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 №6:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНAМИ(РОТОТАБЕЛЬНИЙ КОМПОЗИЦІЙНИЙ ПЛАН)»

Виконав:

студент групи ІО-83

Тимочко Дмитро Ігорович

Залікова книжка № 8326

Перевірив Регіда П. Г.

Київ 2020р.

**Лабораторна робота №6**

**Тема:** ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНАМИ(РОТОТАБЕЛЬНИЙ КОМПОЗИЦІЙНИЙ ПЛАН).

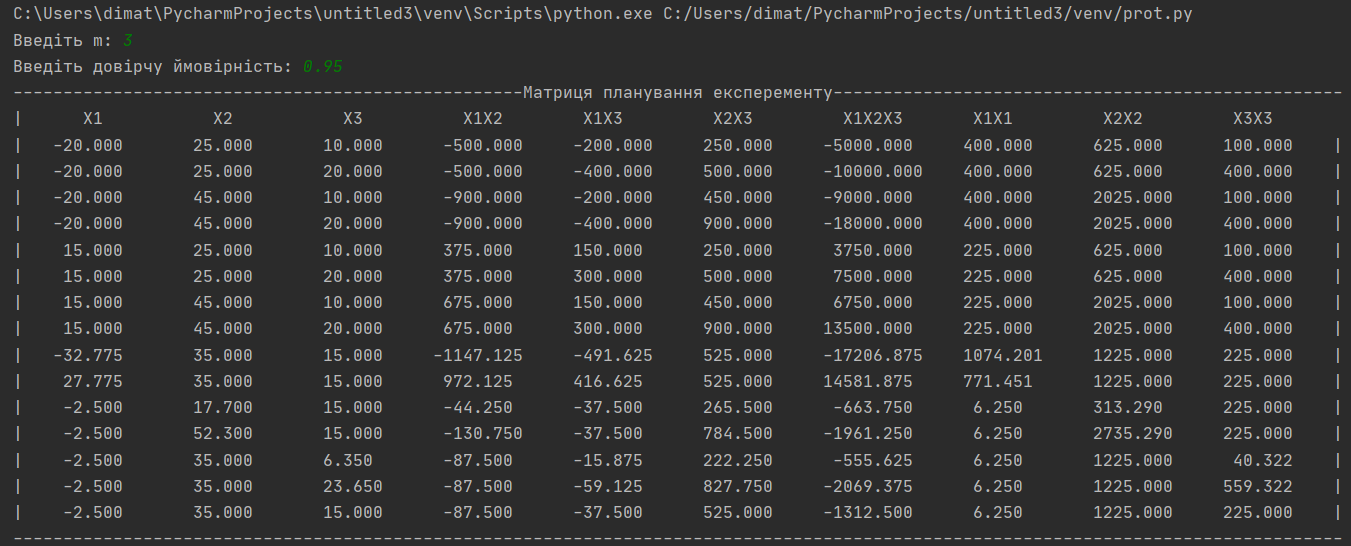
**Мета:** провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи рототабельний композиційний план. Знайти рівняння регресії адекватне об'єкту.

