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

Методи оптимізації та планування експерименту

Лабораторна робота №3:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконав:
студент групи ІВ-81
Денисюк Ярослав
Залікова книжка № 8110
Перевірив Регіда П. Г.

Київ 2020р.

Лабораторна робота №3

Тема: ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

Мета: провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Виконання:

Варіант – 110.

110	-25	-5	-30	45	-5	5

1. Лістинг програми:

```
2. import random
   import numpy.linalg as l
   import math


   G_table = (
       (9985, 9750, 9392, 9057, 8772, 8534, 8332, 8159, 8010, 7880),
       (9669, 8709, 7977, 7457, 7071, 6771, 6530, 6333, 6167, 6025),
       (9065, 7679, 6841, 6287, 5892, 5598, 5365, 5175, 5017, 4884),
       (8412, 6838, 5981, 5440, 5063, 4783, 4564, 4387, 4241, 4118),
       (7808, 6161, 5321, 4803, 4447, 4184, 3980, 3817, 3682, 3568),
       (7271, 5612, 4800, 4307, 3974, 3726, 3535, 3384, 3259, 3154),
       (6798, 5157, 4377, 3910, 3595, 3362, 3185, 3043, 2926, 2829),
       (6385, 4775, 4027, 3584, 3286, 3067, 2901, 2768, 2659, 2568),
       (6020, 4450, 3733, 3311, 3029, 2823, 2666, 2541, 2439, 2353))

   t_table = [12.71, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262, 2.228,
   2.201, 2.179, 2.16, 2.145, 2.131, 2.12]

   F_table = (
        (164.4, 199.5, 215.7, 224.6, 230.2, 234),
        (18.5, 19.2, 19.2, 19.3, 19.3, 19.3),
        (10.1, 9.6, 9.3, 9.1, 9, 8.9),
        (7.7, 6.9, 6.6, 6.4, 6.3, 6.2),
        (6.6, 5.8, 5.4, 5.2, 5.1, 5),
        (6, 5.1, 4.8, 4.5, 4.4, 4.3),
        (5.5, 4.7, 4.4, 4.1, 4, 3.9),
        (5.3, 4.5, 4.1, 3.8, 3.7, 3.6),
        (5.1, 4.3, 3.9, 3.6, 3.5, 3.4),
        (5, 4.1, 3.7, 3.5, 3.3, 3.2),
        (4.8, 4, 3.6, 3.4, 3.2, 3.1),
        (4.8, 3.9, 3.5, 3.3, 3.1, 3))




   x1 = [-25, -5]
   x2 = [-30, 45]
   x3 = [-5, 5]

   m = 3
   N = 4
   q = 0.05

   x_matr = [[min(x1), min(x1), max(x1), max(x1)],
             [min(x2), max(x2), min(x2), max(x2)],
```

```
            [min(x3), max(x3), max(x3), min(x3)]]

x_ser_max = (max(x1) + max(x2) + max(x3)) / 3
x_ser_min = (min(x1) + min(x2) + min(x3)) / 3

y_max = 200 + x_ser_max
y_min = 200 + x_ser_min

#m
while True:
      y = [[round(random.uniform(y_min, y_max), 4) for i in range(m)] for j in
range(4)]
      for i in range(len(y)):
            print("y"+str(i+1)+":"+str(y[i]))
      print("\n")
      for i in range(len(x_matr)):
            print("x" + str(i + 1) + ":" + str(x_matr[i]))

      def ser_arr(arr):
            return sum(arr) / len(arr)

      y_ser_arr = list(map(ser_arr, y))
      print("\nСередні значення фунції відгуку: "+str(y_ser_arr))

      mx_arr = list(map(ser_arr, x_matr))
      print("\nmx: "+str(mx_arr))

      my = ser_arr(y_ser_arr)
      print("\nmy = "+str(my))

      def a(x):
            return (x[0] * y_ser_arr[0] + x[1] * y_ser_arr[1] + x[2] *
y_ser_arr[2] + x[3] * y_ser_arr[3]) / 4

      a1 = a(x_matr[0])
      a2 = a(x_matr[1])
      a3 = a(x_matr[2])

      a11 = (x_matr[0][0] * x_matr[0][0] + x_matr[0][1] * x_matr[0][1] +
x_matr[0][2] * x_matr[0][2] + x_matr[0][3] * x_matr[0][3]) / 4
      a22 = (x_matr[1][0] * x_matr[1][0] + x_matr[1][1] * x_matr[1][1] +
x_matr[1][2] * x_matr[1][2] + x_matr[1][3] *x_matr[1][3]) / 4
      a33 = (x_matr[2][0] * x_matr[2][0] + x_matr[2][1] * x_matr[2][1] +
