**ЛАБОРАТОРНА РОБОТА № 2**

з дисципліни «Методи оптимізації та планування експерименту» на тему

” ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ”

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІВ-83

Кочерук Д.А.

Варіант: 311

ПЕРЕВІРИВ:

Регіда П. Г.

Київ – 2020



Варіант:

X1min = -25 X2min = 5 X3min = 15

X1max = 75 X2max = 40 X3max = 25

**Код програми:**

import random  
import numpy as np  
x1min = -25  
x1max = 75  
x2min = 5  
x2max = 40  
x3min = 15  
x3max = 25  
x\_midlle\_max = (x1max + x2max + x3max) / 3  
x\_midlle\_min = (x1min + x2min + x3min) / 3  
y\_max = 200 + x\_midlle\_max  
y\_min = 200 + x\_midlle\_min  
  
m, n, q = 3, 4, 0.5  
  
def mtrx(matrix):  
 for i in range(len(matrix)):  
 print("{}.".format(i + 1), end = "")  
 for j in range(len(matrix[i])):  
 print("{:4}".format(matrix[i][j]), end = "")  
 print()  
  
def main(m, n, q):  
 print("\nMatrix of code values")  
 codeX = np.array([[+1, -1, -1, -1], [+1, -1, +1, +1], [+1, +1, -1, +1], [+1, +1, +1, -1]])  
 mtrx(codeX)  
  
 print("\nX-matrix:")  
 x = np.array([[x1min, x2min, x3min], [x1min, x2max, x3max], [x1max, x2min, x3max], [x1max, x2max, x3min]])  
 mtrx(x)  
  
 print("\nY-matrix:")  
 y = np.random.randint(y\_min, y\_max, size = (n, m))  
 mtrx(y)  
  
 print("\nAverage of the response features:")  
 y\_midlle = np.sum(y, axis = 1) / len(y[0])  
 y\_1, y\_2, y\_3, y\_4 = y\_midlle  
 print(f"y\_1 = {y\_1:.2f}\ny\_2 = {y\_2:.2f}\ny\_3 = {y\_3:.2f}\ny\_4 = {y\_4:.2f}")  
 mx\_1, mx\_2, mx\_3 = [i / len(x) for i in np.sum(x, axis = 0)]  
 my = sum(y\_midlle) / len(y\_midlle)  
  
 a\_1 = sum([x[i][0] \* y\_midlle[i] for i in range(len(x))]) / len(x)  
 a\_2 = sum([x[i][1] \* y\_midlle[i] for i in range(len(x))]) / len(x)  
 a\_3 = sum([x[i][2] \* y\_midlle[i] for i in range(len(x))]) / len(x)  
  
 a\_11 = sum([x[i][0] \*\* 2 for i in range(len(x))]) / len(x)  
 a\_22 = sum([x[i][1] \*\* 2 for i in range(len(x))]) / len(x)  
 a\_33 = sum([x[i][2] \*\* 2 for i in range(len(x))]) / len(x)  
 a\_12 = sum([x[i][0] \* x[i][1] for i in range(len(x))]) / len(x)  
 a\_13 = sum([x[i][0] \* x[i][2] for i in range(len(x))]) / len(x)  
 a\_23 = a\_32 = sum([x[i][1] \* x[i][2] for i in range(len(x))]) / len(x)  
  
 det = np.linalg.det([[1, mx\_1, mx\_2, mx\_3], [mx\_1, a\_11, a\_12, a\_13], [mx\_2, a\_12, a\_22, a\_32], [mx\_3, a\_13, a\_23, a\_33]])  
 det\_0 = np.linalg.det([[my, mx\_1, mx\_2, mx\_3], [a\_1, a\_11, a\_12, a\_13], [a\_2, a\_12, a\_22, a\_32], [a\_3, a\_13, a\_23, a\_33]])  
 det\_1 = np.linalg.det([[1, my, mx\_2, mx\_3], [mx\_1, a\_1, a\_12, a\_13], [mx\_2, a\_2, a\_22, a\_32], [mx\_3, a\_3, a\_23, a\_33]])  
 det\_2 = np.linalg.det([[1, mx\_1, my, mx\_3], [mx\_1, a\_11, a\_1, a\_13], [mx\_2, a\_12, a\_2, a\_32], [mx\_3, a\_13, a\_3, a\_33]])  
 det\_3 = np.linalg.det([[1, mx\_1, mx\_2, my], [mx\_1, a\_11, a\_12, a\_1], [mx\_2, a\_12, a\_22, a\_2], [mx\_3, a\_13, a\_23, a\_3]])  
  
