Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки

Методи оптимізації та планування

Лабораторна робота №3

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З

ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконала:

студентка групи ІО-82

Халіл Хана

Залікова книжка № 8225

Номер у списку групи 23

Перевірив ас. Регіда П. Г.

Київ 2020 р.

**Лістинг програми**

import random

import numpy

import tkinter

import tkinter.messagebox

root = tkinter.Tk()

x1\_min = -30

x1\_max = 0

x2\_min = -15

x2\_max = 35

x3\_min = -30

x3\_max = -25

xm\_min = (x1\_min + x2\_min + x3\_min) / 3

xm\_max = (x1\_max + x2\_max + x3\_max) / 3

y\_min = 200 + xm\_min

y\_max = 200 + xm\_max

gt = {1: 0.9065, 2: 0.7679, 3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7:0.5365, 8: 0.5175, 9: 0.5017, 10: 0.4884}

tt = {4: 2.776, 8: 2.306, 12: 2.179, 16: 2.120, 20: 2.086, 24: 2.064, 28: 2.048}

ft = {1: {4: 7.7, 8: 5.3, 12: 4.8, 16: 4.5, 20: 4.4, 24: 4.3, 28: 4.2},

2: {4: 6.9, 8: 4.5, 12: 3.9, 16: 3.6, 20: 3.5, 24: 3.4, 28: 3.3},

3: {4: 6.6, 8: 4.1, 12: 3.5, 16: 3.2, 20: 3.1, 24: 3.0, 28: 3.0},

4: {4: 6.4, 8: 3.8, 12: 3.3, 16: 3.0, 20: 2.9, 24: 2.8, 28: 2.7},

5: {4: 6.3, 8: 3.7, 12: 3.1, 16: 2.9, 20: 2.7, 24: 2.6, 28: 2.6},

6: {4: 6.2, 8: 3.6, 12: 3.0, 16: 2.7, 20: 2.6, 24: 2.5, 28: 2.4}}

xn = [[-1, -1, -1],

[-1, 1, 1],

[1, -1, 1],

[1, 1, -1]]

x = [[-30, -15, -30],

[-30, 35, -25],

[0, -15, -25],

[0, 35, -30]]

m = 2

y = [[random.randint(int(y\_min), int(y\_max)) for i in range(m)] for j in range(4)]

def kohren(dispersion, m, gt):

gp = max(dispersion) / sum(dispersion)

return gp < gt[m - 1]

def student(dispersion\_reproduction, m, y\_mean, xn):

dispersion\_statistic\_mark = (dispersion\_reproduction / (4 \* m)) \*\* 0.5

beta = [1 / 4 \* sum(y\_mean[j] for j in range(4))]

for i in range(3):

b = 0

for j in range(4):

b += y\_mean[j] \* xn[j][i]

beta.append(1 / 4 \* b)

t = []

for i in beta:

t.append(abs(i) / dispersion\_statistic\_mark)

return t[0] > tt[(m - 1) \* 4], t[1] > tt[(m - 1) \* 4], t[2] > tt[(m - 1)\*4], t[3] > tt[(m - 1) \* 4]

def normalized\_multiplier(x, y\_mean):

mx1 = (x[0][0] + x[1][0] + x[2][0] + x[3][0]) / 4

mx2 = (x[0][1] + x[1][1] + x[2][1] + x[3][1]) / 4

mx3 = (x[0][2] + x[1][2] + x[2][2] + x[3][2]) / 4

my = sum(y\_mean) / 4

a11 = (x[0][0] \*\* 2 + x[1][0] \*\* 2 + x[2][0] \*\* 2 + x[3][0] \*\* 2) / 4

a22 = (x[0][1] \*\* 2 + x[1][1] \*\* 2 + x[2][1] \*\* 2 + x[3][1] \*\* 2) / 4

a33 = (x[0][2] \*\* 2 + x[1][2] \*\* 2 + x[2][2] \*\* 2 + x[3][2] \*\* 2) / 4

a12 = (x[0][0] \* x[0][1] + x[1][0] \* x[1][1] + x[2][0] \* x[2][1] + x[3][0] \* x[3][1]) / 4