x_matr[2][2] * x_matr[2][2] + x_matr[2][3] *x_matr[2][3]) / 4
      a12 = a21 = (x_matr[0][0] * x_matr[1][0] + x_matr[0][1] * x_matr[1][1] +
x_matr[0][2] * x_matr[1][2] + x_matr[0][3] *x_matr[1][3]) / 4
      a13 = a31 = (x_matr[0][0] * x_matr[2][0] + x_matr[0][1] * x_matr[2][1] +
x_matr[0][2] * x_matr[2][2] + x_matr[0][3] *x_matr[2][3]) / 4
      a23 = a32 = (x_matr[1][0] * x_matr[2][0] + x_matr[1][1] * x_matr[2][1] +
x_matr[1][2] * x_matr[2][2] + x_matr[1][3] *x_matr[2][3]) / 4

      print(
            "\na1= {:<6} a2= {:<6} a3= {:<6}\na11= {:<6} a22= {:<6} a33=
{:<6}\na12=a21= {:<6} a13=a31= {:<6} a23=a32= {:<6}\n".format(
                  round(a1,3),
                  round(a2,3),
                  round(a3,3),
                  round(a11,3),
                  round(a22,3),
                  round(a33,3),
                  round(a12,3),
                  round(a13,3),
                  round(a23,3),
            )
      )

      def det(sq_matr):
```

```
            return round(l.det(sq_matr), 4)

      mm = [[1, mx_arr[0], mx_arr[1], mx_arr[2]],
            [mx_arr[0], a11, a12, a13],
            [mx_arr[1], a12, a22, a32],
            [mx_arr[2], a13, a23, a33]]

      m0 = [[my, mx_arr[0], mx_arr[1], mx_arr[2]],
            [a1, a11, a12, a13],
            [a2, a12, a22, a32],
            [a3, a13, a23, a33]]

      m1 = [[1, my, mx_arr[1], mx_arr[2]],
            [mx_arr[0], a1, a12, a13],
            [mx_arr[1], a2, a22, a32],
            [mx_arr[2], a3, a23, a33]]

      m2 = [[1, mx_arr[0], my, mx_arr[2]],
            [mx_arr[0], a11, a1, a13],
            [mx_arr[1], a12, a2, a32],
            [mx_arr[2], a13, a3, a33]]

      m3 = [[1, mx_arr[0], mx_arr[1], my],
            [mx_arr[0], a11, a12, a1],
            [mx_arr[1], a12, a22, a2],
            [mx_arr[2], a13, a23, a3]]

      b = [det(m0) / det(mm), det(m1) / det(mm), det(m2) / det(mm), det(m3) /
det(mm)]

      print(
            "b1= {:<6} b2= {:<6} b3= {:<6} b4= {:<6}\n".format(
                  round(b[0],4),
                  round(b[1],4),
                  round(b[2],4),
                  round(b[3],4),
            )
      )

      y1 = b[0] + b[1] * min(x1) + b[2] * min(x2) + b[3] * min(x3)
      y2 = b[0] + b[1] * min(x1) + b[2] * max(x2) + b[3] * max(x3)
      y3 = b[0] + b[1] * max(x1) + b[2] * min(x2) + b[3] * max(x3)
      y4 = b[0] + b[1] * max(x1) + b[2] * max(x2) + b[3] * min(x3)

      D1 = (pow((y[0][0] - y1), 2) + pow((y[0][1] - y1), 2) + pow((y[0][2] - y1),
2)) / 3
      D2 = (pow((y[1][0] - y2), 2) + pow((y[1][1] - y2), 2) + pow((y[1][2] - y2),
2)) / 3
      D3 = (pow((y[2][0] - y3), 2) + pow((y[2][1] - y3), 2) + pow((y[2][2] - y3),
2)) / 3
      D4 = (pow((y[3][0] - y4), 2) + pow((y[3][1] - y4), 2) + pow((y[3][2] - y4),
2)) / 3
      D = [D1, D2, D3, D4]

      print(
            "D1= {:<6} D2= {:<6} D3= {:<6} D4= {:<6}\n".format(
                  round(D[0], 4),
                  round(D[1], 4),
                  round(D[2], 4),
                  round(D[3], 4),
            )
      )
      f1 = m - 1
      f2 = N
      print("f1 = m - 1 ="+str(f1))
      print("f2 = N ="+str(f2))
      Gp = max(D) / sum(D)
```

```
        Gt = G_table[f2-2][f1-1]*0.0001
        print("Критерій Кохрена: ")
        print("Gp = "+str(round(Gp,5)))
        print("Gt = "+str(round(Gt,5)))