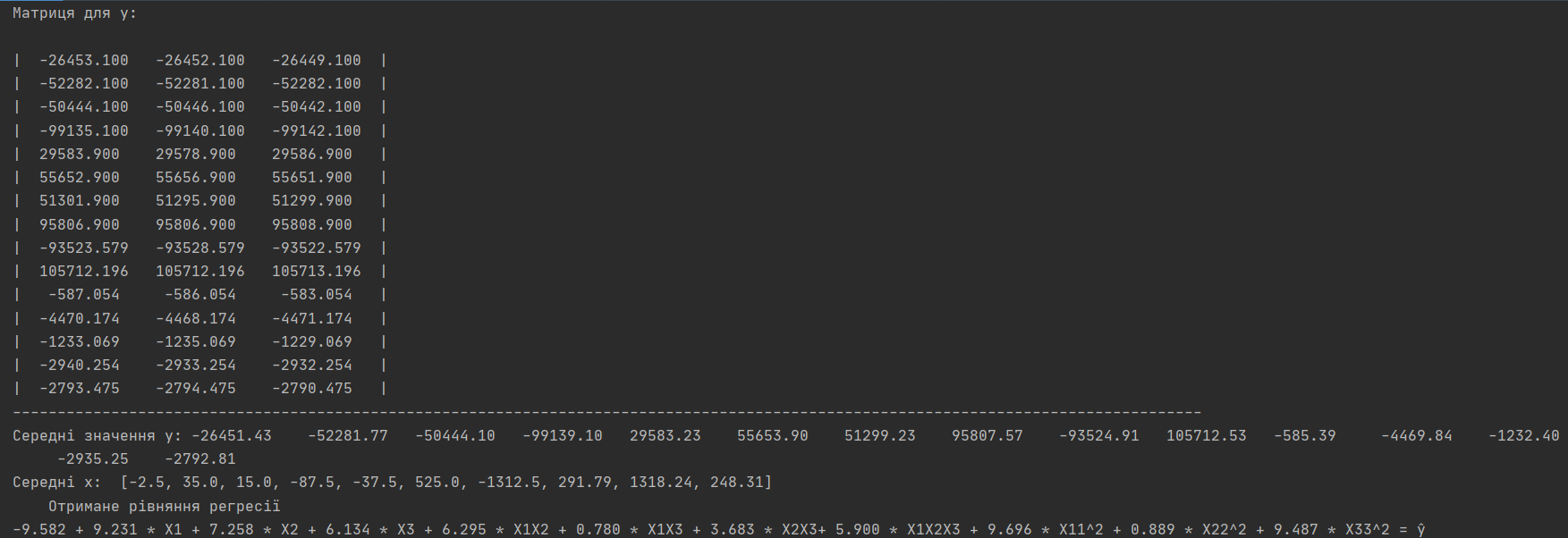
**Виконання:**

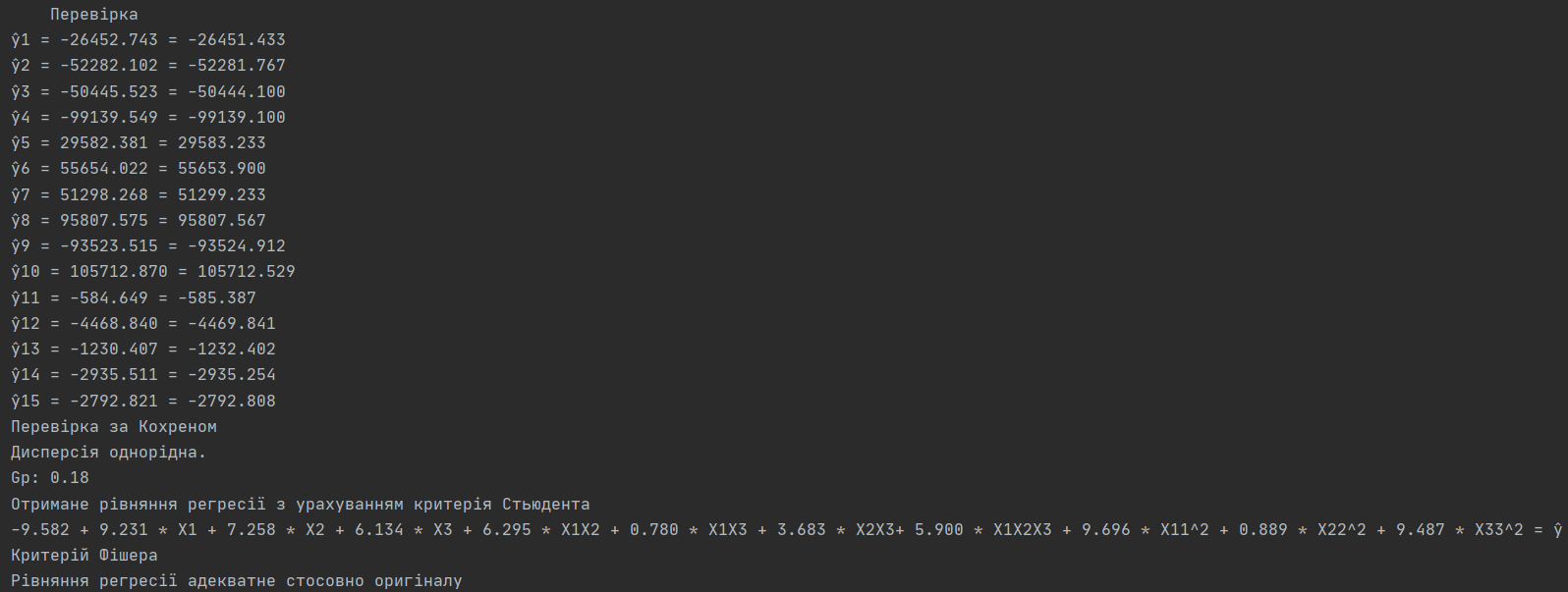
Варіант – 325.



