НаціональнийтехнічнийуніверситетУкраїни

«КиївськийполітехнічнийінститутіменіІгоряСікорського»

Факультет інформатики та обчислювальноїтехніки

Кафедра обчислювальноїтехніки

**Методи оптимізації та планування експерименту**

Лабораторна робота №4:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ»

Виконав:

студент групи ІО-83

Веселовський Андрій

Залікова книжка № 8304

Перевірив Регіда П. Г.

Київ 2020р.

**Лабораторна робота №4**

**Тема:**ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ.

**Мета:**провести повнийтрьохфакторнийексперемент. Знайтирівняннярегресіїадекватнеоб'єкту.

**Виконання:**

Варіант – 306.



1. Лістинг програми:
2. from math import \*  
   from numpy.linalg import det  
   from random import randrange  
     
     
   class Critical\_values:  
    @staticmethod  
    def get\_cohren\_value(size\_of\_selections, qty\_of\_selections, significance):  
    from \_pydecimal import Decimal  
    from scipy.stats import f  
    size\_of\_selections += 1  
    partResult1 = significance / (size\_of\_selections - 1)  
    params = [partResult1, qty\_of\_selections, (size\_of\_selections - 1 - 1) \* qty\_of\_selections]  
    fisher = f.isf(\*params)  
    result = fisher / (fisher + (size\_of\_selections - 1 - 1))  
    return Decimal(result).quantize(Decimal('.0001')).\_\_float\_\_()  
     
    @staticmethod  
    def get\_student\_value(f3, significance):  
    from \_pydecimal import Decimal  
    from scipy.stats import t  
    return Decimal(abs(t.ppf(significance / 2, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
     
    @staticmethod  
    def get\_fisher\_value(f3, f4, significance):  
    from \_pydecimal import Decimal  
    from scipy.stats import f  
    return Decimal(abs(f.isf(significance, f4, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
   N = 8  
   m = 3  
   p = 0.95  
   matrix\_pfe = [  
    [-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, -1, +1, +1],  
    [+1, -1, +1, -1, +1, -1, -1],  
    [+1, +1, -1, +1, -1, -1, -1],  
    [+1, +1, +1, +1, +1, +1, +1],  
   ]  
   x1\_min, x1\_max = -30, 20  
   x2\_min, x2\_max = -25, 10  
   x3\_min, x3\_max = -30, -15  
   y\_min = 200 + int((x1\_min + x2\_min + x3\_min) / 3)  
   y\_max = 200 + int((x1\_max + x2\_max + x3\_max) / 3)  
   matrix\_x, matrix\_3x = [[] for x in range(N)], [[] for x in range(N)]  
   for i in range(len(matrix\_x)):  
    x1 = x1\_min if matrix\_pfe[i][0] == -1 else x1\_max  
    x2 = x2\_min if matrix\_pfe[i][1] == -1 else x2\_max  
    x3 = x3\_min if matrix\_pfe[i][2] == -1 else x3\_max  
    matrix\_x[i] = [x1, x2, x3, x1 \* x2, x1 \* x3, x2 \* x3, x1 \* x2 \* x3]  
    matrix\_3x[i] = [x1, x2, x3]  
     
   adekwatna, odnoridna = False, False # Адекватність і однорідність по замовчуванні False  
   while not adekwatna:  
    while not odnoridna:  
    def generate\_matrix():  
    matrix\_with\_y = [[randrange(y\_min, y\_max) for y in range(m)] for x in range(N)] # Генеруємо матрицю  
    return matrix\_with\_y  
     
    def find\_average(lst, orientation):  
    average = []  
    if orientation == 1: # Середнє значення по рядку  
    for rows in range(len(lst)):  
    average.append(sum(lst[rows]) / len(lst[rows]))  
    else: # Середнє значення по колонкі  
    for column in range(len(lst[0])):  
    number\_lst = []  
    for rows in range(len(lst)):  
    number\_lst.append(lst[rows][column])  
    average.append(sum(number\_lst) / len(number\_lst))  
    return average  
     
    def student\_test(b\_lst, number\_x=4):  
    dispersion\_b = sqrt(sum(dispersion\_y) / (N \* N \* m))  
    t\_lst = [0.0 for x in range(N)]  
    for k in range(number\_x):  
    for x in range(N):  
    if k == 0:  
    t\_lst[x] += average\_y[x] / N  
    else:  
    t\_lst[x] += average\_y[x] \* matrix\_pfe[x][k - 1] / N  
    for i in range(len(t\_lst)):  
    t\_lst[i] = fabs(t\_lst[i]) / dispersion\_b  
    tt = Critical\_values.get\_student\_value(f3, q)  
    for i in range(number\_x):  
    if t\_lst[i] > tt:  
    continue  
    else:  
    t\_lst[i] = 0  
    for j in range(number\_x):  
    b\_lst[j] = 0 if t\_lst[j] == 0 else b\_lst[j]  
    return b\_lst  
     
