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 оптимізації та планування експерименту» на тему

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ »

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІВ-81

Мисак Олександр

Варіант: 118

ПЕРЕВІРИВ:

Регіда П. Г.

Київ – 2020

**Код програми**

import math

import numpy as np

from numpy.linalg import solve

from scipy.stats import f, t

from functools import partial

from random import randint

from prettytable import PrettyTable

# testing functions

def cochrane(g\_prac, g\_teor):

return g\_prac < g\_teor

def student(t\_teor, t\_pr):

return t\_pr < t\_teor

def fischer(f\_teor, f\_prac):

return f\_teor > f\_prac

def cochrane\_teor(f1, f2, q=0.05):

q1 = q / f1

fischer\_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) \* f2)

return fischer\_value / (fischer\_value + f1 - 1)

fischer\_teor = partial(f.ppf, q=1 - 0.05)

student\_teor = partial(t.ppf, q=1 - 0.025)

X1min = 20

X1max = 70

X2min = 25

X2max = 65

X3min = 25

X3max = 35

Xmin\_average = (X1min + X2min + X3min) / 3 # Xcp(min)

Xmax\_average = (X1max + X2max + X3max) / 3 # Xcp(max)

y\_max = round(200 + Xmax\_average)

y\_min = round(200 + Xmin\_average)

while True:

# matrix

x0\_factor = [1, 1, 1, 1, 1, 1, 1, 1]

x1\_factor = [-1, -1, 1, 1, -1, -1, 1, 1]

x2\_factor = [-1, 1, -1, 1, -1, 1, -1, 1]

x3\_factor = [-1, 1, 1, -1, 1, -1, -1, 1]

x1x2\_factor = [a \* b for a, b in zip(x1\_factor, x2\_factor)]

x1x3\_factor = [a \* b for a, b in zip(x1\_factor, x3\_factor)]

x2x3\_factor = [a \* b for a, b in zip(x2\_factor, x3\_factor)]

x1x2x3\_factor = [a \* b \* c for a, b, c in zip(x1\_factor, x2\_factor, x3\_factor)]

factors = [x0\_factor, x1\_factor, x2\_factor, x3\_factor, x1x2\_factor, x1x3\_factor, x2x3\_factor, x1x2x3\_factor]

m = 3 # repeat qty

y1, y2, y3 = [randint(y\_min, y\_max) for i in range(8)], [randint(y\_min, y\_max) for i in range(8)], [randint(y\_min, y\_max) for i in range(8)] # filling y

# table rows (y values)

y\_dict = {}

for x in range(1, 9):

y\_dict["y\_row{0}".format(x)] = [y1[x - 1], y2[x - 1], y3[x - 1]]

# averages for each y-row

y\_avg\_dict = {}

for x in range(1, 9):

y\_avg\_dict["y\_avg{0}".format(x)] = np.average(y\_dict[f'y\_row{x}'])

Y\_average = [round(val, 3) for val in y\_avg\_dict.values()]

# reinitializing arrs + swapping to min/max values

x0 = [1, 1, 1, 1, 1, 1, 1, 1]

x1 = [X1min, X1min, X1max, X1max, X1min, X1min, X1max, X1max]

x2 = [X2min, X2max, X2min, X2max, X2min, X2max, X2min, X2max]

x3 = [X3min, X3max, X3max, X3min, X3max, X3min, X3min, X3max]

x1x2 = [a \* b for a, b in zip(x1, x2)]

x1x3 = [a \* b for a, b in zip(x1, x3)]

x2x3 = [a \* b for a, b in zip(x2, x3)]

x1x2x3 = [a \* b \* c for a, b, c in zip(x1, x2, x3)]

x\_arr = [x0, x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3]

b\_list = factors

a\_list = list(zip(x0, x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3))

N = 8 # test repeat quantity

list\_bi = [] # b(i)

for k in range(N):

S = 0

for i in range(N):

S += (b\_list[k][i] \* Y\_average[i]) / N

list\_bi.append(round(S, 5))

disp\_dict = {} # dispersions

for x in range(1, 9):

disp\_dict["disp{0}".format(x)] = 0

for i in range(m):

ctr = 1

for key, value in disp\_dict.items():

row = y\_dict[f'y\_row{ctr}']

disp\_dict[key] += ((row[i] - np.average(row)) \*\* 2) / m

ctr += 1

disp\_sum = sum(disp\_dict.values())

disp\_list = [round(disp, 3) for disp in disp\_dict.values()]

column\_names = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", "Y1", "Y2", "Y3", "Y", "S^2"] # назви

pt = PrettyTable()

factors.extend([y1, y2, y3, Y\_average, disp\_list])

for k in range(len(factors)):

pt.add\_column(column\_names[k], factors[k])

print(pt, "\n")

# Regression eq with interaction effect

print("y = {} + {}\*x1 + {}\*x2 + {}\*x3 + {}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3 \n".format(list\_bi[0], list\_bi[1],

list\_bi[2], list\_bi[3],

list\_bi[4], list\_bi[5],

list\_bi[6], list\_bi[7]))

pt = PrettyTable()

x\_arr.extend([y1, y2, y3, Y\_average, disp\_list])

for k in range(len(factors)):

pt.add\_column(column\_names[k], x\_arr[k])

print(pt, "\n")

list\_ai = [round(i, 5) for i in solve(a\_list, Y\_average)]

print("y = {} + {}\*x1 + {}\*x2 + {}\*x3 + {}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3".format(list\_ai[0], list\_ai[1],

list\_ai[2], list\_ai[3],

list\_ai[4], list\_ai[5],

list\_ai[6], list\_ai[7]))