1. Лістинг програми:
2. **from** numpy **import** \*  
   **from** math **import** \*  
   **import** numpy **as** np  
   **from** scipy.stats **import** f, t, ttest\_ind, norm  
   **from** \_pydecimal **import** Decimal, ROUND\_UP, ROUND\_FLOOR  
   **from** numpy.linalg **import** solve  
   **from** math **import** fabs  
   **from** random **import** randrange  
     
     
   *# Пошук критеріїв***class** Criteries:  
    @staticmethod  
    **def** get\_cohren\_value(size\_of\_selections, qty\_of\_selections, significance):  
    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):  
    **return** Decimal(abs(t.ppf(significance / 2, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
     
    @staticmethod  
    **def** get\_fisher\_value(f3, f4, significance):  
    **return** Decimal(abs(f.isf(significance, f4, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
     
     
   cr = Criteries()  
     
   m = int(input(**"Введіть m: "**))  
   p = float(input(**"Введіть довірчу ймовірність: "**))  
   N = 15  
     
   *# Задані за варіантом значення х*x1\_min, x1\_max = -20, 15  
   x2\_min, x2\_max = 25, 45  
   x3\_min, x3\_max = 10, 20  
     
   *# Матриця кодованих значень х*matrix\_x\_code = [[-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, -1, +1, +1, +1],  
    [+1, +1, -1, +1, -1, -1, -1, +1, +1, +1],  
    [+1, +1, +1, +1, +1, +1, +1, +1, +1, +1],  
    [-1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0, 0],  
    [+1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0, 0],  
    [0, -1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0],  
    [0, +1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0],  
    [0, 0, -1.73, 0, 0, 0, 0, 0, 0, +2.9979],  
    [0, 0, +1.73, 0, 0, 0, 0, 0, 0, +2.9979],  
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]  
     
   *# Розрахунки по формули для зоряної точки  
   # ========================================*x01 = (x1\_max + x1\_min) / 2  
   x02 = (x2\_max + x2\_min) / 2  
   x03 = (x3\_max + x3\_min) / 2  
   delta\_x1 = x1\_max - x01  
   delta\_x2 = x2\_max - x02  
   delta\_x3 = x3\_max - x03  
     
     
   **def** x\_formula(l1, l2, l3):  
    a = l1 \* delta\_x1 + x01  
    b = l2 \* delta\_x2 + x02  
    c = l3 \* delta\_x3 + x03  
    **return** [a, b, c]  
     
     
   *# ========================================  
     
   # Заповнюємо матрицю х*matrix\_x = [[] **for** x\_formula **in** range(N)]  
   **for** i **in** range(len(matrix\_x\_code)):  
    **if** i < 8:  
    x1 = x1\_min **if** matrix\_x\_code[i][0] == -1 **else** x1\_max  
    x2 = x2\_min **if** matrix\_x\_code[i][1] == -1 **else** x2\_max  
    x3 = x3\_min **if** matrix\_x\_code[i][2] == -1 **else** x3\_max  
    **else**:  
    x\_lst = x\_formula(matrix\_x\_code[i][0], matrix\_x\_code[i][1], matrix\_x\_code[i][2])  
    x1, x2, x3 = x\_lst  
     
    matrix\_x[i] = [x1, x2, x3, x1 \* x2, x1 \* x3, x2 \* x3, x1 \* x2 \* x3, x1 \*\* 2, x2 \*\* 2, x3 \*\* 2]  
     
   print(**"-"** \* 51 + **"Матриця планування експеременту"** + **"-"** \* 51)  
   print(**"| X1 X2 X3 X1X2 X1X3 X2X3 X1X2X3 X1X1"  
    " X2X2 X3X3"**)  
   *# Виводимо нашу матрицю***for** i **in** range(N):  
    print(**"|"**, end=**' '**)  
    **for** j **in** range(len(matrix\_x[0])):  
    print(**"{:^12.3f}"**.format(matrix\_x[i][j]), end=**' '**)  
    print(**"|"**)  
   print(**"-"** \* 133)  
     
     
   *# Генерація матриці у за заданою функцією по варіанту***def** random\_y():  
    **def** x(X1, X2, X3):  
    y = 9.9 + 9.0 \* X1 + 6.3 \* X2 + 5.3 \* X3 + 9.7 \* X1 \* X1 + 0.9 \* X2 \* X2 + 9.5 \* X3 \* X3 + 6.3 \* X1 \* X2 + \  
    0.8 \* X1 \* X3 + 3.7 \* X2 \* X3 + 5.9 \* X1 \* X2 \* X3 + randrange(0, 10) - 5  
    **return** y  
     
    matrix\_with\_y = [[x(matrix\_x[j][0], matrix\_x[j][1], matrix\_x[j][2]) **for** i **in** range(m)] **for** j **in** range(N)]  
    **return** matrix\_with\_y  
     
     
   check = **True  
   while** check:  
     
