НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Дисципліна:**

«Методи наукових досліджень»

**Звіт**

Лабораторної роботи №5

**на тему:** «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

**Виконав:**

Студент групи: ІВ-92

Кубишка Юрій Сергійович

Варіант №13

**Перевірив:**

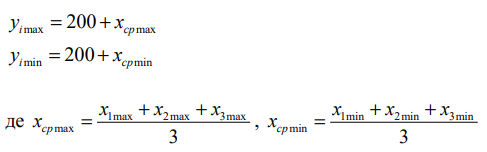
Регіда Павло Геннадійович

**Київ-2021**

**Мета:** Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

**Завдання на лабораторну роботу:**

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються за номером в списку в журналі викладача.

****

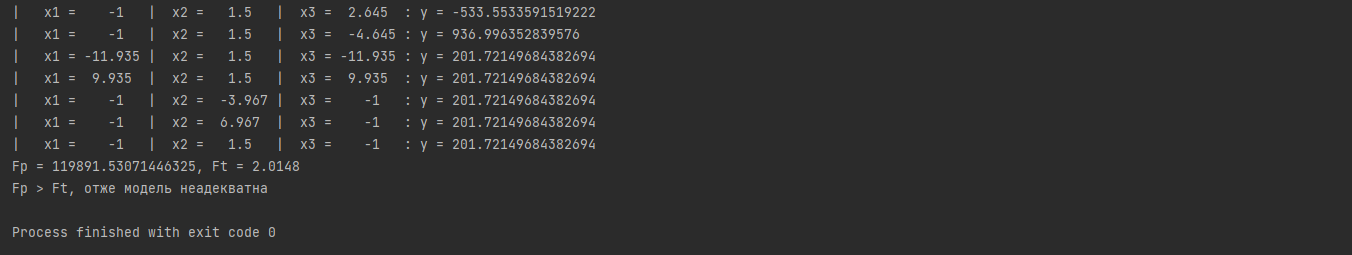
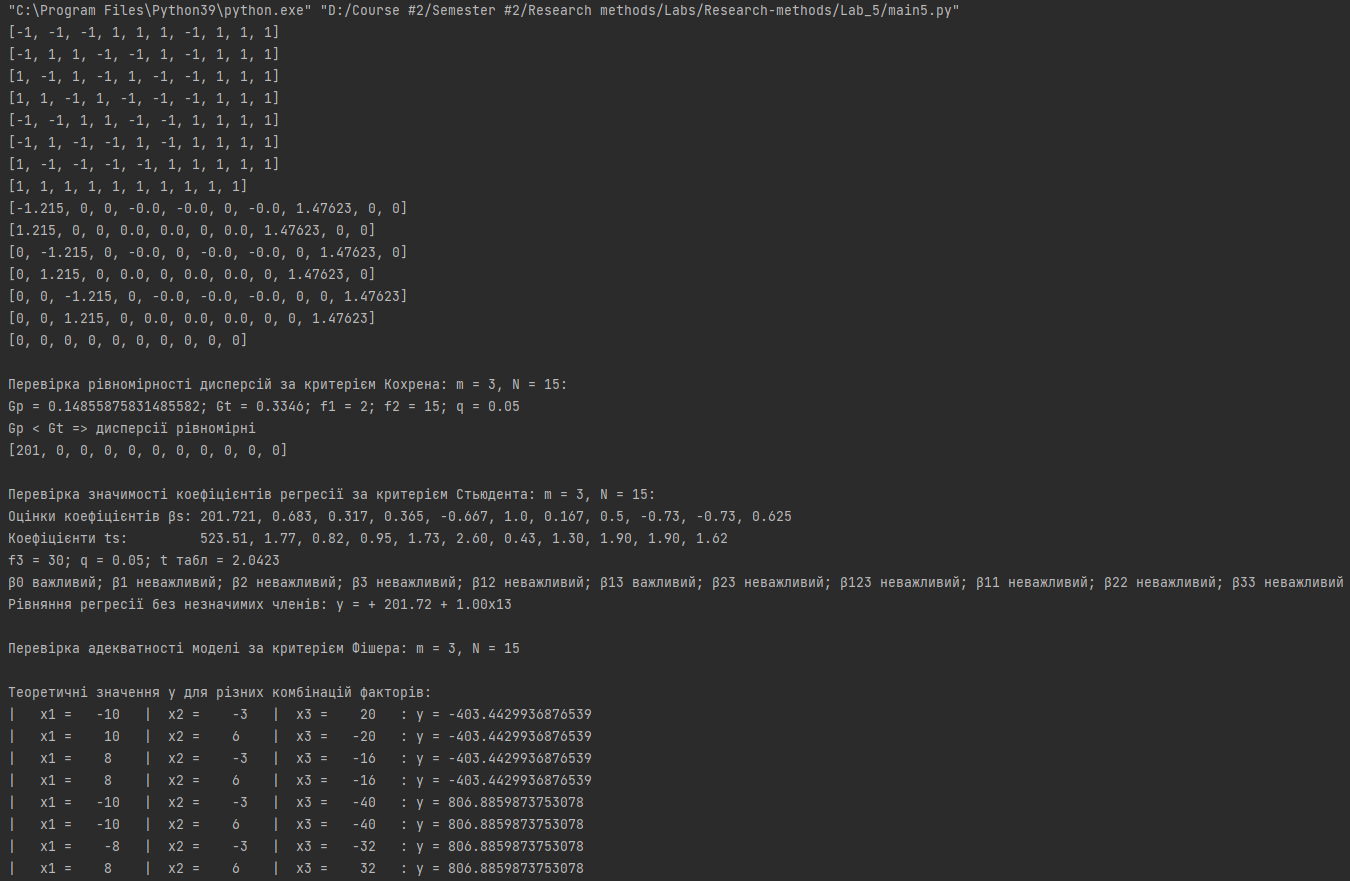
1. Розрахувати коефіцієнти рівняння регресії і записати його.
2. Провести 3 статистичні перевірки.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | x1 | | x2 | | x3 | |
|  | min | max | min | max | min | max |
| **213** | -2 | 4 | -10 | 8 | -3 | 6 |

