МІНІСТЕРСТЕРСВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛБНОЇ ТЕХНІКИ

Звіт з лабораторних робіт

З курсу «Методи оптимізаціі та планування експерименту»

Лабораторна робота №4

Виконала:

Бугайченко П.І.

Номер заліковки: 8203

Перевірив:

Регіда П.Г.

Київ 2020 р.

**Виконання роботи**

**Варіант 202 (-10,50)(-20,40)(-20, -15)**

import random  
import numpy  
import math  
from scipy.stats import t, f  
  
  
def table\_student(prob, n, m):  
 x\_vec = [i\*0.0001 for i in range(int(5/0.0001))]  
 par = 0.5 + prob/0.1\*0.05  
 f3 = (m - 1) \* n  
 for i in x\_vec:  
 if abs(t.cdf(i, f3) - par) < 0.000005:  
 return i  
  
  
def table\_fisher(prob, n, m, d):  
 x\_vec = [i\*0.001 for i in range(int(10/0.001))]  
 f3 = (m - 1) \* n  
 for i in x\_vec:  
 if abs(f.cdf(i, n-d, f3)-prob) < 0.0001:  
 return i  
  
  
def getb(xnat, ym):  
 def ai1(x, k):  
 a = [0, 0, 0, 0, 0, 0, 0, 0]  
 for i in range(0, 8):  
 a[0] += x[i][k]  
 a[1] += xnat[i][0] \* x[i][k]  
 a[2] += xnat[i][1] \* x[i][k]  
 a[3] += xnat[i][2] \* x[i][k]  
 a[4] += xnat[i][0] \* xnat[i][1] \* x[i][k]  
 a[5] += xnat[i][0] \* xnat[i][2] \* x[i][k]  
 a[6] += xnat[i][1] \* xnat[i][2] \* x[i][k]  
 a[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2] \* x[i][k]  
 return a  
  
 def ai2(x, k, l):  
 a = [0, 0, 0, 0, 0, 0, 0, 0]  
 for i in range(0, 8):  
 a[0] += x[i][k] \* x[i][l]  
 a[1] += xnat[i][0] \* x[i][k] \* x[i][l]  
 a[2] += xnat[i][1] \* x[i][k] \* x[i][l]  
 a[3] += xnat[i][2] \* x[i][k] \* x[i][l]  
 a[4] += xnat[i][0] \* xnat[i][1] \* x[i][k] \* x[i][l]  
 a[5] += xnat[i][0] \* xnat[i][2] \* x[i][k] \* x[i][l]  
 a[6] += xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l]  
 a[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l]  
 return a  
  
