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 №3
З дисципліни «Методи оптимізації та планування»
ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

ВИКОНАЛА:
Студентка II курсу ФІОТ
Групи ІО-92
Бондар Х.В. - 9201

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

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

Варіант завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
201	-10	50	-20	40	-20	-15

Програмний код:

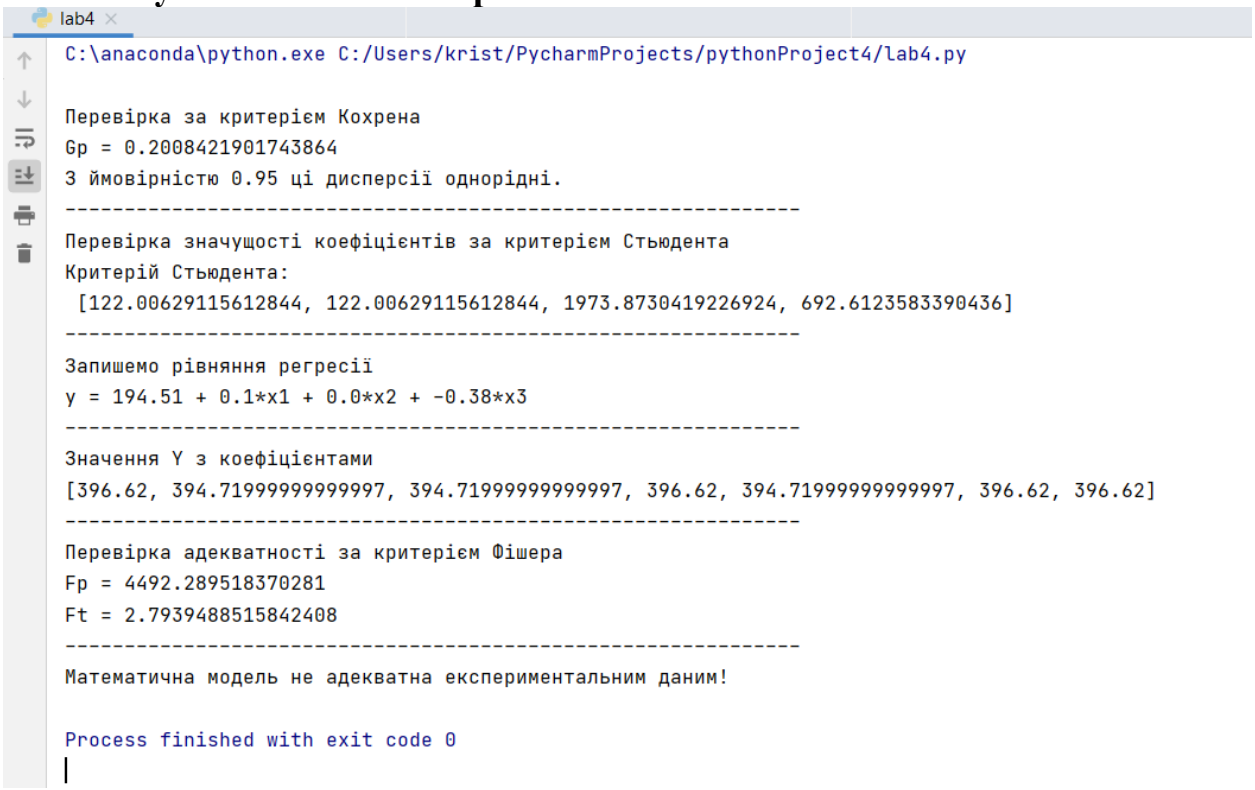
```
from random import *
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial


class Experiment:
    def __init__(self, n, m):
        self.n = n
        self.m = m
        self.x_min = (-10 - 20 - 20) / 3
        self.x_max = (50 + 40 - 15) / 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, 50), (-20, 40), (-20, -15)] #значення за
варіантом
        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

#функція для підстановки коефіцієнтів у рівняння регресії

    def podstanovka(self, x, b):
        y = sum([x[i] * b[i] for i in range(len(x))])
```

```
        return y


# функція для розрахунку коефіцієнтів р-ння регресії

    def count(self):
        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('----------------------------------------------------------------
')
        print('Запишемо рівняння регресії')
        print(f'y = {B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

        return B

#функція для розрахунку дисперсії

    def count_count_dispersion(self):
        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

#перевірка однорідності дисперсій за критерієм Кохрена

    def kr_kohrena(self):
        q1 = self.q / self.f1
        kr_fishera_value = f.ppf(q=1 - q1, dfn=self.f2, dfd=(self.f1 - 1) *
self.f2)
        G_cr = kr_fishera_value / (kr_fishera_value + self.f1 - 1)
        s = self.count count dispersion()
        Gp = max(s) / sum(s)
        return Gp, G_cr

#перевірка знащущості коефіцієнтів за критерієм Стьюдента

    def kr_studenta(self):
        def bs():
            res = [sum(1 * y for y in self.y_av) / self.n]
```

```
            for i in range(3):
                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.count_count_dispersion()
        s_kv_aver = sum(S_kv) / self.n

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

#перевірка адекватності за критерієм Фішера

    def kr_fishera(self, d):
        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.count_count_dispersion()
        S_kv_aver = sum(S_kv) / self.n
        F_p = S_ad / S_kv_aver
        return F_p

#перевірка

    def perevirka(self):
        kr_studenta = partial(t.ppf, q=1 - 0.025)
        t_kr_studenta = kr_studenta(df=self.f3)

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

        ts = self.kr_studenta()
        print('Перевірка значущості коефіцієнтів за критерієм Стьюдента')
        print('Критерій Стьюдента:\n', ts)
        res = [t for t in ts if t > t_kr_studenta]
        B = self.count()
        final_k = [B[ts.index(i)] for i in ts if i in res]
        print('----------------------------------------------------------------
')
        for j in range(self.n):
            self.y_new.append(self.podstanovka([self.x[j][ts.index(i)] for i
in ts if i in res], final_k))

        print(f'Значення Y з коефіцієнтами')
        print(self.y_new)
        print('----------------------------------------------------------------
')

        d = len(res)
        f4 = self.n - d
        F_p = self.kr_fishera(d)
```

```
            kr_fishera = partial(f.ppf, q=1 - 0.05)
            f_t = kr_fishera(dfn=f4, dfd=self.f3)  # табличне знач
            print('Перевірка адекватності за критерієм Фішера')
            print('Fp =', F_p)
            print('Ft =', f_t)
            print('----------------------------------------------------------------
')
            if F_p < f_t:
                print('Математична модель адекватна експериментальним даним!')
            else:
                print('Математична модель не адекватна експериментальним даним!')


experiment = Experiment(7, 8)
experiment.perevirka()
```

Результат виконання роботи:



Висновок:

В даній лабораторній роботі я провела дробовий трьохфакторний експеримент з трьома статистичними перевірками і отримала коефіцієнти рівняння регресії. Вдалий запуск програми підтверджує правильність написання програмного коду. Кінцева мета досягнута.

Відповіді на контрольні питання:

1. Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту.

2. Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3. За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння регресії.

4. Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об’єкту.