**Код програми:**

import math  
import random  
from \_pydecimal import Decimal  
from functools import reduce  
from itertools import compress  
  
import numpy as np  
from scipy.stats import f, t  
  
  
def generate\_factors\_table(raw\_array):  
 return [row + [row[0] \* row[1], row[0] \* row[2], row[1] \* row[2], row[0] \* row[1] \* row[2]]  
 + list(map(lambda x: round(x \*\* 2, 5), row))  
 for row in raw\_array]  
  
  
def x\_i(i, raw\_factors\_table):  
 try:  
 assert i <= 10  
 except:  
 raise AssertionError(" i повинно бути <=10")  
 with\_null\_factor = list(map(lambda x: [1] + x, generate\_factors\_table(raw\_factors\_table)))  
 res = [row[i] for row in with\_null\_factor]  
 return np.array(res)  
  
  
def m\_ij(\*arrays):  
 return np.average(reduce(lambda accum, el: accum \* el, arrays))  
  
  
def cochran\_criteria(m, N, y\_table):  
 print(f"\nПеревірка рівномірності дисперсій за критерієм Кохрена: m = {m}, N = {N}: ")  
 y\_variations = [np.var(i) for i in y\_table]  
 max\_y\_variation = max(y\_variations)  
 gp = max\_y\_variation / sum(y\_variations)  
 f1 = m - 1  
 f2 = N  
 p = 0.95  
 q = 1 - p  
 gt = get\_cochran\_value(f1, f2, q)  
 print(f"Gp = {gp}; Gt = {gt}; f1 = {f1}; f2 = {f2}; q = {q:.2f}")  
 if gp < gt:  
 print("Gp < Gt => дисперсії рівномірні")  
 return True  
 else:  
 print("Gp > Gt => дисперсії нерівномірні - потрібно додати експерименти")  
 return False  
  
  
def get\_cochran\_value(f1, f2, q):  
 partResult1 = q / f2  
 params = [partResult1, f1, (f2 - 1) \* f1]  
 fisher = f.isf(\*params)  
 result = fisher / (fisher + (f2 - 1))  
 return Decimal(result).quantize(Decimal('.0001')).\_\_float\_\_()  
  
  
def student\_criteria(m, N, y\_table, beta\_coefficients):  
 print(f"\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {m}, N = {N}: ")  
 average\_variation = np.average(list(map(np.var, y\_table)))  
  
 variation\_beta\_s = average\_variation / N / m  
 standard\_deviation\_beta\_s = math.sqrt(variation\_beta\_s)  
 t\_i = np.array([abs(beta\_coefficients[i]) / standard\_deviation\_beta\_s for i in range(len(beta\_coefficients))])  
 f3 = (m - 1) \* N  
 q = 0.05  
  
 t = get\_student\_value(f3, q)  
 importance = [True if el > t else False for el in list(t\_i)]  
  
 print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x: str(round(float(x), 3)), beta\_coefficients))))  
 print("Коефіцієнти ts: " + ", ".join(list(map(lambda i: "{:.2f}".format(i), t\_i))))  
 print(f"f3 = {f3}; q = {q}; t табл = {t}")  
  
 beta\_i = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123", "β11", "β22", "β33"]  
 importance\_to\_print = ["важливий" if i else "неважливий" for i in importance]  
 to\_print = map(lambda x: x[0] + " " + x[1], zip(beta\_i, importance\_to\_print))  
 x\_i\_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23", "x123", "x1^2", "x2^2", "x3^2"], importance))  
 betas\_to\_print = list(compress(beta\_coefficients, importance))  
 print(\*to\_print, sep="; ")  
 equation = " ".join(["".join(i) for i in zip(list(map(lambda x: f"+ {x:.2f}", betas\_to\_print)), x\_i\_names)])  
 print(f"Рівняння регресії без незначимих членів: y = {equation}")  
 return importance  
  