 def ai3(x, k, l, m):  
 a = [0, 0, 0, 0, 0, 0, 0, 0]  
 for i in range(0, 8):  
 a[0] += x[i][k] \* x[i][l] \* x[i][m]  
 a[1] += xnat[i][0] \* x[i][k] \* x[i][l] \* x[i][m]  
 a[2] += xnat[i][1] \* x[i][k] \* x[i][l] \* x[i][m]  
 a[3] += xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]  
 a[4] += xnat[i][0] \* xnat[i][1] \* x[i][k] \* x[i][l] \* x[i][m]  
 a[5] += xnat[i][0] \* xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]  
 a[6] += xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]  
 a[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]  
 return a  
 a = []  
 a1 = [8, 0, 0, 0, 0, 0, 0, 0]  
 for i in range(0, 8):  
 a1[1] += xnat[i][0]  
 a1[2] += xnat[i][1]  
 a1[3] += xnat[i][2]  
 a1[4] += xnat[i][0] \* xnat[i][1]  
 a1[5] += xnat[i][0] \* xnat[i][2]  
 a1[6] += xnat[i][1] \* xnat[i][2]  
 a1[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2]  
 a.append(a1)  
 a.append(ai1(xnat, 0))  
 a.append(ai1(xnat, 1))  
 a.append(ai1(xnat, 2))  
 a.append(ai2(xnat, 0, 1))  
 a.append(ai2(xnat, 0, 2))  
 a.append(ai2(xnat, 1, 2))  
 a.append(ai3(xnat, 0, 1, 2))  
 c = [0, 0, 0, 0, 0, 0, 0, 0]  
 for i in range(0, 8):  
 c[0] += ym[i]  
 c[1] += ym[i] \* xnat[i][0]  
 c[2] += ym[i] \* xnat[i][1]  
 c[3] += ym[i] \* xnat[i][2]  
 c[4] += ym[i] \* xnat[i][0] \* xnat[i][1]  
 c[5] += ym[i] \* xnat[i][0] \* xnat[i][2]  
 c[6] += ym[i] \* xnat[i][1] \* xnat[i][2]  
 c[7] += ym[i] \* xnat[i][0] \* xnat[i][1] \* xnat[i][2]  
 ax = numpy.array([[a[0][0], a[0][1], a[0][2], a[0][3], a[0][4], a[0][5], a[0][6], a[0][7]],  
 [a[1][0], a[1][1], a[1][2], a[1][3], a[1][4], a[1][5], a[1][6], a[1][7]],  
 [a[2][0], a[2][1], a[2][2], a[2][3], a[2][4], a[2][5], a[2][6], a[2][7]],  
 [a[3][0], a[3][1], a[3][2], a[3][3], a[3][4], a[3][5], a[3][6], a[3][7]],  
 [a[4][0], a[4][1], a[4][2], a[4][3], a[4][4], a[4][5], a[4][6], a[4][7]],  
 [a[5][0], a[5][1], a[5][2], a[5][3], a[5][4], a[5][5], a[5][6], a[5][7]],  
 [a[6][0], a[6][1], a[6][2], a[6][3], a[6][4], a[6][5], a[6][6], a[6][7]],  
 [a[7][0], a[7][1], a[7][2], a[7][3], a[7][4], a[7][5], a[7][6], a[7][7]]])  
 cx = numpy.array([c[0], c[1], c[2], c[3], c[4], c[5], c[6], c[7]])  
 b = numpy.linalg.solve(ax, cx)  
 return b  
  
  
def getbn(xnorm, ym):  
 b = [0 for i in range(0, N)]  
 b[0] = sum(ym) / N  
 for i in range(0, N):  
 b[1] += ym[i] \* xnorm[i][1] / N  
 b[2] += ym[i] \* xnorm[i][2] / N  
 b[3] += ym[i] \* xnorm[i][3] / N  
 b[4] += ym[i] \* xnorm[i][1] \* xnorm[i][2] / N  
 b[5] += ym[i] \* xnorm[i][1] \* xnorm[i][3] / N  
 b[6] += ym[i] \* xnorm[i][2] \* xnorm[i][3] / N  
 b[7] += ym[i] \* xnorm[i][1] \* xnorm[i][2] \* xnorm[i][3] / N  
 return b  
  
  
def dispersion(array\_y, array\_y\_average):  
 array\_dispersion = []  
  
 for j in range(N):  
 array\_dispersion.append(0)  
 for g in range(m):  
 array\_dispersion[j] += (array\_y[j][g] - array\_y\_average[j])\*\*2  
 array\_dispersion[j] /= m  
 return array\_dispersion  
  
  
def cohren(y\_array, y\_average\_array):  
 dispersion\_array = dispersion(y\_array, y\_average\_array)  
 max\_dispersion = max(dispersion\_array)  
 Gp = max\_dispersion/sum(dispersion\_array)  
 fisher = table\_fisher(0.95, N, m, 1)  
 Gt = fisher/(fisher+(m-1)-2)  
 return Gp < Gt  
  
