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»

Виконав:
студент групи ІО-82
Вербовський І. М.
Залікова книжка № 8205
Номер у списку групи 04
Перевірив ас. Регіда П. Г.

Київ 2020 р.

Лістинг програми

```
import random
import numpy

x1_min = 15
x1_max = 45

x2_min = 15
x2_max = 50

x3_min = 15
x3_max = 30

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 = [[x1_min, x2_min, x3_min],
     [x1_min, x2_max, x3_max],
     [x1_max, x2_min, x3_max],
     [x1_max, x2_max, x3_min]]

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):
    return max(dispersion) / sum(dispersion) < 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):
    mx = [0, 0, 0]
    axx = [0, 0, 0]
    ax = [0, 0, 0]
```

```
    for i in range(3):
        for j in range(4):
            mx[i] += x[j][i]
            axx[i] += x[j][i] ** 2
            ax[i] += x[j][i] * y_mean[j]
        mx[i] /= 4
        axx[i] /= 4
        ax[i] /= 4

    my = sum(y_mean) / 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

    a = numpy.array([[1, *mx],
                     [mx[0], axx[0], a12, a13],
                     [mx[1], a12, axx[1], a23],
                     [mx[2], a13, a23, axx[2]]])
    c = numpy.array([my, *ax])
    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:
        if m > 8:
            print("Current m is more than max number in database of Student criterion. Please
restart")
            exit(0)

        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)))

#console output
print("\n| № | X1 | X2 | X3 |", end="")

for i in range(m):
    print(" Yi{:d} |".format(i+1), end="")

print()
for i in range(4):
    print("| {:1d} | {:2d} | {:2d} | {:2d} |".format(i+1, *x[i]), end="")
    for j in y[i]:
        print(" {:3d} |".format(j), end="")
    print()

print("\nUsing G(Kohren) - criterion, current dispersion is uniform.")
print("Usind T(Student) - criterion, relevance of \n\tb0 is {}, b1 - {} b2 - {}, b3 -
{}".format(*k))
print("Using F(Fisher) - criterion, current regerecy equation is adequate.")

print("\n\tLinear regrecy equation:\tY = {:.2f}".format(b[0]), end="")
for i in range(1,4):
    if b[i] != 0:
        print(" + {0:.2f}".format(b[i]) + "*X" + str(i), end="")

print("\n\nControl result:")
for i in range(4):
    print("\t\t\t\tYs{:d}\t= {:.2f}\n\t b0 + b1*X1 + b2*X2 + b3*X3\t= {:.2f}"
          .format(i+1, y_mean[i], b[0] + b[1] * x[i][0] + b[2] * x[i][1]))
    print()
```

Результат роботи програми

```
| № | X1 | X2 | X3 | Yi1 | Yi2 |
| 1 | 15 | 15 | 15 | 217 | 232 |
| 2 | 15 | 50 | 30 | 223 | 228 |
| 3 | 45 | 15 | 30 | 237 | 238 |
| 4 | 45 | 50 | 15 | 220 | 216 |

Using G(Kohren) - criterion, current dispersion is uniform.
Usind T(Student) - criterion, relevance of
        b0 is True, b1 - False b2 - True, b3 - True
Using F(Fisher) - criterion, current regerecy equation is adequate.

        Linear regrecy equation:        Y = 216.84 + -0.26*X2 + 0.68*X3

Control result:
                                Ys1     = 224.50
         b0 + b1*X1 + b2*X2 + b3*X3     = 212.87

                                Ys2     = 225.50
         b0 + b1*X1 + b2*X2 + b3*X3     = 203.62

                                Ys3     = 237.50
         b0 + b1*X1 + b2*X2 + b3*X3     = 212.87

                                Ys4     = 218.00
         b0 + b1*X1 + b2*X2 + b3*X3     = 203.62
```