­­­Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №5

**«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНАМИ»**

Виконав:

студент групи ІВ-81

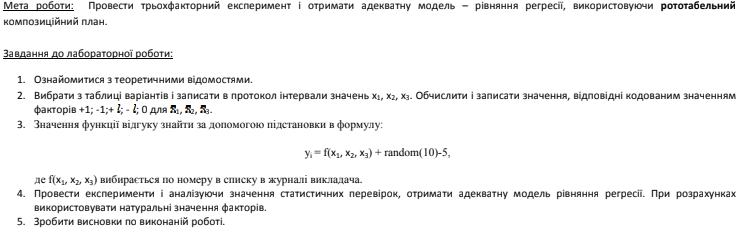
Богданенко В.Ю.

Перевірив:

ас. Регіда П. Г.

Київ

2020 р.



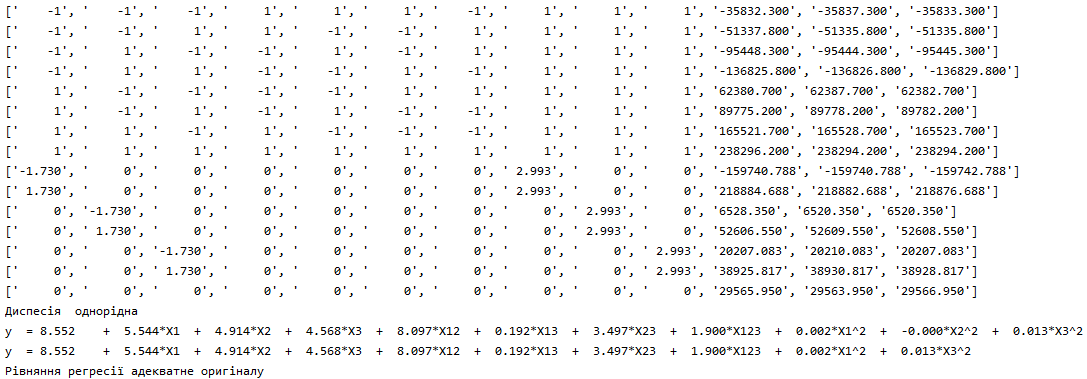
**Варіант завдання:**



**Лістинг програми:**

**import** numpy **as** np  
**import** random  
**from** copy **import** deepcopy  
**from** math **import** sqrt  
**from** scipy.stats **import** f  
**from** functools **import** partial  
**def** lab6():  
 M = 3  
 X1min = -20  
 X1max = 30  
 X2min = 30  
 X2max = 80  
 X3min = 30  
 X3max = 45  
 Matrixplan = [[-1, -1, -1],  
 [-1, -1, 1],  
 [-1, 1, -1],  
 [-1, 1, 1],  
 [1, -1, -1],  
 [1, -1, 1],  
 [1, 1, -1],  
 [1, 1, 1],  
 [-1.73, 0, 0],  
 [1.73, 0, 0],  
 [0, -1.73, 0],  
 [0, 1.73, 0],  
 [0, 0, -1.73],  
 [0, 0, 1.73],  
 [0, 0, 0]]  
 **def** avto\_fill\_matrix(a):  
 **for** i **in** range(len(a)):  
 a[i].append(a[i][0] \* a[i][1])  
 a[i].append(a[i][0] \* a[i][2])  
 a[i].append(a[i][1] \* a[i][2])  
 a[i].append(a[i][0] \* a[i][1] \* a[i][2])  
 a[i].append(a[i][0] \*\* 2)  
 a[i].append(a[i][1] \*\* 2)  
 a[i].append(a[i][2] \*\* 2)  
 **return** a  
 **def** cohren(f1, f2, q=0.05):  
 q1 = q / f1  
 fisher\_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) \* f2)  
 **return** fisher\_value / (fisher\_value + f1 - 1)  
 **def** fill\_matrix(a, x):  
 a1 = []  
 **for** i **in** range(len(a)):  
 a1.append([])  
 **for** j **in** range(3):  
 a1[i].append(0)  
 **for** i **in** range(len(a)):  
 **for** j **in** range(3):  
 **if** a[i][j] == -1:  
 a1[i][j] = (min(x[j]))  
 **elif** a[i][j] == 1:  
 a1[i][j] = (max(x[j]))  
 **else**:  
 a1[i][j] = (x[j][0] + x[j][1]) / 2 + a[i][j] \* (x[j][1] - ((x[j][0] + x[j][1]) / 2))  
 avto\_fill\_matrix(a1)  
 **return** a1  
 **def** lab(m, plan, natural, ymax, ymin):  
 ysplist = []  
 S2ylist = []  
 S2ysum = 0  
 rl = []  
 yklist = []  
 blist = []  
 detlist = []  
 tlist = []  
 sumt = 0  
 bultlist = []  
 ynewlist = []  
 Sad = 0  
 xlist = [**" "**, **"\*X1"**, **"\*X2"**, **"\*X3"**, **"\*X12"**, **"\*X13"**, **"\*X23"**, **"\*X123"**, **"\*X1^2"**, **"\*X2^2"**, **"\*X3^2"**]  
 text3 = **"y = "** text4 = **"y = "** Gt = cohren(m - 1, 15)  