    def fisher(b\_lst, number=3):  
    dispersion\_ad = 0  
    for i in range(N):  
    yj = b\_lst[0]  
    for j in range(number):  
    yj += matrix[i][j] \* b\_lst[j + 1]  
    dispersion\_ad += (average\_y[i] - yj) \*\* 2  
    dispersion\_ad /= m / (N - d)  
    Fp = dispersion\_ad / (sqrt(sum(dispersion\_y) / (N \* N \* m)))  
    Ft = Critical\_values.get\_fisher\_value(f3, f4, q)  
    return True if Fp < Ft else False  
    matrix\_y = generate\_matrix() # Генеруємо матрицю у-ків  
    average\_x = find\_average(lst=matrix\_3x, orientation=0)  
    average\_y = find\_average(lst=matrix\_y, orientation=1)  
    a1, a2, a3, a11, a22, a33, a12, a13, a23 = 0, 0, 0, 0, 0, 0, 0, 0, 0  
    for i in range(N):  
    a1 += matrix\_x[i][0] \* average\_y[i] / N  
    a2 += matrix\_x[i][1] \* average\_y[i] / N  
    a3 += matrix\_x[i][2] \* average\_y[i] / N  
    a11 += matrix\_x[i][0] \*\* 2 / N  
    a22 += matrix\_x[i][1] \*\* 2 / N  
    a33 += matrix\_x[i][2] \*\* 2 / N  
    a12 += matrix\_x[i][0] \* matrix\_x[i][1] / N  
    a13 += matrix\_x[i][0] \* matrix\_x[i][2] / N  
    a23 += matrix\_x[i][1] \* matrix\_x[i][2] / N  
    a21 = a12  
    a31 = a13  
    a32 = a23  
    my = sum(average\_y) / len(average\_y)  
    b0\_numerator = [[my, average\_x[0], average\_x[1], average\_x[2]], [a1, a11, a12, a13], [a2, a21, a22, a23],  
    [a3, a31, a32, a33]]  
    b1\_numerator = [[1, my, average\_x[1], average\_x[2]], [average\_x[0], a1, a12, a13], [average\_x[1], a2, a22, a23],  
    [average\_x[2], a3, a32, a33]]  
    b2\_numerator = [[1, average\_x[0], my, average\_x[2]], [average\_x[0], a11, a1, a13], [average\_x[1], a21, a2, a23],  
    [average\_x[2], a31, a3, a33]]  
    b3\_numerator = [[1, average\_x[0], average\_x[1], my], [average\_x[0], a11, a12, a1], [average\_x[1], a21, a22, a2],  
    [average\_x[2], a31, a32, a3]]  
    b\_denominator = [[1, average\_x[0], average\_x[1], average\_x[2]], [average\_x[0], a11, a12, a13],  
    [average\_x[1], a21, a22, a23], [average\_x[2], a31, a32, a33]]  
    b0 = det(b0\_numerator) / det(b\_denominator)  
    b1 = det(b1\_numerator) / det(b\_denominator)  
    b2 = det(b2\_numerator) / det(b\_denominator)  
    b3 = det(b3\_numerator) / det(b\_denominator)  
    dispersion\_y = [0.0 for x in range(N)]  
    for i in range(N):  
    dispersion\_i = 0  
    for j in range(m):  
    dispersion\_i += (matrix\_y[i][j] - average\_y[i]) \*\* 2  
    dispersion\_y.append(dispersion\_i / (m - 1))  
    f1 = m - 1  
    f2 = N  
    f3 = f1 \* f2  
    q = 1 - p  
    Gp = max(dispersion\_y) / sum(dispersion\_y)  
    print("\tКритерій Кохрена")  
    Gt = Critical\_values.get\_cohren\_value(f2, f1, q)  
    if Gt > Gp or m >= 25:  
    print("\t\tДисперсія однорідна при рівні значимості {:.2f}!\n\tЗбільшувати m не потрібно.".format(q))  
    odnoridna = True  
    else:  
    print("\t\tДисперсія не однорідна при рівні значимості {:.2f}!".format(q))  
    m += 1  
    if m == 25:  
    exit()  
    matrix = []  
    for i in range(N):  
    matrix.append(matrix\_3x[i] + matrix\_y[i])  
    print("\tМатриця з натуральних значень факторів")  
    print("| X1 X2 X3 Y1 Y2 Y3 |")  
    for i in range(len(matrix)):  
    print("|", end=" ")  
    for j in range(len(matrix[i])):  
    print(matrix[i][j], end=" ")  
    print("|")  
    print("\tРівняння регресії")  
    print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 = ŷ".format(b0, b1, b2, b3))  
    print("\tКритерій Стьюдента")  
    beta\_1 = [b0, b1, b2, b3]  
    need\_koef = student\_test(beta\_1)  
    print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 = ŷ".format(need\_koef[0],  
    need\_koef[1],  
    need\_koef[2],  
    need\_koef[3]))  
    d = len(need\_koef) - need\_koef.count(0)  
    f4 = N - d  
    print("\tКритерій Фішера")  
    if not fisher(need\_koef):  
    print("\tРівняння регресії неадекватне стосовно оригіналу\nЕфект взаємодії!")  
    beta = [0 for i in range(N)]  
    for i in range(N):  
    if i == 0:  
    beta[i] += sum(average\_y) / len(average\_y)  
    else:  
    for j in range(7):  
    beta[i] += average\_y[i] \* matrix\_pfe[i][j] / N  
    print("\tРівняння регресії з ефектом взаємодії")  
    print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
    "+ {:.3f} \* Х1Х2X3= ŷ".format(beta[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6], beta[7]))  
    print("\tКритерій Кохрена")  
    Gt = Critical\_values.get\_cohren\_value(f2, f1, q)  
    if Gt > Gp or m >= 25:  
    print("\t\tДисперсія однорідна при рівні значимості {:.2f}!\n\tЗбільшувати m не потрібно.".format(q))  
    odnoridna = True  
    else:  
    print("\t\tДисперсія не однорідна при рівні значимості {:.2f}!".format(q))  
    m += 1  
    if m == 25:  
    exit()  
    need\_koef = student\_test(beta, 8)  
    print("\tКритерій Стьюдента")  
    print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
    "+ {:.3f} \* Х1Х2X3= ŷ".format(need\_koef[0], need\_koef[1],  
    need\_koef[2],  
    need\_koef[3],  
    need\_koef[4],  
    need\_koef[5],  
    need\_koef[6],  
    need\_koef[7]))  
    d = len(need\_koef) - need\_koef.count(0)  
    f4 = N - d  
    if student\_test(beta, 7):  
    print("Рівняння регресії адекватне стосовно оригіналу")  
    adekwatna = True  
    else:  
    print("\tРівняння регресії адекватне стосовно оригіналу")  
    adekwatna = True
3. Результатвиконання роботи програми:

