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 № 4

З дисципліни «Методи наукових досліджеь» на тему

«Проведення трьохфакторного експерименту

При використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконала:

Студентка ІІ курсу ФІОТ

Групи ІВ-93

Трибунська Кароліна

Варіант: 326

Перевірив:

Регіда П. Г.

Київ – 2021

**Варіант**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | X1 | | X2 | | X3 | |
| min | max | min | max | min | max |
| 326 | -20 | 15 | 25 | 45 | 10 | 20 |

**Код програми**

import random  
import copy  
import math  
from tabulate import tabulate  
from scipy.stats import f, t  
  
  
def make\_experiment(m=3):  
 def dispersion(y\_list, avg\_y\_list, m):  
 Sy = []  
 elem = 0  
 for i in range(N):  
 for j in range(m):  
 elem += (y\_list[i][j] - avg\_y\_list[i]) \*\* 2  
 Sy.append(elem / m)  
 elem = 0  
 kar = [round(i, 2) for i in Sy]  
 print(kar)  
 return kar  
  
 def r(floatNumber):  
 return round(floatNumber, 2)  
  
 def str\_y():  
 return 'y = {} + ({}) \* x1 + ({}) \* x2 + ({}) \* x3 + ({}) \* x1x2 + ({}) \* x1x3 + ({}) \* x2x3 + ({}) \* x1x2x3'  
  
 def cochrane\_criterion(Sy):  
 print("\n=================Тест Кохрена=================\n")  
 Gp = max(Sy) / sum(Sy)  
 q = 0.05  
 q\_ = q / f2  
 chr = f.ppf(q=1 - q\_, dfn=f1, dfd=(f2 - 1) \* f1)  
 Gt = chr / (chr + f2 - 1)  
 print("Тест Кохрена: Gr = " + str(round(Gp, 3)))  
 if Gp < Gt:  
 print("Дисперсії однорідні з імовірністю 95%.")  
 pass  
 else:  
 print("\nДисперсії однорідні.\nПроведіть експеримент для m + = 1\n")  
 make\_experiment(4)  
  
 def student\_criterion(Sy, d):  
 print("\n=================Тест Стьюдента=================\n")  
 bettaList = [sum([Sy[i] \* x0[0] for i in range(N)]) / N,  
 sum([Sy[i] \* x1i[i] for i in range(N)]) / N,  
 sum([Sy[i] \* x2i[i] for i in range(N)]) / N,  
 sum([Sy[i] \* x3i[i] for i in range(N)]) / N,  
 sum([Sy[i] \* xFactors12Norm[i] for i in range(N)]) / N,  
 sum([Sy[i] \* xFactors13Norm[i] for i in range(N)]) / N,  
 sum([Sy[i] \* xFactors23Norm[i] for i in range(N)]) / N,  
 sum([Sy[i] \* xFactors123Norm[i] for i in range(N)]) / N]  
 bettaList = [round(i, 2) for i in bettaList]  
  
 tList = [bettaList[i] \* S for i in range(N)]  
  
 for i in range(N):  
 if tList[i] < t.ppf(q=0.975, df=f3):  
 bList[i] = 0  
 d -= 1  
 print('Виключити з рівняння коефіцієнт b' + str(i))  
 print("\n=================Кінцеве рівняння=================")  
 print(str\_y().format(r(bList[0]), r(bList[1]), r(bList[2]), r(bList[3]), r(bList[4]), r(bList[5]), r(bList[6]),  
 r(bList[7])))  
  
 def fisher\_criterion(d):  
 print("\n=================Критерій Фішера=================")  
 f4 = N - d  
 S\_ad = (m \* sum(  
 [(bList[0] + bList[1] \* x1i[i] + bList[2] \* x2i[i] + bList[3] \* x3i[i] + bList[4] \* xFactors12Norm[i] +  
 bList[5] \* xFactors13Norm[i] + bList[6] \* xFactors23Norm[i] + bList[7] \* xFactors123Norm[i]  
 - avgYList[i]) \*\* 2 for i in range(N)]) / f4)  
 Fp = S\_ad / Sb  
  
 if Fp > f.ppf(q=0.95, dfn=f4, dfd=f3): # перевірка за критерієм Фішера з використанням scipy  
 print('Рівняння регресії неадекватне оригіналу на рівні значущості 0,05.\nПовторіть експеримент для '  
 'm + 1')  
 make\_experiment(4)  
 else:  
 print('Рівняння регресії адекватне оригіналу на рівні значущості 0,05')  
  