 **for** j **in** range(len(plan)):  
 **for** i **in** range(len(plan[14]), m + 10):  
 natural[j].append(random.randint(0, 10) - 5 + 0.7 + 5.4 \* natural[j][0] + 4.8 \* natural[j][1] + 5.3 \* natural[j][2] + 8.1 \* natural[j][3] + 0.2 \* natural[j][4] + 3.5 \* natural[j][5] + 1.9 \* natural[j][6])  
 plan[j].append(random.randint(0, 10) - 5 + 0.7 + 5.4 \* natural[j][0] + 4.8 \* natural[j][1] + 5.3 \* natural[j][2] + 8.1 \*natural[j][3] + 0.2 \* natural[j][4] + 3.5 \* natural[j][5] + 1.9 \* natural[j][6])  
 **for** i **in** range(len(plan)):  
 ysp = 0  
 **for** j **in** range(10, len(plan[0])):  
 ysp = ysp + plan[i][j]  
 ysp = ysp / m  
 ysplist.append(ysp)  
 **for** i **in** range(len(plan)):  
 S2y = 0  
 **for** j **in** range(10, len(plan[0])):  
 S2y = S2y + (plan[i][j] - ysplist[i]) \*\* 2  
 S2y = S2y / m  
 S2ylist.append(S2y)  
 S2ysum = S2ysum + S2y  
 Gp = max(S2ylist) / S2ysum  
 **if** Gp > Gt:  
 m = m + 1  
 lab((m, plan, natural, ymax, ymin))  
 **else**:  
 deepcool\_natural = deepcopy(natural)  
 **for** i **in** range(len(deepcool\_natural)):  
 deepcool\_natural[i].insert(0, 1)  
 **for** z **in** range(11):  
 k0l = []  
 **for** u **in** range(11):  
 k0 = 0  
 **for** i **in** range(15):  
 k0 = k0 + deepcool\_natural[i][z] \* deepcool\_natural[i][u]  
 k0 = k0  
 k0l.append(k0)  
 rl.append(k0l)  
 det0 = np.linalg.det(rl)  
 **for** j **in** range(11):  
 yk = 0  
 **for** i **in** range(15):  
 yk = yk + ysplist[i] \* deepcool\_natural[i][j]  
 yklist.append(yk)  
 **for** j **in** range(11):  
 v = deepcopy(rl)  
 **for** i **in** range(11):  
 v[i][j] = yklist[i]  
 detlist.append(np.linalg.det(v))  
 **for** i **in** range(len(detlist)):  
 blist.append(detlist[i] / det0)  
 S2B = S2ysum / 15  
 S2b = S2B / (15 \* m)  
 Sb = sqrt(S2b)  
 plan1 = deepcopy(plan)  
 **for** i **in** range(len(plan1)):  
 plan1[i].insert(0, 1)  
 rl = []  
 **for** z **in** range(11):  
 k0l = []  
 **for** u **in** range(11):  
 k0 = 0  
 **for** i **in** range(15):  
 k0 = k0 + plan1[i][z] \* plan1[i][u]  
 k0 = k0  
 k0l.append(k0)  
 rl.append(k0l)  
 det0 = np.linalg.det(rl)  
 yklist = []  
 **for** j **in** range(11):  
 yk = 0  
 **for** i **in** range(15):  
 yk = yk + ysplist[i] \* plan1[i][j]  
 yklist.append(yk)  
 detlist = []  
 **for** j **in** range(11):  
 v = deepcopy(rl)  
 **for** i **in** range(11):  
 v[i][j] = yklist[i]  
 detlist.append(np.linalg.det(v))  