    *# Створюємо матрицю для у* random\_matrix\_y = random\_y();  
    print(**"Матриця для у: \n"**)  
    **for** i **in** range(N):  
    print(**"|"**, end=**' '**)  
    **for** j **in** range(len(random\_matrix\_y[0])):  
    print(**"{:^12.3f}"**.format(random\_matrix\_y[i][j]), end=**' '**)  
    print(**"|"**)  
    print(**"-"** \* 133)  
     
     
    *# Шукаємо середні значення у* **def** sum\_rows(random\_matrix\_y):  
    y = np.sum(random\_matrix\_y, axis=1) / m  
    **return** y  
     
     
    Yavg = sum\_rows(random\_matrix\_y)  
    print(**"Середні значення у: {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t "  
    "{:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t "  
    "{:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}"**.format(Yavg[0], Yavg[1], Yavg[2], Yavg[3],  
    Yavg[4], Yavg[5], Yavg[6], Yavg[7],  
    Yavg[8], Yavg[9], Yavg[10], Yavg[11],  
    Yavg[12], Yavg[13], Yavg[14]))  
     
     
    *# Шукаємо середні значення х* **def** sum\_columns(matrix\_x):  
    mx = np.sum(matrix\_x, axis=0) / 15  
    **return** mx  
     
     
    mx\_i = sum\_columns(matrix\_x)  
    c = []  
    **for** i **in** mx\_i:  
    c.append(round(i, 2))  
    print(**"Середні х: "**, c)  
     
     
    *# Шукаємо середнє значення my наших середніх Yavg* **def** sum\_my(y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15):  
    my = (y1 + y2 + y3 + y4 + y5 + y6 + y7 + y8 + y9 + y10 + y11 + y12 + y13 + y14 + y15) / 15  
    **return** my  
     
     
    my = sum\_my(Yavg[0], Yavg[1], Yavg[2], Yavg[3], Yavg[4],  
    Yavg[5], Yavg[6], Yavg[7], Yavg[8], Yavg[9],  
    Yavg[10], Yavg[11], Yavg[12], Yavg[13], Yavg[14])  
     
     
    *# Пошук коефіціентів a* **def** a(x, y):  
    a = 0  
    **for** j **in** range(N):  
    a += matrix\_x[j][x - 1] \* matrix\_x[j][y - 1] / N  
    **return** a  
     
     
    *# Пошук коефіціентів а1, а2, a3...an* **def** find(n):  
    a = 0  
    **for** j **in** range(N):  
    a += Yavg[j] \* matrix\_x[j][n - 1] / N  
    **return** a  
     
     
    *# Рахуємо коефіціенти для b* b = [  
    [1, mx\_i[0], mx\_i[1], mx\_i[2], mx\_i[3], mx\_i[4], mx\_i[5], mx\_i[6], mx\_i[7], mx\_i[8], mx\_i[9]],  
    [mx\_i[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1, 7), a(1, 8), a(1, 9), a(1, 10)],  
    [mx\_i[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2, 7), a(2, 8), a(2, 9), a(2, 10)],  
    [mx\_i[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3, 7), a(3, 8), a(3, 9), a(3, 10)],  
    [mx\_i[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4, 7), a(4, 8), a(4, 9), a(4, 10)],  
    [mx\_i[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5, 7), a(5, 8), a(5, 9), a(5, 10)],  
    [mx\_i[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6, 7), a(6, 8), a(6, 9), a(6, 10)],  
    [mx\_i[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7, 7), a(7, 8), a(7, 9), a(7, 10)],  
    [mx\_i[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8, 7), a(8, 8), a(8, 9), a(8, 10)],  
    [mx\_i[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9, 7), a(9, 8), a(9, 9), a(9, 10)],  
    [mx\_i[9], a(10, 1), a(10, 2), a(10, 3), a(10, 4), a(10, 5), a(10, 6), a(10, 7), a(10, 8), a(10, 9), a(10, 10)]  
    ]  
     
    zadany = [my, find(1), find(2), find(3), find(4), find(5), find(6), find(7), find(8), find(9), find(10)]  
    beta = solve(b, zadany)  
     
    matrix = [(matrix\_x[i] + random\_matrix\_y[i]) **for** i **in** range(N)]  
     
