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 експерименту» на тему

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ.»

ВИКОНАВ: студент ІІ курсу ФІОТ

групи ІО-93

Куцурайс Георгій Олегович

Варіант: 316

ПЕРЕВІРИВ:

ас. Регіда П. Г.

Київ – 2021

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

**Виконання лабораторної роботи:**

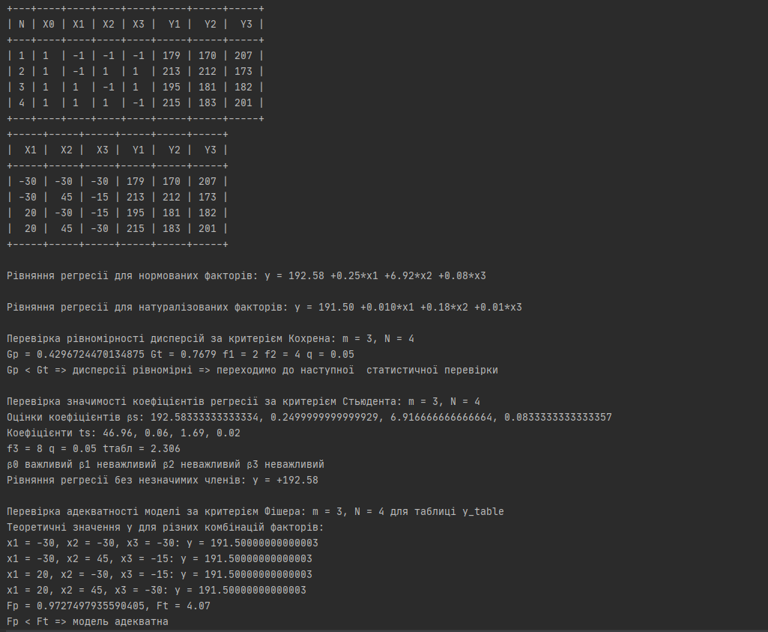
Варіант



**Код програми:**

import xlrd  
import random  
import numpy as np  
import math  
import itertools  
from prettytable import PrettyTable  
  
  
class Lab3:  
 def \_\_init\_\_(self):  
 self.N = 4  
 self.m = 3  
 self.x1\_min = -10  
 self.x1\_max = 50  
 self.x2\_min = -20  
 self.x2\_max = 60  
 self.x3\_min = 50  
 self.x3\_max = 55  
 self.x\_average\_min = int((self.x1\_min + self.x2\_min + self.x3\_min)/3)  
 self.x\_average\_max = int((self.x1\_max + self.x2\_max + self.x3\_max)/3)  
 self.y\_min = 200 + self.x\_average\_min  
 self.y\_max = 200 + self.x\_average\_max  
  
 self.factors\_table = [[1, -1, -1, -1],  
 [1, -1, +1, +1],  
 [1, +1, -1, +1],  
 [1, +1, +1, -1]]  
  
 self.generate\_matrix()  
  
 def generate\_matrix(self):  
 self.matrix = [[random.randint(self.y\_min, self.y\_max) for i in range(self.m)] for j in range(4)]  
 print("Дані варіанту 316 : y\_max = {} y\_min = {} x1\_min = {} x1\_max = {} x2\_min = {} x2\_max = {} x3\_min = {} x3\_max = {}".format(  
 self.y\_max, self.y\_min, self.x1\_min, self.x1\_max, self.x2\_min, self.x2\_max, self.x3\_min, self.x3\_max))  
 self.naturalized\_factors\_table = [[self.x1\_min, self.x2\_min, self.x3\_min], [self.x1\_min, self.x2\_max, self.x3\_max], [self.x1\_max, self.x2\_min, self.x3\_max], [self.x1\_max, self.x2\_max, self.x3\_min]]  
  
  
 table0 = PrettyTable()  
 table0.field\_names = (["N", "X0", "X1", "X2", "X3"] + ["Y{}".format(i+1) for i in range(self.m)])  
 for i in range(self.N):  
 table0.add\_row([i+1] + self.factors\_table[i] + self.matrix[i])  
 print(table0)  
  
 table1 = PrettyTable()  
 table1.field\_names = (["X1", "X2", "X3"] + ["Y{}".format(i + 1) for i in range(self.m)])  
 for i in range(self.N):  
 table1.add\_row(self.naturalized\_factors\_table[i] + self.matrix[i])  
 print(table1)  
  
 self.calculate()  
  
 def calculate(self):  
 self.average\_Y1 = sum(self.matrix[0][j] for j in range(self.m)) / self.m  
 self.average\_Y2 = sum(self.matrix[1][j] for j in range(self.m)) / self.m  
 self.average\_Y3 = sum(self.matrix[2][j] for j in range(self.m)) / self.m  
 self.average\_Y4 = sum(self.matrix[3][j] for j in range(self.m)) / self.m  
 self.average\_Y = [self.average\_Y1, self.average\_Y2, self.average\_Y3, self.average\_Y4]  
  