Gp = max(disp\_dict.values()) / disp\_sum # experimental

F1 = m - 1

N = len(y1)

F2 = N

Gt = cochrane\_teor(F1, F2) # theoretical

print("\nGp = ", Gp, " Gt = ", Gt)

if cochrane(Gp, Gt):

print("Dispresion is homogeneous!\n")

Dispersion\_B = disp\_sum / N

Dispersion\_beta = Dispersion\_B / (m \* N)

S\_beta = math.sqrt(abs(Dispersion\_beta))

beta\_dict = {} # beta values

for x in range(8):

beta\_dict["beta{0}".format(x)] = 0

for i in range(len(x0\_factor)):

ctr = 0

for key, value in beta\_dict.items():

beta\_dict[key] += (Y\_average[i] \* factors[ctr][i]) / N

ctr += 1

beta\_list = list(beta\_dict.values())

t\_list = [abs(beta) / S\_beta for beta in beta\_list]

F3 = F1 \* F2

d = 0

T = student\_teor(df=F3)

print("t table = ", T)

for i in range(len(t\_list)):

if student(t\_list[i], T):

beta\_list[i] = 0

print("Hypothesis has been confirmed, beta{} = 0".format(i))

else:

print("Hypothesis was not confirmed.\nbeta{} = {}".format(i, beta\_list[i]))

d += 1

# y\_student = [beta\_list[0] + beta\_list[1] \* x1[i] + beta\_list[2] \* x2[i] + beta\_list[3] \* x3[i] + beta\_list[4] \* x1x2[i] + beta\_list[5] \* x1x3[i] + beta\_list[6] \* x2x3[i] + beta\_list[7] \* x1x2x3[i] for i in range(8)]

# Optimizing the value list generation above

factors[0] = None

y\_student = [sum([a \* b[x\_idx] if b else a for a, b in zip(beta\_list, x\_arr)]) for x\_idx in range(8)]

F4 = N - d

Dispersion\_ad = 0

for i in range(len(y\_student)):

Dispersion\_ad += ((y\_student[i] - Y\_average[i]) \*\* 2) \* m / (N - d)

Fp = Dispersion\_ad / Dispersion\_beta

Ft = fischer\_teor(dfn=F4, dfd=F3)

if fischer(Ft, Fp):

print("Regression equation is adequate")

break

else:

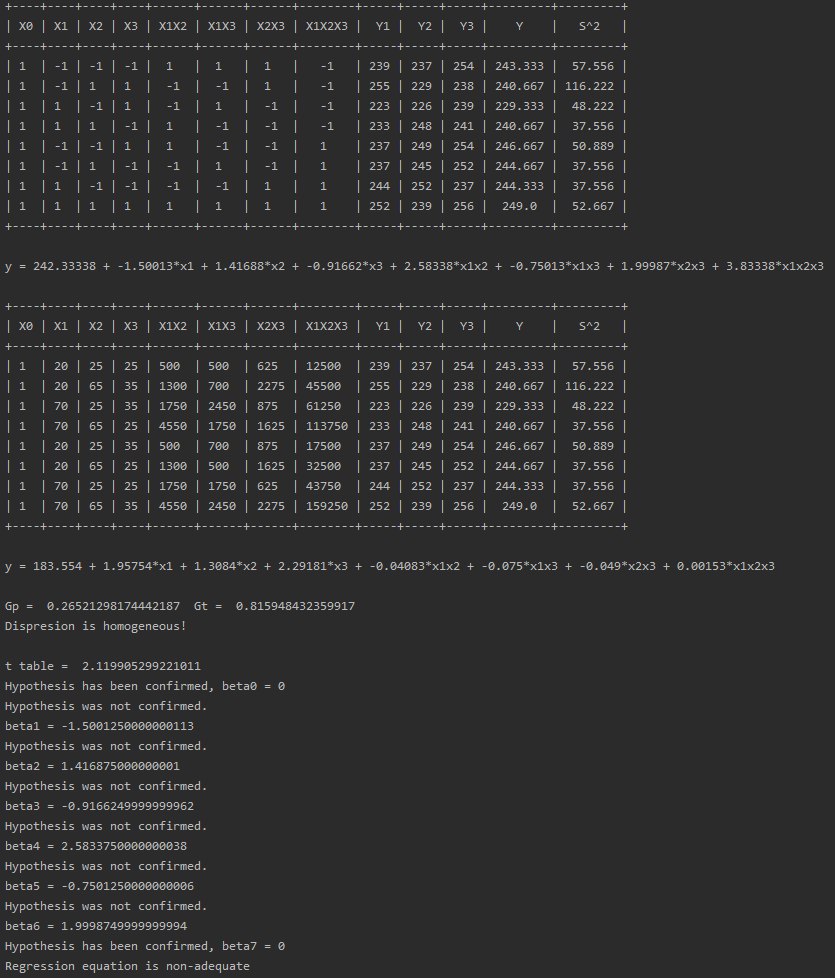
print("Regression equation is non-adequate")

break

else:

print("Dispersion is not homogeneous.")

m += 1

****