     
    *# Перевірка* **def** check(b\_lst, k):  
    y\_norm = b\_lst[0] + b\_lst[1] \* matrix[k][0] + b\_lst[2] \* matrix[k][1] + b\_lst[3] \* matrix[k][2] + \  
    b\_lst[4] \* matrix[k][3] + b\_lst[5] \* matrix[k][4] + b\_lst[6] \* matrix[k][5] + b\_lst[7] \* matrix[k][6] + \  
    b\_lst[8] \* matrix[k][7] + b\_lst[9] \* matrix[k][8] + b\_lst[10] \* matrix[k][9]  
    **return** y\_norm  
     
     
    print(**"\tОтримане рівняння регресії"**)  
    print(**"{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
    "+ {:.3f} \* Х1Х2X3 + {:.3f} \* X11^2 + {:.3f} \* X22^2 + {:.3f} \* X33^2 = ŷ\n\tПеревірка"** .format(beta[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6], beta[7], beta[8], beta[9], beta[10]))  
    **for** i **in** range(N):  
    print(**"ŷ{} = {:.3f} = {:.3f}"**.format((i + 1), check(beta, i), Yavg[i]))  
     
    X = []  
    **for** i **in** range(N):  
    X.append(check(beta, i))  
     
    *# Критерій Кохрена* print(**"Перевірка за Кохреном"**)  
    dispersion\_y = [0.0 **for** x **in** range(N)]  
    **for** i **in** range(N):  
    dispersion\_i = 0  
    **for** j **in** range(m):  
    dispersion\_i += ((random\_matrix\_y[i][j] - Yavg[i]) \*\* 2) / m  
    dispersion\_y.append(dispersion\_i)  
    Gp = max(dispersion\_y) / sum(dispersion\_y)  
    f1 = m - 1  
    f2 = N  
    q = 1 - p  
    Gt = cr.get\_cohren\_value(f2, f1, q)  
    **if** Gp <= Gt:  
    print(**"Дисперсія однорідна."**)  
    check = **False  
    else**:  
    m += 1  
    print(**"Отримали неоднорідну дисперсію, збільшуємо m."**)  
    print(**"Gp: {:.2f}"**.format(Gp))  
     
    *# ===================== Стьюдент =============================* f1 = m - 1  
    f2 = N  
    f3 = f1 \* f2  
    Ft = cr.get\_student\_value(f3, q)  
    S\_B = sum(dispersion\_y) / len(dispersion\_y)  
    S2\_beta = S\_B / (m \* N)  
    S\_b = S2\_beta \*\* (1 / 2)  
     
     
    **def** student(b\_lst, number\_x=10):  
    dispersion\_b = sqrt(dispersion\_b2)  
    **for** column **in** range(number\_x):  
    t\_practice = 0  
    t\_theoretical = cr.get\_student\_value(f3, q)  
    **for** row **in** range(N):  
    **if** column == 0:  
    t\_practice += Yavg[row] / N  
    **else**:  
    t\_practice += Yavg[row] \* matrix\_x\_code[row][column - 1]  
    **if** fabs(t\_practice / dispersion\_b) < t\_theoretical:  
    b\_lst[column] = 0  
    **return** b\_lst  
     
     
    dispersion\_b2 = sum(dispersion\_y) / (N)  
    student\_lst = list(student(beta))  
     
    print(**"Отримане рівняння регресії з урахуванням критерія Стьюдента"**)  
    print(**"{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
    "+ {:.3f} \* Х1Х2X3 + {:.3f} \* X11^2 + {:.3f} \* X22^2 + {:.3f} \* X33^2 = ŷ"** .format(student\_lst[0], student\_lst[1], student\_lst[2], student\_lst[3], student\_lst[4], student\_lst[5],  
    student\_lst[6], student\_lst[7], student\_lst[8], student\_lst[9], student\_lst[10]))  
     
     
    *# =======================Фішер==========================* **def** fisher\_test():  
    dispersion\_ad = 0  
    f4 = N - d  
    **for** row **in** range(len(Yavg)):  
    dispersion\_ad += (m \* (X[i] - Yavg[row])) / (N - d)  
    F\_practice = dispersion\_ad / dispersion\_b2  
    F\_theoretical = cr.get\_fisher\_value(f3, f4, q)  
    **return** F\_practice < F\_theoretical  
     
     
    print(**"Критерій Фішера"**)  
    d = 11 - student\_lst.count(0)  
    **if** fisher\_test():  
    print(**"Рівняння регресії адекватне стосовно оригіналу"**)  
    **else**:  
    print(**"Рівняння регресії неадекватне стосовно оригіналу"**)
3. Результат виконання роботи програми:







**Висновок:** Отже, у ході виконання лабораторної роботи № 6 провели трьохфакторний експеримент при використанні рівняння з урахуванням квадратичних членів. 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!