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.»

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

Київ 2020 р

Лістинг програми

```
import random
import numpy
from scipy.stats import t,f


#generate new experiment data
def gen_y(n, m, ymin, ymax):
    y = [[random.randint(ymin, ymax) for i in range(m)] for k in range(n)]
    return y


#do anotherone experiment
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


#return all combinations of three array elements
def comb(arr):
    return [1, *arr, arr[0]*arr[1], arr[0]*arr[2], arr[1]*arr[2], arr[0]*arr[1]*arr[2]]


#calculate b in normal equation
def get_b_norm(xnorm, ym):
    n = len(ym)
    b = [sum([comb(xnorm[j][1:])[i] * ym[j] / n for j in range(n)]) for i in range(n)]
    return b


#calculate b in natural equation
def get_b_nat(m, xnat, ym):
    n = len(ym)
    a = [[sum([comb(xnat[j])[i] * comb(xnat[j])[k] for j in range(n)]) for i in
range(n)]for k in range(n)]
    c = [sum([comb(xnat[j])[i] * ym[j] for j in range(n)]) for i in range(n)]
    return numpy.linalg.solve(numpy.array(a), numpy.array(c))


#return table for Student criteria
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


#return table for Fisher criteria
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


#test if dispersion is uniform using Kohren criteria
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


#test relevance of b using Student criteria
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


#test adequativity of equation using Fisher criteria
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_output():
    #natural====================================================================
    titles_x = ["№", "X0", "X1", "X2", "X3", "X1*X2", "X1*X3", "X2*X3", "X1*X2*X3"]

    for j in range(N+1):
        s = ""
        if j == 0:
            s = "| {:1s} |"
        if j == 1:
            s = " {:2s} |"
        if j >= 2 and j < 5:
            s = "  {} |"
        if j >= 5 and j < 8:
            s = " {:5s} |"
        if j == 8:
            s = " {:8s} |"
        print(s.format(titles_x[j]), end="")

    for i in range(m):
        print(" Yi{:d} |".format(i+1), end="")
    print("   Ys   |   Ye   |   S^2  |", end="")

    print()
    for i in range(N):
        print("| {:1d} |".format(i), end="")
        for j in range(N):
            s = ""
            if j < 1:
                s = " {:2d} |"
            if j >= 1 and j < 4:
                s = " {:3d} |"
            if j >= 4 and j < 7:
                s = " {:5d} |"
            if j == 7:
                s = " {:8d} |"
            print(s.format(comb(xnat[i])[j]), end="")

        for j in y[i]:
            print(" {:3d} |".format(j), end="")
```

```
        print(" {:6.2f} | {:6.2f} | {:6.2f} |"
              .format(ym[i], sum([comb(xnat[i])[j] * b[j] * d_arr[j] for j in range(N)]),
disp[i]), end="")

        print()

    print("\n\tNatural linear regrecy equation:  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="")

    #normal=============================================================================

    print("\n")
    for j in range(N+1):
        s = ""
        if j == 0:
            s += "| {} |"
        if j == 1:
            s += " {} |"
        if j >= 2 and j < 5:
            s += " {} |"
        if j >= 5 and j < 8:
            s += " {} |"
        if j == 8:
            s += " {} |"
        print(s.format(titles_x[j]), end="")

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

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

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

        for j in range(N):
            s = ""
            if j < 4:
                s = " {:2d} |"
            if j >= 4 and j < 7:
                s = " {:5d} |"
            if j == 7:
                s = " {:8d} |"
            print(s.format(comb(xnorm[i][1:])[j]), end="")
        for j in y[i]:
            print(" {:3d} |".format(j), end="")

        print(" {:6.2f} | {:6.2f} | {:6.2f} |"
              .format(ym[i], sum([comb(xnorm[i][1:])[j] * bnorm[j] * d_arr[j] for j in
range(N)]), disp[i]), end="")
        print()

    print("\n\tNormal linear regrecy equation:  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 = -20
x1max = 30
x2min = -25
x2max = 10
x3min = -25
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 algorythm with N = {}".format(N))
    y = gen_y(N, m, ymin, ymax)
    while True:
        print("\nCurrent m = {}\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_output()
    if fis:
        break
    else:
        N ^= 12 #change N between 8 and 4
        m = 3
```

Результати роботи програми

```
Start algorythm with N = 4

Current 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  |
| 0  |  1 | -20 | -25 | -25 | 199 | 204 | 201 | 201.33 | 199.24 |   4.22 |
| 1  |  1 | -20 |  10 | -20 | 202 | 182 | 184 | 189.33 | 189.24 |  80.89 |
| 2  |  1 |  30 | -25 | -20 | 188 | 197 | 184 | 189.67 | 189.24 |  29.56 |
| 3  |  1 |  30 |  10 | -25 | 188 | 204 | 201 | 197.67 | 199.24 |  48.22 |

        Natural linear regrecy equation:  Y = 149.23810 + -2.00000*X3

| №  | X0 | X1 | X2 | X3 | Yi1 | Yi2 | Yi3 |   Ys   |   Ye   |   S^2  |
| 1  |  1 | -1 | -1 | -1 | 199 | 204 | 201 | 201.33 | 199.50 |   4.22 |
| 2  |  1 | -1 |  1 |  1 | 202 | 182 | 184 | 189.33 | 189.50 |  80.89 |
| 3  |  1 |  1 | -1 |  1 | 188 | 197 | 184 | 189.67 | 189.50 |  29.56 |
| 4  |  1 |  1 |  1 | -1 | 188 | 204 | 201 | 197.67 | 199.50 |  48.22 |

        Normal linear regrecy equation:  Y = 194.50000 + -5.00000*X3
```