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

Методи оптимізації та планування

Лабораторна робота №4

**«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту**

**взаємодії.»**

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

from scipy.stats import t,f

#create new y array

def new\_y(n, m, ymin, ymax):

y = [[random.randint(ymin, ymax) for i in range(m)] for k in range(n)]

return y

#add y column

def append\_y(n, y, ymin, ymax):

for i in y:

i.append(random.randint(ymin, ymax))

def my(ymat):

ymmat = [sum(i)/3 for i in ymat]

return ymmat

def get\_b\_norm(xnorm, ym):

n = len(ym)

b = [0 for i in range(n)]

b[0] = sum(ym) / n

for i in range(n):

b[1] += ym[i] \* xnorm[i][1] / n

b[2] += ym[i] \* xnorm[i][2] / n

b[3] += ym[i] \* xnorm[i][3] / n

if n == 8:

b[4] += ym[i] \* xnorm[i][1] \* xnorm[i][2] / n

b[5] += ym[i] \* xnorm[i][1] \* xnorm[i][3] / n

b[6] += ym[i] \* xnorm[i][2] \* xnorm[i][3] / n

b[7] += ym[i] \* xnorm[i][1] \* xnorm[i][2] \* xnorm[i][3] / n

return b

def get\_b\_nat(m, xnat, ym):

n = len(ym)

def ai1(x, k):

a = [0 for i in range(n)]

for i in range(n):

a[0] += x[i][k]

a[1] += xnat[i][0] \* x[i][k]

a[2] += xnat[i][1] \* x[i][k]

a[3] += xnat[i][2] \* x[i][k]

if n == 8:

a[4] += xnat[i][0] \* xnat[i][1] \* x[i][k]

a[5] += xnat[i][0] \* xnat[i][2] \* x[i][k]

a[6] += xnat[i][1] \* xnat[i][2] \* x[i][k]

a[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2] \* x[i][k]

return a

def ai2(x, k, l):

a = [0 for i in range(n)]

for i in range(n):

a[0] += x[i][k] \* x[i][l]

a[1] += xnat[i][0] \* x[i][k] \* x[i][l]

a[2] += xnat[i][1] \* x[i][k] \* x[i][l]

a[3] += xnat[i][2] \* x[i][k] \* x[i][l]

if n == 8:

a[4] += xnat[i][0] \* xnat[i][1] \* x[i][k] \* x[i][l]

a[5] += xnat[i][0] \* xnat[i][2] \* x[i][k] \* x[i][l]

a[6] += xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l]

a[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l]

return a

def ai3(x, k, l, m):

a = [0 for i in range(n)]

for i in range(n):

a[0] += x[i][k] \* x[i][l] \* x[i][m]

a[1] += xnat[i][0] \* x[i][k] \* x[i][l] \* x[i][m]

a[2] += xnat[i][1] \* x[i][k] \* x[i][l] \* x[i][m]

a[3] += xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]

if n == 8:

a[4] += xnat[i][0] \* xnat[i][1] \* x[i][k] \* x[i][l] \* x[i][m]

a[5] += xnat[i][0] \* xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]

a[6] += xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]

a[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2] \* x[i][k] \* x[i][l] \* x[i][m]

return a

a = []

a1 = [0 for i in range(n)]

a1[0] = n

for i in range(n):

a1[1] += xnat[i][0]

a1[2] += xnat[i][1]

a1[3] += xnat[i][2]

if n == 8:

a1[4] += xnat[i][0] \* xnat[i][1]

a1[5] += xnat[i][0] \* xnat[i][2]

a1[6] += xnat[i][1] \* xnat[i][2]

a1[7] += xnat[i][0] \* xnat[i][1] \* xnat[i][2]

a.append(a1)

a.append(ai1(xnat, 0))

a.append(ai1(xnat, 1))

a.append(ai1(xnat, 2))

if n == 8:

a.append(ai2(xnat, 0, 1))

a.append(ai2(xnat, 0, 2))

a.append(ai2(xnat, 1, 2))

a.append(ai3(xnat, 0, 1, 2))

c = [0 for i in range(n)]

for i in range(n):

c[0] += ym[i]

c[1] += ym[i] \* xnat[i][0]

c[2] += ym[i] \* xnat[i][1]

c[3] += ym[i] \* xnat[i][2]

if n == 8:

c[4] += ym[i] \* xnat[i][0] \* xnat[i][1]

c[5] += ym[i] \* xnat[i][0] \* xnat[i][2]

c[6] += ym[i] \* xnat[i][1] \* xnat[i][2]

c[7] += ym[i] \* xnat[i][0] \* xnat[i][1] \* xnat[i][2]

ax = numpy.array(a)

cx = numpy.array(c)

