
            self.m += 1
            FractionalExperiment(self.n, self.m)

        fisher = partial(f.ppf, q=1 - 0.05)
        f_t = fisher(dfn=f4, dfd=self.f3)  # табличне знач
        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')


experiment = FractionalExperiment(7, 8)
experiment.check()
```

Результати роботи програми:

```
Рівняння регресії
y = 226.31 + 0.05*x1 + 0.03*x2 + 0.13*x3

Значення "y" з коефіцієнтами [226.31, 226.31, 0.03, 0.13]
[458.57, 459.82000000000005, 459.22, 459.17, 459.22, 459.17, 458.57]

Критерій Стьюдента:
 [313.1107214389738, 313.1107214389738, 7169.021421277649, 10513.475861139666]
Коефіцієнти [0.05] статистично незначущі, тому ми виключаємо їх з рівняння.

Перевірка за критерієм Кохрена
Gp = 0.260341412471331
З ймовірністю 0.95 дисперсії однорідні.

Перевірка адекватності за критерієм Фішера
Fp => 30310.51059030382
F_t => 2.7939488515842408
Математична модель не адекватна експериментальним даним

Process finished with exit code 0
```

Контрольні запитання

1) *Що називається дробовим факторним експериментом?*

У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

2) *Для чого потрібно розрахункове значення Кохрена?*

Критерій Кохрена використовують для порівняння трьох і більше виборок однакового обсягу n.

3) *Для чого перевіряється критерій Стьюдента?*

Якщо теоретичний коефіцієнт bi= 0, це означає, що в апроксимуючому поліномі відповідний доданок (фактор) відсутній. Чим менше значення bi, тим менше вплив відповідного фактора. За критерієм Стьюдента перевіряється значущість коефіцієнтів.

4) *Чим визначається критерій Фішера і як його застосовувати?*

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.

Висновки:

В ході виконання лабораторної роботи було проведено трьохфакторний експеримент. Складено матрицю планування, знайдено коефіцієнти рівняння регресії, проведено 3 статистичні перевірки.