 b\_0 = det\_0 / det  
 b\_1 = det\_1 / det  
 b\_2 = det\_2 / det  
 b\_3 = det\_3 / det  
 b = [b\_0, b\_1, b\_2, b\_3]  
  
 print("\nThe normalized regression equation: y = {0} + {1} \* x1 + {2} \* x2 + {3} \* x3\n".format(round(b\_0, 5), round(b\_1, 5), round(b\_2, 5), round(b\_3, 5)))  
  
 print("Audit:")  
 y\_1\_exp = b\_0 + b\_1 \* x[0][0] + b\_2 \* x[0][1] + b\_3 \* x[0][2]  
 y\_2\_exp = b\_0 + b\_1 \* x[1][0] + b\_2 \* x[1][1] + b\_3 \* x[1][2]  
 y\_3\_exp = b\_0 + b\_1 \* x[2][0] + b\_2 \* x[2][1] + b\_3 \* x[2][2]  
 y\_4\_exp = b\_0 + b\_1 \* x[3][0] + b\_2 \* x[3][1] + b\_3 \* x[3][2]  
 print(f"y\_1 = {b\_0:.3f} + {b\_1:.3f} \* {x[0][0]} + {b\_2:.3f} \* {x[0][1]} + {b\_3:.3f} \* {x[0][2]} = {y\_1\_exp:.3f}"  
 f"\ny\_2 = {b\_0:.3f} + {b\_1:.3f} \* {x[1][0]} + {b\_2:.3f} \* {x[1][1]} + {b\_3:.3f} \* {x[1][2]} = {y\_2\_exp:.3f}"  
 f"\ny\_3 = {b\_0:.3f} + {b\_1:.3f} \* {x[2][0]} + {b\_2:.3f} \* {x[2][1]} + {b\_3:.3f} \* {x[2][2]} = {y\_3\_exp:.3f}"  
 f"\ny\_4 = {b\_0:.3f} + {b\_1:.3f} \* {x[3][0]} + {b\_2:.3f} \* {x[3][1]} + {b\_3:.3f} \* {x[3][2]} = {y\_4\_exp:.3f}")  
  
 print("\n[ Kohren's test ]")  
 f\_1 = m - 1  
 f\_2 = n  
 s\_1 = sum([(i - y\_1) \*\* 2 for i in y[0]]) / m  
 s\_2 = sum([(i - y\_2) \*\* 2 for i in y[1]]) / m  
 s\_3 = sum([(i - y\_3) \*\* 2 for i in y[2]]) / m  
 s\_4 = sum([(i - y\_4) \*\* 2 for i in y[3]]) / m  
 s\_array = np.array([s\_1, s\_2, s\_3, s\_4])  
 gP = max(s\_array) / sum(s\_array)  
  
 table = {3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7: 0.5365, 8: 0.5175, 9: 0.5017, 10: 0.4884, range(11, 17): 0.4366, range(17, 37): 0.3720, range(37, 145): 0.3093}  
 gT = table.get(m)  
  
 if(gP < gT):  
 print(f"The variance is homogeneous: Gp = {gP:.5} < Gt = {gT}")  
 else:  
 print(f"The variance is not homogeneous Gp = {gP:.5} < Gt = {gT}")  
 m = m + 1  
 main(m + 1, n, q)  
 return  
  
 print("\n[ Student's test ]")  
 s2\_B = s\_array.sum() / n  
 s2\_beta\_S = s2\_B / (n \* m)  
 s\_beta\_S = pow(s2\_beta\_S, 1/2)  
  