b = numpy.linalg.solve(ax, cx)

return b

def table\_student(prob, n, m):

x\_vec = [i\*0.0001 for i in range(int(5/0.0001))]

par = 0.5 + prob/0.1\*0.05

f3 = (m - 1) \* n

for i in x\_vec:

if abs(t.cdf(i, f3) - par) < 0.000005:

return i

def table\_fisher(prob, n, m, d):

x\_vec = [i\*0.001 for i in range(int(10/0.001))]

f3 = (m - 1) \* n

for i in x\_vec:

if abs(f.cdf(i, n-d, f3)-prob) < 0.0001:

return i

def comb(arr):

return [1, \*arr, arr[0]\*arr[1], arr[0]\*arr[2], arr[1]\*arr[2], arr[0]\*arr[1]\*arr[2]]

def kohren(n, m, prob, disp):

fisher = table\_fisher(prob, n, m, 1)

gt = fisher/(fisher+(m-1)-2)

return max(disp) / sum(disp) < gt

def student(m, prob, disp, xnorm, ym):

n = len(ym)

sbt = (sum(disp) / m) \*\* (0.5) / n

beta = [sum([comb(xnorm[j][1:])[i] \* ym[j] / n for j in range(n)]) for i in range(n)]

tt = table\_student(prob, n, m)

st = [(abs(i) / sbt) > tt for i in beta]

return st

def fisher(m, prob, disp, ym, xnat, b, d):

n = len(ym)

if d == n:

return False

sad = sum([(sum([comb(xnat[i])[j] \* b[j] for j in range(n)]) - ym[i]) \*\* 2 for i in range(n)])

sad = sad \* m / (n - d)

fp = sad / sum(disp) / n

ft = table\_fisher(prob, n, m, d)

return fp < ft

def console\_print():

titles\_x = ["№", "X0", "X1", "X2", "X3", "X1\*X2", "X1\*X3", "X2\*X3", "X1\*X2\*X3"]

# cycle for pretty printing title of table with normal parameters

for j in range(N+1):

s = ""

if j == 0:

s = " {:1s} " # for №

if j == 1:

s = " {:2s} " # for X0

if j >= 2 and j < 5:

s = " {} " # for X + num

if j >= 5 and j < 8:

s = " {:5s} " # for X\*X, with different combinations

if j == 8:

s = " {:8s} " # for X\*X\*X

print(s.format(titles\_x[j]), end="") # taking all titles from list

# this cycle is used for printing Yi in title of table

for i in range(m):

print(" Yi{:d} ".format(i+1), end="")

# printing Y middle, Y experimental and dispersion

print(" Ys Ye S^2 ", end="")

print()

# fill table with data

for i in range(N):

print(" {:1d} {:2d} {:3d} {:3d} {:3d} ".format(i+1, xnorm[i][0], \*xnat[i]), end="")

if N == 8:

print(" {:5d} {:5d} {:5d} {:8d} "

.format(xnat[i][0]\*xnat[i][1], xnat[i][0]\*xnat[i][2], xnat[i][1]\*xnat[i][2],

xnat[i][0]\*xnat[i][1]\*xnat[i][2]), end="")

for j in y[i]:

print(" {:3d} ".format(j), end="")

yss = b[0]\*d\_arr[0] + b[1]\*xnat[i][0]\*d\_arr[1] + b[2]\*xnat[i][1]\*d\_arr[2] + b[3]\*xnat[i][2]\*d\_arr[3]

if N == 8:

yss += b[4]\*xnat[i][0]\*xnat[i][1]\*d\_arr[4] + b[5]\*xnat[i][0]\*xnat[i][2]\*d\_arr[5] + b[6]\*xnat[i][1]\*xnat[i][2]\*d\_arr[6] + b[7]\*xnat[i][0]\*xnat[i][1]\*xnat[i][2]\*d\_arr[7]

print(" {:6.2f} {:6.2f} {:6.2f} ".format(ym[i], yss, disp[i]), end="")

print()

print("\nNatural linear regrecy: Y = ", end="")

if d\_arr[0] != 0:

print("{:.5f}".format(b[0]), end="")

for i in range(1, N):

if d\_arr[i] != 0:

print(" + {:.5f}\*{}".format(b[i], titles\_x[i+1]), end="")

print("\n")

# the same style of printing table with natural parameters

for j in range(N+1):

s = ""

if j == 0:

s += " {} " # for №

if j == 1:

s += " {} " # for X0

if j >= 2 and j < 5:

s += " {} " # for X + num

if j >= 5 and j < 8:

s += " {} " # for X\*X, with different combinations

if j == 8:

s += " {} " # for X\*X\*X

print(s.format(titles\_x[j]), end="") # taking all titles from list

# this cycle is used for printing Yi in title of table

for i in range(m):

print(" Yi{:d} ".format(i+1), end="")

