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»

Виконав:

студент ІІ курсу ФІОТ

групи ІО-82

Рикун Олег Віталійович

Залікова книжка № 8218

Варіант: 216

Перевірив:

ст. вик.

Регіда П. Г.

Київ – 2020

Код програми:

import math

from scipy.stats import f, t

from functools import partial

from random import randint

import numpy as np

from prettytable import PrettyTable

from numpy.linalg import solve

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

return g\_prac < g\_teor

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

return t\_pr < t\_teor

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

return f\_teor > f\_prac

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

q1 = q / f1

fisher\_t = partial(f.ppf, q=1 - 0.05)

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

x1min = 10

x1max = 50

x2min = -20

x2max = 60

x3min = 10

x3max = 15

xmin = (x1min + x2min + x3min)/3

xmax = (x1max + x2max + x3max)/3

ymin = round(200 + xmin)

ymax = round(200 + xmax)

def lab4(n, m):

global vur

while True:

x0f = [1, 1, 1, 1, 1, 1, 1, 1]

x1f = [-1, -1, -1, -1, 1, 1, 1, 1]

x2f = [-1, -1, 1, 1, -1, -1, 1, 1]

x3f = [-1, 1, -1, 1, -1, 1, -1, 1]

x12f = [i\*j for i, j in zip(x1f, x2f)]

x13f = [i\*j for i, j in zip(x1f, x3f)]

x23f = [i\*j for i, j in zip(x2f, x3f)]

x123f = [i\*j\*k for i, j, k in zip(x1f, x2f, x3f)]

list\_fact = [x0f, x1f, x2f, x3f, x12f, x13f, x23f, x123f]

y1 = [randint(ymin, ymax) for i in range(n)]

y2 = [randint(ymin, ymax) for i in range(n)]

y3 = [randint(ymin, ymax) for i in range(n)]

y\_rows = {}

for i in range(1, 9):

y\_rows["Рядок{0}".format(i)] = [y1[i-1], y2[i-1], y3[i-1]]

y\_row\_ser = {}

for i in range(1, 9):

y\_row\_ser["Середнє значення Y у рядку{0}".format(i)] = np.average(y\_rows[f"Рядок{i}"])

y\_ser = [round(val, 3) for val in y\_row\_ser.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]

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

list\_bi = []

for k in range(n):

S = 0

for i in range(n):

S += (list\_fact[k][i] \* y\_ser[i]) / n

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

disp = {}

for i in range(1, n+1):

disp["Дисперсія{0}".format(i)] = 0

for i in range(m):

ctr = 1

for key, value in disp.items():

row = y\_rows[f'Рядок{ctr}']

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

ctr += 1

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

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

column = ["x0", "x1", "x2", "x3", "x1x2", "x1x3", "x2x3", "x1x2x3", "y1", "y2", "y3", "y", "s^2"]

pt = PrettyTable()

list\_fact.extend([y1, y2, y3, y\_ser, disp\_list])

for k in range(len(list\_fact)):

pt.add\_column(column[k], list\_fact[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\_ser, disp\_list])

for k in range(len(list\_fact)):

pt.add\_column(column[k], list\_fact[k])

print(pt, "\n")

list\_ai = [round(i, 5) for i in solve(list\_x, y\_ser)]

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.values()) / disp\_sum

f1 = m -1

f2 = n

gt = cochrane\_teor(f1, f2)

if cochrane(gp, gt):

print("Дисперсія однорідна!\n")

dispersion\_b = disp\_sum / n

dispersion\_beta = dispersion\_b / (m \* n)

s\_beta = math.sqrt(abs(dispersion\_beta))

beta = {} # beta values

for x in range(8):

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

for i in range(len(x0f)):

ctr = 0

for key, value in beta.items():

beta[key] += (y\_ser[i] \* list\_fact[ctr][i]) / n

ctr += 1

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

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

f3 = f1 \* f2

d = 0

t = student\_t(df=f3)

print("t = ", t)

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

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

beta\_list[i] = 0

print("Гіпотеза підтвердилась, beta{} = 0".format(i))

else:

print("Гіпотеза не підтвердилась, beta{} = {}".format(i, beta\_list[i]))

d += 1

list\_fact[0] = None

y\_student = [sum([a \* b[x\_idx] if b else a for a, b in zip(beta\_list, list\_x)]) 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\_ser[i]) \*\* 2) \* m / (n - d)

fp = dispersion\_ad / dispersion\_beta

ft = fisher\_t(dfn=f4, dfd=f3)

if fisher(ft, fp):

vur = "Рівняння регресії адекватне"

print(vur)

break

else:

vur = "Рівняння регресії не адекватне"

print(vur)

break

else:

print("дисперсія не однорідна")

m += 1

n = 8

m = 3

vur = ""

lab4(n, m)

# цей цикл потрібний для повтору виклику функції, якщо рівнянн регресії не адекватно

# таких повторів в даній програмі десять, щоб не було зациклення

p = 0

while p <= 10:

if vur == "Рівняння регресії не адекватне":

lab4(n, m)

p += 1