  
def student(y\_array, y\_average\_array):  
 general\_dispersion = sum(dispersion(y\_array, y\_average\_array)) / N  
 statistic\_dispersion = math.sqrt(general\_dispersion / (N\*m))  
 beta = []  
 for i in range(N):  
 b = 0  
 for j in range(3):  
 b += y\_average\_array[i] \* xn[i][j]  
 beta.append(b / N)  
 ts = [abs(beta[i]) / statistic\_dispersion for i in range(N)]  
 f3 = (m-1)\*N  
 return ts[0] > table\_student(0.95, N, m), ts[1] > table\_student(0.95, N, m),\  
 ts[2] > table\_student(0.95, N, m), ts[3] > table\_student(0.95, N, m),\  
 ts[4] > table\_student(0.95, N, m), ts[5] > table\_student(0.95, N, m),\  
 ts[6] > table\_student(0.95, N, m), ts[7] > table\_student(0.95, N, m)  
  
  
def fisher(y\_average\_array, y0\_array, y\_array):  
 dispersion\_adequacy = 0  
 for i in range(N):  
 dispersion\_adequacy += (y0\_array[i] - y\_average\_array[i]) \*\* 2  
 dispersion\_adequacy = dispersion\_adequacy \* m / (N - d)  
 dispersion\_reproducibility = sum(dispersion(y\_array, y\_average\_array)) / N  
 Fp = dispersion\_adequacy / dispersion\_reproducibility  
 f3 = (m-1)\*N  
 f4 = N - d  
 return Fp < table\_fisher(0.95, N, m, d)  
  
#variant 202  
m = 3  
N = 8  
x1min = -10  
x1max = 50  
x2min = -20  
x2max = 40  
x3min = -20  
x3max = -15  
x\_min = (x1min + x2min + x3min) / 3  
x\_max = (x1max + x2max + x3max) / 3  
y\_min = round(200 + x\_min)  
y\_max = round(200 + x\_max)  
  
xn = [  
 [+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]  
]  
x = [  
 [x1min, x2min, x3min],  
 [x1min, x2min, x3max],  
 [x1min, x2max, x3min],  
 [x1min, x2max, x3max],  
 [x1max, x2min, x3min],  
 [x1max, x2min, x3max],  
 [x1max, x2max, x3min],  
 [x1max, x2max, x3max]  
]  
condition\_cohren = False  
condition\_fisher = False  
  
  
while not condition\_fisher:  
 while not condition\_cohren:  
 print(f'm={m}')  
 y = [[random.randint(y\_min, y\_max) for \_ in range(m)] for \_ in range(N)]  
 y\_average = [sum(y[i])/m for i in range(N)]  
 b = getb(x, y\_average)  
 print('рівняння регресії для натуральних значень факторів:')  
 print(f'y = {b[0]:.2f} + {b[1]:.2f} x1 + {b[2]:.2f} x2 + {b[3]:.2f} x3 + {b[4]:.2f} x1x2 + {b[5]:.2f} x1x3'  
 f' + {b[6]:.2f} x2x3 + {b[7]:.2f} x1x2x3')  
 bn = getbn(xn, y\_average)  
 print('рівняння регресії для нормованих значень факторів:')  
 print(f'y = {bn[0]:.2f} + {bn[1]:.2f} x1 + {bn[2]:.2f} x2 + {bn[3]:.2f} x3 + {bn[4]:.2f} x1x2 + {bn[5]:.2f} x1x3'  
 f' + {bn[6]:.2f} x2x3 + {bn[7]:.2f} x1x2x3')  
 condition\_cohren = cohren(y, y\_average)  
 if not condition\_cohren: #in way cohren is false m+1 and cycle again  
 m += 1  
 d = sum(student(y, y\_average))  
 print(f'значущі коефіціентм:{d}')  
 yo = []  
 for i in range(N):  
 yo.append(b[0] + b[1] \* x[i][0] + b[2] \* x[i][1] + b[3] \* x[i][2] + b[4]\*x[i][0]\*x[i][1] +  
 b[5]\*x[i][0]\*x[i][2] + b[6]\*x[i][1]\*x[i][2] + b[7]\*x[i][0]\*x[i][1]\*x[i][2])  
 if d != N:  
 condition\_fisher = fisher(y\_average, yo, y)  
 else:  
 condition\_fisher = True  
 if condition\_fisher:  
 print('модель адекватна експериментальним даним')  
 else:  
 print('неадекватно оригіналу')