# printing Y middle, Y experimental and dispersion

print(" Ys Ye S^2 ", end="")

print()

# fill table with data

for i in range(N):

print(" {:1d} {:2d} {:2d} {:2d} {:2d} ".format(i+1, \*xnorm[i]), end="")

if N == 8:

print(" {:5d} {:5d} {:5d} {:8d} "

.format(xnorm[i][1]\*xnorm[i][2], xnorm[i][1]\*xnorm[i][3], xnorm[i][2]\*xnorm[i][3],

xnorm[i][1]\*xnorm[i][2]\*xnorm[i][3]), end="")

for j in y[i]:

print(" {:3d} ".format(j), end="")

yss = bnorm[0]\*d\_arr[0] + bnorm[1]\*xnorm[i][1]\*d\_arr[1] + bnorm[2]\*xnorm[i][2]\*d\_arr[2] + bnorm[3]\*xnorm[i][3]\*d\_arr[3]

if N == 8:

yss += bnorm[4]\*xnorm[i][1]\*xnorm[i][2]\*d\_arr[4] + bnorm[5]\*xnorm[i][1]\*xnorm[i][3]\*d\_arr[5] + bnorm[6]\*xnorm[i][2]\*xnorm[i][3]\*d\_arr[6] + bnorm[7]\*xnorm[i][1]\*xnorm[i][2]\*xnorm[i][3]\*d\_arr[7]

print(" {:6.2f} {:6.2f} {:6.2f} ".format(ym[i], yss, disp[i]), end="")

print()

print("\nNormal linear regrecy: Y = ", end="")

if d\_arr[0] != 0:

print("{:.5f}".format(bnorm[0]), end="")

for i in range(1, N):

if d\_arr[i] != 0:

print(" + {:.5f}\*{}".format(bnorm[i], titles\_x[i+1]), end="")

m = 3

N = 4

prob = 0.95

x1min = -5

x1max = 15

x2min = -25

x2max = 10

x3min = -5

x3max = 20

xmmin = (x1min + x2min + x3min) / 3

xmmax = (x1max + x2max + x3max) / 3

ymax = round(200 + xmmax)

ymin = round(200 + xmmin)

xnorm = [[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, 1, -1, -1],

[1, 1, 1, 1]]

xnat = [[x1min, x2min, x3min],

[x1min, x2max, x3max],

[x1max, x2min, x3max],

[x1max, x2max, x3min],

[x1min, x2min, x3max],

[x1min, x2max, x3min],

[x1max, x2min, x3min],

[x1max, x2max, x3max]]

while True:

print("\n\nStart N = {}".format(N))

y = new\_y(N, m, ymin, ymax)

while True:

print("\nm = {}\n".format(m))

ym = my(y)

b = get\_b\_nat(m, xnat, ym)

bnorm = get\_b\_norm(xnorm, ym)

disp = []

for i in range(len(ym)):

s = 0

for k in range(m):

s += (ym[i] - y[i][k]) \*\* 2

disp.append(s / m)

koh = kohren(N, m, prob, disp)

print("Dispersion uniform is {}, with probability = {:.2}".format(koh, prob))

if koh:

break

else:

m += 1

append\_y(N, y, ymin, ymax)

d\_arr = student(m, prob, disp, xnorm, ym)

d = sum(d\_arr)

fis = fisher(m, prob, disp, ym, xnat, b, d)

print("Equation adequativity is {}, with probability = {:.2f}\n".format(fis, prob))

console\_print()

if fis:

break

else:

if N == 4:

N = 8

else:

N = 4

m = 3

**Результати роботи програми**

Start N = 4

m = 3

Dispersion uniform is True, with probability = 0.95

Equation adequativity is True, with probability = 0.95

№ X0 X1 X2 X3 Yi1 Yi2 Yi3 Ys Ye S^2

1 1 -5 -25 -5 208 195 198 200.33 198.60 30.89

2 1 -5 10 20 200 212 188 200.00 198.60 96.00

3 1 15 -25 20 204 199 203 202.00 198.60 4.67

4 1 15 10 -5 192 196 202 196.67 198.60 16.89

Natural linear regrecy: Y = 198.60119

№ X0 X1 X2 X3 Yi1 Yi2 Yi3 Ys Ye S^2

1 1 -1 -1 -1 208 195 198 200.33 199.75 30.89

2 1 -1 1 1 200 212 188 200.00 199.75 96.00

3 1 1 -1 1 204 199 203 202.00 199.75 4.67

4 1 1 1 -1 192 196 202 196.67 199.75 16.89

Normal linear regrecy: Y = 199.75000