 def print\_matrix():  
 print("\n" + "-\_-\_" \* 9 + "Матриця ПФЕ" + "-\_-\_" \* 9 + "\n",  
 tabulate([tableHeader,  
 x0 + [xFactors[0][0]] + [xFactors[0][1]] + [xFactors[0][2]] + xFactors12[0] + xFactors13[0] +  
 xFactors23[0] + xFactors123[0] + yList[0] + [avgYList[0]] + [Sy[0]],  
 x0 + [xFactors[1][0]] + [xFactors[1][1]] + [xFactors[1][2]] + xFactors12[1] + xFactors13[1] +  
 xFactors23[1] + xFactors123[1] + yList[1] + [avgYList[1]] + [Sy[1]],  
 x0 + [xFactors[2][0]] + [xFactors[2][1]] + [xFactors[2][2]] + xFactors12[2] + xFactors13[2] +  
 xFactors23[2] + xFactors123[2] + yList[2] + [avgYList[2]] + [Sy[2]],  
 x0 + [xFactors[3][0]] + [xFactors[3][1]] + [xFactors[3][2]] + xFactors12[3] + xFactors13[3] +  
 xFactors23[3] + xFactors123[3] + yList[3] + [avgYList[3]] + [Sy[3]],  
 x0 + [xFactors[4][0]] + [xFactors[4][1]] + [xFactors[4][2]] + xFactors12[4] + xFactors13[4] +  
 xFactors23[4] + xFactors123[4] + yList[4] + [avgYList[4]] + [Sy[4]],  
 x0 + [xFactors[5][0]] + [xFactors[5][1]] + [xFactors[5][2]] + xFactors12[5] + xFactors13[5] +  
 xFactors23[5] + xFactors123[5] + yList[5] + [avgYList[5]] + [Sy[5]],  
 x0 + [xFactors[6][0]] + [xFactors[6][1]] + [xFactors[6][2]] + xFactors12[6] + xFactors13[6] +  
 xFactors23[6] + xFactors123[6] + yList[6] + [avgYList[6]] + [Sy[6]],  
 x0 + [xFactors[7][0]] + [xFactors[7][1]] + [xFactors[7][2]] + xFactors12[7] + xFactors13[7] +  
 xFactors23[7] + xFactors123[7] + yList[7] + [avgYList[7]] + [Sy[7]],  
 ], headers="firstrow", tablefmt="pretty"))  
  
 print("\n" + "-\_-\_" \* 7 + "Нормалізована матриця ПФЕ" + "-\_-\_" \* 7 + "\n",  
 tabulate([tableHeader,  
 x0 + [xFactorsNorm[0][0]] + [xFactorsNorm[0][1]] + [xFactorsNorm[0][2]] + [xFactors12Norm[0]] +  
 [xFactors13Norm[0]] + [xFactors23Norm[0]] + [xFactors123Norm[0]] + yList[0] + [avgYList[0]] + [  
 Sy[0]],  
 x0 + [xFactorsNorm[1][0]] + [xFactorsNorm[1][1]] + [xFactorsNorm[1][2]] + [xFactors12Norm[1]] +  
 [xFactors13Norm[1]] + [xFactors23Norm[1]] + [xFactors123Norm[1]] + yList[1] + [avgYList[1]] + [  
 Sy[1]],  
 x0 + [xFactorsNorm[2][0]] + [xFactorsNorm[2][1]] + [xFactorsNorm[2][2]] + [xFactors12Norm[2]] +  
 [xFactors13Norm[2]] + [xFactors23Norm[2]] + [xFactors123Norm[2]] + yList[2] + [avgYList[2]] + [  
 Sy[2]],  
 x0 + [xFactorsNorm[3][0]] + [xFactorsNorm[3][1]] + [xFactorsNorm[3][2]] + [xFactors12Norm[3]] +  
 [xFactors13Norm[3]] + [xFactors23Norm[3]] + [xFactors123Norm[3]] + yList[3] + [avgYList[3]] + [  
 Sy[3]],  
 x0 + [xFactorsNorm[4][0]] + [xFactorsNorm[4][1]] + [xFactorsNorm[4][2]] + [xFactors12Norm[4]] +  
 [xFactors13Norm[4]] + [xFactors23Norm[4]] + [xFactors123Norm[4]] + yList[4] + [avgYList[4]] + [  
 Sy[4]],  
 x0 + [xFactorsNorm[5][0]] + [xFactorsNorm[5][1]] + [xFactorsNorm[5][2]] + [xFactors12Norm[5]] +  
 [xFactors13Norm[5]] + [xFactors23Norm[5]] + [xFactors123Norm[5]] + yList[5] + [avgYList[5]] + [  
 Sy[5]],  
 x0 + [xFactorsNorm[6][0]] + [xFactorsNorm[6][1]] + [xFactorsNorm[6][2]] + [xFactors12Norm[6]] +  
 [xFactors13Norm[6]] + [xFactors23Norm[6]] + [xFactors123Norm[6]] + yList[6] + [avgYList[6]] + [  
 Sy[6]],  
 x0 + [xFactorsNorm[7][0]] + [xFactorsNorm[7][1]] + [xFactorsNorm[7][2]] + [xFactors12Norm[7]] +  
 [xFactors13Norm[7]] + [xFactors23Norm[7]] + [xFactors123Norm[7]] + yList[7] + [avgYList[7]] + [  
 Sy[7]],  
 ], headers="firstrow", tablefmt="pretty"))  
  