**Критерій Кохрена**

**Дисперсія однорідна при рівні значимості 0.05!**

**Збільшувати m не потрібно.**

**Матриця з натуральних значень факторів**

**| X1 X2 X3 Y1 Y2 Y3 |**

**| -30 -25 -30 179 188 174 |**

**| -30 -25 -15 202 177 186 |**

**| -30 10 -30 182 195 172 |**

**| -30 10 -15 197 184 179 |**

**| 20 -25 -30 200 190 173 |**

**| 20 -25 -15 188 185 174 |**

**| 20 10 -30 185 180 176 |**

**| 20 10 -15 198 177 203 |**

**Рівняння регресії**

**192.498 + 0.023 \* X1 + 0.029 \* X2 + 0.311 \* X3 = ŷ**

**Критерій Стьюдента**

**192.498 + 0.000 \* X1 + 0.000 \* X2 + 0.311 \* X3 = ŷ**

**Критерій Фішера**

**Рівняння регресії неадекватне стосовно оригіналу**

**Ефект взаємодії!**

**Рівняння регресії з ефектом взаємодії**

**185.167 + -23.542 \* X1 + -22.875 \* X2 + -23.333 \* X3 + -23.458 \* Х1X2 + -22.792 \* Х1X3 + -22.542 \* Х2X3+ 168.583 \* Х1Х2X3= ŷ**

**Критерій Кохрена**

**Дисперсія однорідна при рівні значимості 0.05!**

**Збільшувати m не потрібно.**

**Критерій Стьюдента**

**0.000 + 0.000 \* X1 + 0.000 \* X2 + 0.000 \* X3 + 0.000 \* Х1X2 + 0.000 \* Х1X3 + 0.000 \* Х2X3+ 168.583 \* Х1Х2X3= ŷ**

**Рівняння регресії адекватне стосовно оригіналу**

**Висновок:**Отже, у ході виконання лабораторної роботи № 4 провели повний трьохфакторний експеримент при використанні рівняння з ефектом взаємодії. 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!