a13 = (x[0][0] \* x[0][2] + x[1][0] \* x[1][2] + x[2][0] \* x[2][2] + x[3][0] \* x[3][2]) / 4

a23 = (x[0][1] \* x[0][2] + x[1][1] \* x[1][2] + x[2][1] \* x[2][2] + x[3][1] \* x[3][2]) / 4

a1 = (x[0][0] \* y\_mean[0] + x[1][0] \* y\_mean[1] + x[2][0] \* y\_mean[2] + x[3][0] \* y\_mean[3]) / 4

a2 = (x[0][1] \* y\_mean[0] + x[1][1] \* y\_mean[1] + x[2][1] \* y\_mean[2] + x[3][1] \* y\_mean[3]) / 4

a3 = (x[0][2] \* y\_mean[0] + x[1][2] \* y\_mean[1] + x[2][2] \* y\_mean[2] + x[3][2] \* y\_mean[3]) / 4

a = numpy.array([[1, mx1, mx2, mx3],

[mx1, a11, a12, a13],

[mx2, a12, a22, a23],

[mx3, a13, a23, a33]])

c = numpy.array([my, a1, a2, a3])

b = numpy.linalg.solve(a, c)

return b

def fisher(m, d, y\_mean, yo, dispersion\_reproduction, ft):

dispersion\_ad = 0

for i in range(4):

dispersion\_ad += (yo[i] - y\_mean[i]) \*\* 2

dispersion\_ad = dispersion\_ad \* m / (4 - d)

fp = dispersion\_ad / dispersion\_reproduction

return fp < ft[4 - d][(m - 1) \* 4]

while True:

while True:

y\_mean = []

for i in range(4):

y\_mean.append(sum(y[i]) / m)

dispersion = []

for i in range(len(y)):

dispersion.append(0)

for j in range(m):

dispersion[i] += (y\_mean[i] - y[i][j]) \*\* 2

dispersion[i] /= m

dispersion\_reproduction = sum(dispersion) / 4

if kohren(dispersion, m, gt):

break

else:

m += 1

for i in range(4):

y[i].append(random.randint(int(y\_min), int(y\_max)))

k = student(dispersion\_reproduction, m, y\_mean, xn)

d = sum(k)

b = normalized\_multiplier(x, y\_mean)

b = [b[i] \* k[i] for i in range(4)]

yo = []

for i in range(4):

yo.append(b[0] + b[1] \* x[i][0] + b[2] \* x[i][1] + b[3] \* x[i][2])

if d == 4:

m += 1

for i in range(4):

y[i].append(random.randint(int(y\_min), int(y\_max)))

elif fisher(m, d, y\_mean, yo, dispersion\_reproduction, ft):

break

else:

m += 1

for i in range(4):

y[i].append(random.randint(int(y\_min), int(y\_max)))

tkinter.Label(text="x1").grid()

tkinter.Label(text="x2").grid(row=0, column=1)

tkinter.Label(text="x3").grid(row=0, column=2)

for i in range(m):

tkinter.Label(text="yi" + str(i + 1)).grid(row=0, column=i + 3)

for i in range(len(x)):

for j in range(len(x[i])):

tkinter.Label(text=x[i][j]).grid(row=i + 1, column=j)

for i in range(len(y)):

for j in range(len(y[i])):

tkinter.Label(text=(y[i][j])).grid(row=i + 1, column=j + 3)

tkinter.Label(text="Рівняння регресії:").grid(columnspan=m + 3)

text = "y = " + "{0:.2f}".format(b[0])

for i in range(3):

if b[i + 1] != 0:

text = text + " + {0:.2f}".format(b[i + 1]) + " \* x" + str(i + 1)

tkinter.Label(text=text).grid(columnspan=m + 3)

tkinter.Label(text="Перевірка:").grid(columnspan=m + 3)

for i in range(4):

tkinter.Label(text="yc" + str(i + 1) + " =" + "{0:.2f}".format(y\_mean[i])).grid(columnspan=m + 3)

tkinter.Label(text="y" + str(i + 1) + " = " + "{0:.2f}".format(yo[i])).grid(columnspan=m + 3)

root.mainloop()

**Результат роботи програми**