 N = 8  
 if m == 3:  
 tableHeader = ["X0", "X1", "X2", "X3", "X12", "X13", "X23", "X123", "Y1", "Y2", "Y3", "avgY", "S^2"]  
 elif m == 4:  
 tableHeader = ["X0", "X1", "X2", "X3", "X12", "X13", "X23", "X123", "Y1", "Y2", "Y3", "Y4", "avgY", "S^2"]  
 else:  
 print("Невдалий експеримент")  
  
 xMinList = [-20, 25, 10]  
 xMaxList = [15, 45, 20]  
  
 avgXMin = sum(xMinList) / len(xMinList)  
 avgXMax = sum(xMaxList) / len(xMaxList)  
  
 yMin = 200 + avgXMin  
 yMax = 200 + avgXMax  
  
 yList = [[random.randint(int(yMin), int(yMax)) for yi in range(m)] for list\_y in range(N)]  
 avgYList = [round(sum(yList[i]) / len(yList[i]), 2) for i in range(len(yList))]  
  
 Sy = dispersion(yList, avgYList, m)  
  
 f1 = m - 1  
 f2 = N  
 f3 = f1 \* f2  
 d = 4  
  
 Sb = sum(Sy) / N  
 S = math.sqrt(Sb / (N \* m))  
  
 x0 = [1]  
 xFactorsNorm = [[-1, -1, -1],  
 [-1, -1, 1],  
 [-1, 1, -1],  
 [-1, 1, 1],  
 [1, -1, -1],  
 [1, -1, 1],  
 [1, 1, -1],  
 [1, 1, 1]]  
  
 xFactors12Norm = [xFactorsNorm[i][0] \* xFactorsNorm[i][1] for i in range(len(xFactorsNorm))]  
 xFactors13Norm = [xFactorsNorm[i][0] \* xFactorsNorm[i][2] for i in range(len(xFactorsNorm))]  
 xFactors23Norm = [xFactorsNorm[i][1] \* xFactorsNorm[i][2] for i in range(len(xFactorsNorm))]  
 xFactors123Norm = [xFactorsNorm[i][0] \* xFactorsNorm[i][1] \* xFactorsNorm[i][2] for i in range(len(xFactorsNorm))]  
  
 xFactors = [[-20, 25, 10],  
 [-20, 25, 20],  
 [-20, 45, 10],  
 [-20, 45, 20],  
 [15, 25, 10],  
 [15, 25, 20],  
 [15, 45, 10],  
 [15, 45, 20]]  
  
 xFactors12 = [[xFactors[i][0] \* xFactors[i][1]] for i in range(len(xFactors))]  
 xFactors13 = [[xFactors[i][0] \* xFactors[i][2]] for i in range(len(xFactors))]  
 xFactors23 = [[xFactors[i][1] \* xFactors[i][2]] for i in range(len(xFactors))]  
 xFactors123 = [[xFactors[i][0] \* xFactors[i][1] \* xFactors[i][2]] for i in range(len(xFactors))]  
  
 # Знаходження коефіцієнтів регресії для нормованих факторів ПФЕ  
 x1i = [xFactorsNorm[i][0] for i in range(N)]  
 x2i = [xFactorsNorm[i][1] for i in range(N)]  
 x3i = [xFactorsNorm[i][2] for i in range(N)]  
  
 bList = [0] \* 8  
 bList[0] = sum(avgYList) / N # b0  
 for i in range(N):  
 bList[1] += (avgYList[i] \* x1i[i]) / N # b1  
 bList[2] += (avgYList[i] \* x2i[i]) / N # b2  
 bList[3] += (avgYList[i] \* x3i[i]) / N # b3  
 bList[4] += (avgYList[i] \* x1i[i] \* x2i[i]) / N # b12  
 bList[5] += (avgYList[i] \* x1i[i] \* x3i[i]) / N # b13  
 bList[6] += (avgYList[i] \* x2i[i] \* x3i[i]) / N # b23  
 bList[7] += (avgYList[i] \* x1i[i] \* x2i[i] \* x3i[i]) / N # b123  
  
 print\_matrix()  
 print("\n=================Рівняння=================\n")  
 print(str\_y().format(r(bList[0]), r(bList[1]), r(bList[2]), r(bList[3]), r(bList[4]), r(bList[5]), r(bList[6]),  
 r(bList[7])))  
 cochrane\_criterion(Sy)  
 student\_criterion(Sy, d)  
 fisher\_criterion(d)  
  
  
make\_experiment()