  
def get\_student\_value(f3, q):  
 return Decimal(abs(t.ppf(q / 2, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
  
  
def calculate\_theoretical\_y(x\_table, b\_coefficients, importance):  
 x\_table = [list(compress(row, importance)) for row in x\_table]  
 b\_coefficients = list(compress(b\_coefficients, importance))  
 y\_vals = np.array([sum(map(lambda x, b: x \* b, row, b\_coefficients)) for row in x\_table])  
 return y\_vals  
  
  
def fisher\_criteria(m, N, d, naturalized\_x\_table, y\_table, b\_coefficients, importance):  
 f3 = (m - 1) \* N  
 f4 = N - d  
 q = 0.05  
  
 theoretical\_y = calculate\_theoretical\_y(naturalized\_x\_table, b\_coefficients, importance)  
 theoretical\_values\_to\_print = list(  
 zip(map(lambda x: f"|\tx1 = {x[1]:^8}|\tx2 = {x[2]:^8}|\tx3 = {x[3]:^8}", naturalized\_x\_table), theoretical\_y))  
  
 y\_averages = np.array(list(map(np.average, y\_table)))  
 s\_ad = m / (N - d) \* (sum((theoretical\_y - y\_averages) \*\* 2))  
 y\_variations = np.array(list(map(np.var, y\_table)))  
 s\_v = np.average(y\_variations)  
 f\_p = float(s\_ad / s\_v)  
 f\_t = get\_fisher\_value(f3, f4, q)  
  
 print(f"\nПеревірка адекватності моделі за критерієм Фішера: m = {m}, N = {N}")  
 print(f"\nТеоретичні значення y для різних комбінацій факторів:")  
 print(f"\n".join([f"{i[0]}: y = {i[1]}" for i in theoretical\_values\_to\_print]))  
 print(f"Fp = {f\_p}, Ft = {f\_t}")  
 print(f"Fp < Ft, отже модель адекватна" if f\_p < f\_t else f"Fp > Ft, отже модель неадекватна")  
 return True if f\_p < f\_t else False  
  
  
def get\_fisher\_value(f3, f4, q):  
 return Decimal(abs(f.isf(q, f4, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
  
  
def main():  
 m = 3  
 N = 15  
 ymin = 195  
 ymax = 206  
  
 raw\_naturalized\_factors\_table = [[-2, -10, -3], [-2, 10, 6], [-2, 8, -3], [-2, 8, 6],  
 [4, -10, -3], [4, -10, 6], [4, -8, -3], [4, 8, 6],  
 [-2.645, -1, 1.5], [4.645, -1, 1.5], [1, -11.935, 1.5], [1, 9.935, 1.5],  
 [1, -1, -3.967], [1, -1, 6.967], [1, -1, 1.5]]  
  
 raw\_factors\_table = [[-1, -1, -1], [-1, +1, +1], [+1, -1, +1], [+1, +1, -1],  
 [-1, -1, +1], [-1, +1, -1], [+1, -1, -1], [+1, +1, +1],  
 [-1.215, 0, 0], [+1.215, 0, 0], [0, -1.215, 0], [0, +1.215, 0],  
 [0, 0, -1.215], [0, 0, +1.215], [0, 0, 0]]  
  
 factors\_table = generate\_factors\_table(raw\_factors\_table)  
 for row in factors\_table:  
 print(row)  
 naturalized\_factors\_table = generate\_factors\_table(raw\_naturalized\_factors\_table)  
  
 y\_arr = [[random.randint(ymin, ymax) for \_ in range(m)] for \_ in range(N)]  
 while not cochran\_criteria(m, N, y\_arr):  
 m += 1  
 y\_arr = [[random.randint(ymin, ymax) for \_ in range(m)] for \_ in range(N)]  
  
 y\_i = np.array([np.average(row) for row in y\_arr])  
  
 coefficients = [[m\_ij(x\_i(column, raw\_factors\_table) \* x\_i(row, raw\_factors\_table)) for column in range(11)] for row  
 in range(11)]  
  
 free\_values = [m\_ij(y\_i, x\_i(i, raw\_factors\_table)) for i in range(11)]  
  
 beta\_coefficients = np.linalg.solve(coefficients, free\_values)  
 print(list(map(int, beta\_coefficients)))  
  
 importance = [student\_criteria(m, N, y\_arr, beta\_coefficients)]  
 d = importance.count(None)  
 fisher\_criteria(m, N, d, naturalized\_factors\_table, y\_arr, beta\_coefficients, importance)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результати роботи програми:**



Висновок:

При виконанні даної лабораторної роботи мною було змодельовано трьохфакторний експеримент з використанням лінійного рівняння регресії, рівняння регресії з ефектом взаємодії та квадратичних членів. В процесі було обчисоено рівняння регресії, адекватне об’єкту, складено матрицю планування, проведено 3 статичні перевірки, знайдено коефіцієнти рівняння регресії. При написанні даної лабораторної роботи познайомився із ОЦКП.