 beta\_0 = sum([codeX[i][0] \* y\_midlle[i] for i in range(len(codeX))]) / n  
 beta\_1 = sum([codeX[i][1] \* y\_midlle[i] for i in range(len(codeX))]) / n  
 beta\_2 = sum([codeX[i][2] \* y\_midlle[i] for i in range(len(codeX))]) / n  
 beta\_3 = sum([codeX[i][3] \* y\_midlle[i] for i in range(len(codeX))]) / n  
  
 t = [abs(beta\_0) / s\_beta\_S, abs(beta\_1) / s\_beta\_S, abs(beta\_2) / s\_beta\_S, abs(beta\_3) / s\_beta\_S ]  
  
 f3 = f\_1 \* f\_2  
 t\_table = {8: 2.306, 9: 2.262, 10: 2.228, 11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131, 16: 2.120, 17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086, 21: 2.08, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.06}  
 d = 4  
  
 for i in range(len(t)):  
 if(t\_table.get(f3) > t[i]):  
 b[i] = 0  
 d -= 1  
  
 print(f"Regression equation: y = {b[0]:.3f} + {b[1]:.3f} \* x1 + {b[2]:.3f} \* x2 + {b[3]:.3f} \* x3")  
 check\_0 = b[0] + b[1] \* x[0][0] + b[2] \* x[0][1] + b[3] \* x[0][2]  
 check\_1 = b[0] + b[1] \* x[1][0] + b[2] \* x[1][1] + b[3] \* x[1][2]  
 check\_2 = b[0] + b[1] \* x[2][0] + b[2] \* x[2][1] + b[3] \* x[2][2]  
 check\_3 = b[0] + b[1] \* x[3][0] + b[2] \* x[3][1] + b[3] \* x[3][2]  
 ckeck\_list = [check\_0, check\_1, check\_2, check\_3]  
 print("Values are normalized: ", ckeck\_list)  
  
 print("\n[ Fisher's test ]")  
 f\_4 = n - d  
 s2\_ad = m / f\_4 \* sum([(ckeck\_list[i] - y\_midlle[i]) \*\* 2 for i in range(len(y\_midlle))])  
 fP = s2\_ad / s2\_B  
 fT = [[164.4, 199.5, 215.7, 224.6, 230.2, 234], [18.5, 19.2, 19.2, 19.3, 19.3, 19.3],  
 [10.1, 9.6, 9.3, 9.1, 9, 8.9], [7.7, 6.9, 6.6, 6.4, 6.3, 6.2], [6.6, 5.8, 5.4, 5.2, 5.1, 5],  
 [6, 5.1, 4.8, 4.5, 4.4, 4.3], [5.5, 4.7, 4.4, 4.1, 4, 3.9], [5.3, 4.5, 4.1, 3.8, 3.7, 3.6],  
 [5.1, 4.3, 3.9, 3.6, 3.5, 3.4], [5, 4.1, 3.7, 3.5, 3.3, 3.2], [4.8, 4, 3.6, 3.4, 3.2, 3.1],  
 [4.8, 3.9, 3.5, 3.3, 3.1, 3], [4.7, 3.8, 3.4, 3.2, 3, 2.9], [4.6, 3.7, 3.3, 3.1, 3, 2.9],  
 [4.5, 3.7, 3.3, 3.1, 2.9, 2.8], [4.5, 3.6, 3.2, 3, 2.9, 2.7], [4.5, 3.6, 3.2, 3, 2.8, 2.7],  
 [4.4, 3.6, 3.2, 2.9, 2.8, 2.7], [4.4, 3.5, 3.1, 2.9, 2.7, 2.6], [4.4, 3.5, 3.1, 2.9, 2.7, 2.6]]  
 if(fP > fT[f3][f\_4]):  
 print(f"fp = {fP} > ft = {fT[f3][f\_4]}.\nThe mathematical model is not adequate to the experimental data\n")  
 else:  
 print(f"fP = {fP} < fT = {fT}.\nThe mathematical model is adequate to the experimental data\n")  
  
print("\nRegression equation --- y = b\_0 + b\_1 \* x1 + b\_1 \* x2 +b\_3 \* x3")  
main(m, n, q)