**Результати роботи програми**

[42.0, 14.89, 18.67, 14.89, 48.67, 20.67, 26.89, 18.67]

-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_Матриця ПФЕ-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

+----+-----+----+----+------+------+-----+--------+-----+-----+-----+--------+-------+

| X0 | X1 | X2 | X3 | X12 | X13 | X23 | X123 | Y1 | Y2 | Y3 | avgY | S^2 |

+----+-----+----+----+------+------+-----+--------+-----+-----+-----+--------+-------+

| 1 | -20 | 25 | 10 | -500 | -200 | 250 | -5000 | 209 | 206 | 221 | 212.0 | 42.0 |

| 1 | -20 | 25 | 20 | -500 | -400 | 500 | -10000 | 218 | 209 | 216 | 214.33 | 14.89 |

| 1 | -20 | 45 | 10 | -900 | -200 | 450 | -9000 | 213 | 221 | 223 | 219.0 | 18.67 |

| 1 | -20 | 45 | 20 | -900 | -400 | 900 | -18000 | 217 | 224 | 215 | 218.67 | 14.89 |

| 1 | 15 | 25 | 10 | 375 | 150 | 250 | 3750 | 212 | 205 | 222 | 213.0 | 48.67 |

| 1 | 15 | 25 | 20 | 375 | 300 | 500 | 7500 | 214 | 207 | 218 | 213.0 | 20.67 |

| 1 | 15 | 45 | 10 | 675 | 150 | 450 | 6750 | 212 | 223 | 212 | 215.67 | 26.89 |

| 1 | 15 | 45 | 20 | 675 | 300 | 900 | 13500 | 218 | 216 | 208 | 214.0 | 18.67 |

+----+-----+----+----+------+------+-----+--------+-----+-----+-----+--------+-------+

-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_Нормалізована матриця ПФЕ-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

+----+----+----+----+-----+-----+-----+------+-----+-----+-----+--------+-------+

| X0 | X1 | X2 | X3 | X12 | X13 | X23 | X123 | Y1 | Y2 | Y3 | avgY | S^2 |

+----+----+----+----+-----+-----+-----+------+-----+-----+-----+--------+-------+

| 1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | 209 | 206 | 221 | 212.0 | 42.0 |

| 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 218 | 209 | 216 | 214.33 | 14.89 |

| 1 | -1 | 1 | -1 | -1 | 1 | -1 | 1 | 213 | 221 | 223 | 219.0 | 18.67 |

| 1 | -1 | 1 | 1 | -1 | -1 | 1 | -1 | 217 | 224 | 215 | 218.67 | 14.89 |

| 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 212 | 205 | 222 | 213.0 | 48.67 |

| 1 | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 214 | 207 | 218 | 213.0 | 20.67 |

| 1 | 1 | 1 | -1 | 1 | -1 | -1 | -1 | 212 | 223 | 212 | 215.67 | 26.89 |

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 218 | 216 | 208 | 214.0 | 18.67 |

+----+----+----+----+-----+-----+-----+------+-----+-----+-----+--------+-------+

=================Рівняння=================

y = 214.96 + (-1.04) \* x1 + (1.88) \* x2 + (0.04) \* x3 + (-0.96) \* x1x2 + (-0.46) \* x1x3 + (-0.54) \* x2x3 + (0.12) \* x1x2x3

=================Тест Кохрена=================

Тест Кохрена: Gr = 0.237

Дисперсії однорідні з імовірністю 95%.

=================Тест Стьюдента=================

Виключити з рівняння коефіцієнт b2

Виключити з рівняння коефіцієнт b3

Виключити з рівняння коефіцієнт b4

Виключити з рівняння коефіцієнт b5

Виключити з рівняння коефіцієнт b7

=================Кінцеве рівняння=================

y = 214.96 + (-1.04) \* x1 + (0) \* x2 + (0) \* x3 + (0) \* x1x2 + (0) \* x1x3 + (-0.54) \* x2x3 + (0) \* x1x2x3

=================Критерій Фішера=================

Рівняння регресії адекватне оригіналу на рівні значущості 0,05

**Висновки**

Проведений повний трьохфакторний експеримент. Знайдено рівняння регресії адекватне об'єкту. Складено матрицю планування для повного трьохфакторногоексперименту, знайдено значення відгуку Y. Знайдено значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Знайдено коефіцієнти рівняння регресії. Проведені 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера. На екран виведені висновки по адекватності регресії та значимості окремих коефіцієнтів і скореговане рівняння регресії.