**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний технічний університет України**

**«Київський Політехнічний Інститут»**

*Факультет інформатики та обчислювальної техніки*

*Кафедра обчислювальної техніки*

**Лабораторна робота №3**

*з дисципліни*

*«Методи оптимізації та планування експерименту»*

*Тема:* ***ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ****.*

**Виконав:**

Студент 2-го курсу ФІОТ

групи ІВ-93

Маловиця Станіслав

**Перевірив:**

*Регіда П. Г*ю

**Київ – 2021**

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

Варант завдання:

****

import random  
import numpy  
import scipy.stats  
  
import tkinter.messagebox  
  
root = tkinter.Tk()  
  
x1\_min = -25  
x1\_max = 75  
  
x2\_min = 5  
x2\_max = 40  
  
x3\_min = 15  
x3\_max = 25  
  
xm\_min = (x1\_min + x2\_min + x3\_min) / 3  
xm\_max = (x1\_max + x2\_max + x3\_max) / 3  
y\_min = 200 + xm\_min  
y\_max = 200 + xm\_max  
  
xn = [[-1, -1, -1],  
 [-1, 1, 1],  
 [1, -1, 1],  
 [1, 1, -1]]  
  
x = [[10, -35, 10],  
 [10, 15, 15],  
 [60, -35, 15],  
 [60, 15, 10]]  
  
m = 2  
y = [[random.randint(int(y\_min), int(y\_max)) for i in range(m)] for j in range(4)]  
  
  
def kohren(dispersion, m):  
 gt = {1: 0.9065, 2: 0.7679, 3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7:0.5365, 8: 0.5175, 9: 0.5017, 10: 0.4884}  
 gp = max(dispersion) / sum(dispersion)  
 return gp < gt[m - 1]  
  
  
def student(dispersion\_reproduction, m, y\_mean, xn):  
 dispersion\_statistic\_mark = (dispersion\_reproduction / (4 \* m)) \*\* 0.5  
  
 beta = [1 / 4 \* sum(y\_mean[j] for j in range(4))]  
 for i in range(3):  
 b = 0  
 for j in range(4):  
 b += y\_mean[j] \* xn[j][i]  
 beta.append(1 / 4 \* b)  
  
 t = []  
 for i in beta:  
 t.append(abs(i) / dispersion\_statistic\_mark)  
  
 check\_st = scipy.stats.t.ppf((1 + 0.95)/2, (m-1) \* 4)  
  
 return t[0] > check\_st, t[1] > check\_st, t[2] > check\_st, t[3] > check\_st  
  
  
def fisher(m, d, y\_mean, yo, dispersion\_reproduction):  
   
 dispersion\_ad = 0  
 for i in range(4):  
 dispersion\_ad += (yo[i] - y\_mean[i]) \*\* 2  
   
 dispersion\_ad = dispersion\_ad \* m / (4 - d)  
  
 fp = dispersion\_ad / dispersion\_reproduction  
 check\_f = scipy.stats.f.ppf(0.95, 4 - d, (m-1)\*4)  
   
 return fp < check\_f  
  
  
def normalized\_multiplier(x, y\_mean):  
 mx = [0, 0, 0]  
 axx = [0, 0, 0]  
 ax = [0, 0, 0]  
 for i in range(3):  
 for j in range(4):  
 mx[i] += x[j][i]  
 axx[i] += x[j][i] \*\* 2  
 ax[i] += x[j][i] \* y\_mean[j]  
 mx[i] /= 4  
 axx[i] /= 4  
 ax[i] /= 4  
   
 my = sum(y\_mean) / 4  
  
 a12 = (x[0][0] \* x[0][1] + x[1][0] \* x[1][1] + x[2][0] \* x[2][1] + x[3][0] \* x[3][1]) / 4  
 a13 = (x[0][0] \* x[0][2] + x[1][0] \* x[1][2] + x[2][0] \* x[2][2] + x[3][0] \* x[3][2]) / 4  
 a23 = (x[0][1] \* x[0][2] + x[1][1] \* x[1][2] + x[2][1] \* x[2][2] + x[3][1] \* x[3][2]) / 4  
  
 a = numpy.array([[1, \*mx],  
 [mx[0], axx[0], a12, a13],  
 [mx[1], a12, axx[1], a23],  
 [mx[2], a13, a23, axx[2]]])  
 c = numpy.array([my, \*ax])  
 b = numpy.linalg.solve(a, c)  
 return b  
  
  
def next\_m(arr):  
 for i in range(4):  
 arr[i].append(random.randint(int(y\_min), int(y\_max)))  
  
  
while True:  
 while True:   
 y\_mean = []  
 for i in range(4):  
 y\_mean.append(sum(y[i]) / m)  
   
 dispersion = []  
 for i in range(len(y)):  
 dispersion.append(0)  
 for j in range(m):  
 dispersion[i] += (y\_mean[i] - y[i][j]) \*\* 2  
 dispersion[i] /= m  
  
 dispersion\_reproduction = sum(dispersion) / 4  
  
 if kohren(dispersion, m):  
 break  
 else:  
 m += 1  
 next\_m(y)  
  
 k = student(dispersion\_reproduction, m, y\_mean, xn)  
 d = sum(k)  
   
 b = normalized\_multiplier(x, y\_mean)  
 b = [b[i] \* k[i] for i in range(4)]  
  
 yo = []  
 for i in range(4):  
 yo.append(b[0] + b[1] \* x[i][0] + b[2] \* x[i][1] + b[3] \* x[i][2])  
   
 if d == 4:  
 m += 1  
 next\_m(y)  
   
 elif fisher(m, d, y\_mean, yo, dispersion\_reproduction):  
 break  
   
 else:  
 m += 1  
 next\_m(y)  
  
  
tkinter.Label(text="x1").grid()  
  
tkinter.Label(text="x2").grid(row=0, column=1)  
tkinter.Label(text="x3").grid(row=0, column=2)  
for i in range(m):  
 tkinter.Label(text="yi" + str(i + 1)).grid(row=0, column=i + 3)  
for i in range(len(x)):  
 for j in range(len(x[i])):  
 tkinter.Label(text=x[i][j]).grid(row=i + 1, column=j)  
for i in range(len(y)):  
 for j in range(len(y[i])):  
 tkinter.Label(text=(y[i][j])).grid(row=i + 1, column=j + 3)  
tkinter.Label(text="Рівняння регресії:").grid(columnspan=m + 3)  
text = "y = " + "{0:.2f}".format(b[0])  
for i in range(3):  
 if b[i + 1] != 0:  
 text = text + " + {0:.2f}".format(b[i + 1]) + " \* x" + str(i + 1)  
  
tkinter.Label(text=text).grid(columnspan=m + 3)  
tkinter.Label(text="Перевірка:").grid(columnspan=m + 3)  
  
for i in range(4):  
 tkinter.Label(text="yc" + str(i + 1) + " =" + "{0:.2f}".format(y\_mean[i])).grid(columnspan=m + 3)  
 tkinter.Label(text="y" + str(i + 1) + " = " + "{0:.2f}".format(yo[i])).grid(columnspan=m + 3)  
  
root.mainloop()