  
 self.mx1 = sum(self.naturalized\_factors\_table[i][0] for i in range(self.N)) / self.N  
 self.mx2 = sum(self.naturalized\_factors\_table[i][1] for i in range(self.N)) / self.N  
 self.mx3 = sum(self.naturalized\_factors\_table[i][2] for i in range(self.N)) / self.N  
  
 self.my = sum(self.average\_Y) / self.N  
  
 self.a1 = sum(self.naturalized\_factors\_table[i][0] \* self.average\_Y[i] for i in range(self.N)) / self.N  
 self.a2 = sum(self.naturalized\_factors\_table[i][1] \* self.average\_Y[i] for i in range(self.N)) / self.N  
 self.a3 = sum(self.naturalized\_factors\_table[i][2] \* self.average\_Y[i] for i in range(self.N)) / self.N  
  
  
 self.a11 = sum((self.naturalized\_factors\_table[i][0]) \*\* 2 for i in range(self.N)) / self.N  
 self.a22 = sum((self.naturalized\_factors\_table[i][1]) \*\* 2 for i in range(self.N)) / self.N  
 self.a33 = sum((self.naturalized\_factors\_table[i][2]) \*\* 2 for i in range(self.N)) / self.N  
  
 self.a12 = sum(self.naturalized\_factors\_table[i][0] \* self.naturalized\_factors\_table[i][1] for i in range(self.N)) / self.N  
 self.a13 = sum(self.naturalized\_factors\_table[i][0] \* self.naturalized\_factors\_table[i][2] for i in range(self.N)) / self.N  
 self.a23 = sum(self.naturalized\_factors\_table[i][1] \* self.naturalized\_factors\_table[i][2] for i in range(self.N)) / self.N  
  
 equations\_sys\_coefficients = [[1, self.mx1, self.mx2, self.mx3],  
 [self.mx1, self.a11, self.a12, self.a13],  
 [self.mx2, self.a12, self.a22, self.a23],  
 [self.mx3, self.a13, self.a23, self.a33]]  
 equations\_sys\_free\_members = [self.my, self.a1, self.a2, self.a3]  
 self.b\_coefficients = np.linalg.solve(equations\_sys\_coefficients, equations\_sys\_free\_members)  
 b\_normalized\_coefficients = np.array([np.average(self.average\_Y),  
 np.average(self.average\_Y \* np.array([i[1] for i in self.factors\_table])),  
 np.average(self.average\_Y \* np.array([i[2] for i in self.factors\_table])),  
 np.average(self.average\_Y \* np.array([i[3] for i in self.factors\_table]))])  
  
 print("\nРівняння регресії для нормованих факторів: y = {0:.2f} {1:+.2f}\*x1 {2:+.2f}\*x2 {3:+.2f}\*x3".format(\*b\_normalized\_coefficients))  
 print("\nРівняння регресії для натуралізованих факторів: y = {0:.2f} {1:+.3f}\*x1 {2:+.2f}\*x2 {3:+.2f}\*x3".format(\*self.b\_coefficients))  
  
 self.cochran\_criteria(self.m, self.N, self.matrix)  
  
  
  
 def cochran\_criteria(self, m, N, y\_table):  
 print("\nПеревірка рівномірності дисперсій за критерієм Кохрена: m = {}, N = {}".format(m, N))  
 cochran\_table = xlrd.open\_workbook("Cochran.xls").sheet\_by\_index(0)  
 y\_variations = [np.var(i) for i in y\_table]  
 max\_y\_variation = max(y\_variations)  
 gp = max\_y\_variation/sum(y\_variations)  
 f1 = m - 1  
 f2 = N  
 p = 0.95  
 q = 1-p  
 column = f1  
 row = f2-1  
 gt = cochran\_table.row\_values(row-1)[column-1]/math.pow(10,4)  
 print("Gp = {} Gt = {} f1 = {} f2 = {} q = {:.2f}".format(gp, gt, f1, f2, q))  
 if gp < gt:  
 print("Gp < Gt => дисперсії рівномірні => переходимо до наступної статистичної перевірки")  
 self.student\_criteria(self.m, self.N, self.matrix, self.factors\_table)  
 else:  
 print("Gp > Gt => дисперсії нерівномірні => змінюємо значення m => m = m+1")  
 self.m = self.m + 1  
 self.generate\_matrix()  
  