        if Gp < Gt:
                print("Дисперсія однорідна (Gp < Gt)")
                break
        else:
                print("Дисперсія неоднорідна (Gp > Gt)")
                m+=1


print("\n\nКритерій Стьюдента:")
S = sum(D) / N
Ssq = math.sqrt(S / (N * m))
print("S{beta} = "+str(Ssq))

beta = [(y1 + y2 + y3 + y4)/4,
        (-y1 - y2 + y3 + y4)/4,
        (-y1 + y2 - y3 + y4)/4,
        (-y1 + y2 + y3 - y4)/4]

print(
            "\nbeta1= {:<6} beta2= {:<6} beta3= {:<6} beta4= {:<6}\n".format(
                  round(beta[0],5),
                  round(beta[1],5),
                  round(beta[2],5),
                  round(beta[3],5),
            )
      )
t=[]
for i in beta:
      t.append(abs(i)/Ssq)
print(
            "\nt1= {:<6} t2= {:<6} t3= {:<6} t4= {:<6}\n".format(
                  round(t[0],5),
                  round(t[1],5),
                  round(t[2],5),
                  round(t[3],5),
            )
      )

f3 = f1*f2
print(f"f3 = f1 * f2 = {f3}")
t_kr = t_table[f3]
print("t_kr = "+str(t_kr))
print("t(i) < t_kr: "+str([i for i in t if i<=t_kr]))
print("\nНезначимі коефіцієнти:"+str([b[i] for i in range(len(t)) if
t[i]<=t_kr]))
print("Значимі коефіцієнти: "+str([b[i] for i in range(len(t)) if t[i]>t_kr]))
t_final = list(filter(lambda x: x<t_kr, t))

for i in range(len(t)):
      if t[i] <= t_kr:
            b[i] = 0

y_t1 = b[0] + b[1] *  x_matr[0][0] + b[2] * x_matr[1][0] + b[3]* x_matr[2][0]
y_t2 = b[0] + b[1] *  x_matr[0][1] + b[2] * x_matr[1][1] + b[3]* x_matr[2][1]
y_t3 = b[0] + b[1] *  x_matr[0][2] + b[2] * x_matr[1][2] + b[3]* x_matr[2][2]
y_t4 = b[0] + b[1] *  x_matr[0][3] + b[2] * x_matr[1][3] + b[3]* x_matr[2][3]
y_t = [y_t1, y_t2, y_t3, y_t4]
print("Y-ки: "+str(y_t))
print("\nКритерій Фішера:")
d = N - len(t_final)
f4 = N-d
print("f4 = N - d = "+str(f4))
```

```
fisher_sum = 0
for i in range(0, N):
      fisher_sum += pow((y_t[i] - y_ser_arr[i]), 2)
D_ad = (m/(N-d))*fisher_sum
Fp = D_ad/S
print(f"Fp = {Fp}")
Ft = F_table[f3-1][f4-1]
print("Ft = "+str(Ft))
if Ft > Fp:
      print("Ft > Fp\nРівняння регресії адекватно оригіналу при рівні значимості
"+str(q))
else:
      print("Ft < Fp\nРівняння регресії неадекватно оригіналу при рівні
значимості "+str(q))
```

Контрольні запитання

1. *Що називається дробовим факторним експериментом?*

   У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

2. *Для чого потрібно розрахункове значення Кохрена?*

   Для перевірки дисперсії на однорідність.

3. *Для чого перевіряється критерій Стьюдента?*

   Для перевірки значущості коефіцієнтів регресії. Тобто, Якщо виконується нерівність $t_s < t_{табл}$, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт β_s є статистично незначущим і його слід виключити з рівняння регресії. Якщо $t_s > t_{табл}$ то гіпотеза не підтверджується, тобто β_s – значимий коефіцієнт і він залишається в рівнянні регресії.

4. *Чим визначається критерій Фішера і як його застосовувати?*

   Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.

Висновок: Отже, у ході виконання лабораторної роботи № 3 провели дробовий трьохфакторний експеримент. Склали матрицю планування, знайшли коефіцієнти рівняння регресії, провели 3 статистичні перевірки. Була написана текстова програма, результати наведені вище. Результати співпадають із калькулятором. Кінцева мета роботи досягнута!