 **for** i **in** range(len(detlist)):  
 tlist.append(abs(detlist[i] / det0) / Sb)  
 **for** i **in** range(len(tlist)):  
 **if** tlist[i] >= 2.042:  
 bultlist.append(1)  
 sumt = sumt + 1  
 **elif** tlist[i] < 2.042:  
 bultlist.append(0)  
 **for** j **in** range(15):  
 ynew = 0  
 **for** i **in** range(11):  
 **if** bultlist[i] == 1:  
 ynew = ynew + blist[i] \* deepcool\_natural[j][i]  
 ynewlist.append(ynew)  
 **for** i **in** range(15):  
 Sad = Sad + ((ynewlist[i] - ysplist[i]) \*\* 2) \* m / (15 - sumt)  
 Fp = Sad / S2B  
 **for** i **in** range(len(plan)):  
 **for** j **in** range(len(plan[i])):  
 **if** type(plan[i][j]) == float:  
 **if** plan[i][j] != 0:  
 plan[i][j] = **'%.3f'** % plan[i][j]  
 **if** (plan[i][j] == 0.0 **or** plan[i][j] == -0.0):  
 plan[i][j] = 0  
 plan[i][j] = (**'%+6s'** % plan[i][j])  
 print(plan[i])  
 blist1 = [str(**'%.3f'** % blist[0]), **" + "** + str(**'%.3f'** % blist[1]), **" + "** + str(**'%.3f'** % blist[2]),  
 **" + "** + str(**'%.3f'** % blist[3]), **" + "** + str(**'%.3f'** % blist[4]), **" + "** + str(**'%.3f'** % blist[5]),  
 **" + "** + str(**'%.3f'** % blist[6]), **" + "** + str(**'%.3f'** % blist[7]), **" + "** + str(**'%.3f'** % blist[8]),  
 **" + "** + str(**'%.3f'** % blist[9]), **" + "** + str(**'%.3f'** % blist[10]), ]  
 **for** i **in** range(len(xlist)):  
 text3 = text3 + (blist1[i]) + xlist[i]  
 **for** i **in** range(len(xlist)):  
 **if** bultlist[i] == 1:  
 text4 = text4 + (blist1[i]) + xlist[i]  
 f4 = 15 - sumt  
 f3 = (m - 1) \* 15  
 fisher = partial(f.ppf, q=1 - 0.05)  
 Ft = fisher(dfn=f4, dfd=f3)  
 **if** Fp < Ft:  
 print(**"Диспесія однорідна"**)  
 print(text3)  
 print(text4)  
 print(**"Рівняння регресії адекватне оригіналу"**)  
 **elif** Fp > Ft:  
 print(**"Диспесія однорідна"**)  
 print(text3)  
 print(text4)  
 print(**"Рівняння регресії неадекватне оригіналу"**)  
 **def** last(X1min, X1max, X2min, X2max, X3min, X3max, M, Matrixplan):  
 xmatrix = [[X1min, X1max], [X2min, X2max], [X3min, X3max]]  
 ymax = 200 + (X3max + X2max + X1max) / 3  
 ymin = 200 + (X3min + X2min + X1min) / 3  
 matrixplan1 = avto\_fill\_matrix(Matrixplan)  
 matrixnatural = fill\_matrix(matrixplan1, xmatrix)  
 lab(M, matrixplan1, matrixnatural, ymax, ymin)  
 last(X1min, X1max, X2min, X2max, X3min, X3max, M, Matrixplan)  
lab6()

**Результат виконання роботи:**



­­­