  
  
 def student\_criteria(self, m, N, y\_table, factors\_table):  
 print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {}, N = {}".format(m, N))  
 student\_table = xlrd.open\_workbook("Student.xls").sheet\_by\_index(0)  
  
 average\_variation = np.average(list(map(np.var, y\_table)))  
 standard\_deviation\_beta\_s = math.sqrt(average\_variation / N / m)  
  
 y\_averages = np.array(list(map(np.average, y\_table)))  
 x\_i = np.array([[el[i] for el in factors\_table] for i in range(len(factors\_table))])  
 coefficients\_beta\_s = np.array([np.average(self.average\_Y\*x\_i[i]) for i in range(len(x\_i))])  
  
 print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(str,coefficients\_beta\_s))))  
 t\_i = np.array([abs(coefficients\_beta\_s[i])/standard\_deviation\_beta\_s for i in range(len(coefficients\_beta\_s))])  
 print("Коефіцієнти ts: " + ", ".join(list(map(lambda i: "{:.2f}".format(i), t\_i))))  
 f3 = (m-1)\*N  
 p = 0.95  
 q = 0.05  
 t = float(student\_table.col\_values(3)[f3].replace(",", "."))  
 self.importance = [True if el > t else False for el in list(t\_i)]  
 # print result data  
 print("f3 = {} q = {} tтабл = {}".format(f3, q, t))  
 beta\_i = ["β{}".format(i) for i in range(N)]  
 importance\_to\_print = ["важливий" if i else "неважливий" for i in self.importance]  
 to\_print = list(zip(beta\_i, importance\_to\_print))  
 x\_i\_names = [""] + list(itertools.compress(["x{}".format(i) for i in range(N)], self.importance))[1:]  
 betas\_to\_print = list(itertools.compress(coefficients\_beta\_s, self.importance))  
 print("{0[0]} {0[1]} {1[0]} {1[1]} {2[0]} {2[1]} {3[0]} {3[1]}".format(\*to\_print))  
 equation = " ".join(["".join(i) for i in zip(list(map(lambda x: "{:+.2f}".format(x), betas\_to\_print)),x\_i\_names)])  
 print("Рівняння регресії без незначимих членів: y = " + equation)  
 self.d = len(betas\_to\_print)  
 self.factors\_table2 = [np.array([1] + list(i)) for i in self.naturalized\_factors\_table]  
 self.fisher\_criteria(self.m, self.N, 1, self.factors\_table2, self.matrix, self.b\_coefficients, self.importance)  
  
 def calculate\_theoretical\_y(self, x\_table, b\_coefficients, importance):  
 x\_table = [list(itertools.compress(row, importance)) for row in x\_table]  
 b\_coefficients = list(itertools.compress(b\_coefficients, importance))  
 y\_vals = np.array([sum(map(lambda x, b: x \* b, row, b\_coefficients)) for row in x\_table])  
 return y\_vals  
  
 def fisher\_criteria(self, m, N, d, factors\_table, matrix, b\_coefficients, importance):  
 print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N = {} для таблиці y\_table".format(m, N))  
 fisher\_table = xlrd.open\_workbook("Fisher.xls").sheet\_by\_index(0)  
  
 f3 = (m - 1) \* N  
 f4 = N - d  
  
 theoretical\_y = self.calculate\_theoretical\_y(factors\_table, b\_coefficients, importance)  
 theoretical\_values\_to\_print = list(  
 zip(map(lambda x: "x1 = {0[1]}, x2 = {0[2]}, x3 = {0[3]}".format(x), factors\_table), theoretical\_y))  
  
 print("Теоретичні значення y для різних комбінацій факторів:")  
 print("\n".join(["{arr[0]}: y = {arr[1]}".format(arr=el) for el in theoretical\_values\_to\_print]))  
 y\_averages = np.array(list(map(np.average, matrix)))  
 s\_ad = m / (N - d) \* (sum((theoretical\_y - y\_averages) \*\* 2))  
 y\_variations = np.array(list(map(np.var, matrix)))  
 s\_v = np.average(y\_variations)  
 f\_p = float(s\_ad / s\_v)  
 f\_t = float((fisher\_table.row\_values(f3) if f3 <= 30 else fisher\_table.row\_values(30))[f4].replace(",", "."))  
 print("Fp = {}, Ft = {}".format(f\_p, f\_t))  
 print("Fp < Ft => модель адекватна" if f\_p < f\_t else "Fp > Ft => модель неадекватна")  
  
results = []  
for i in range(1):  
 Lab3()

**Результат роботи:**



**Висновок**: В даній лабораторній роботі проведено дробовий трьохфакторний експеримент з трьома статистичними перевірками і отримано коефіцієнти рівняння регресії.

**Контрольні запитання**

1. **Що називається дробовим факторним експериментом?**

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

**2. Для чого потрібно розрахункове значення Кохрена?**

Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій..

**3. Для чого перевіряється критерій Стьюдента?**

За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння регресії.

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

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