НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Дисципліна:**

«Методи наукових досліджень»

**Звіт**

Лабораторної роботи №6

**на тему:** «Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

**Виконав:**

Студент групи: ІВ-92

Кубишка Юрій Сергійович

Варіант №13

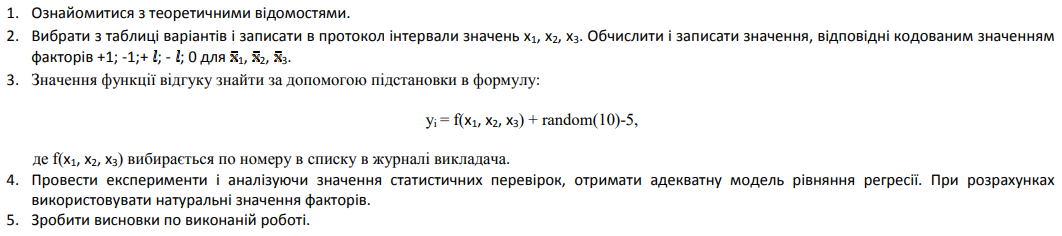
**Перевірив:**

Регіда Павло Геннадійович

**Київ-2021**

**Мета:** Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

**Завдання на лабораторну роботу:**



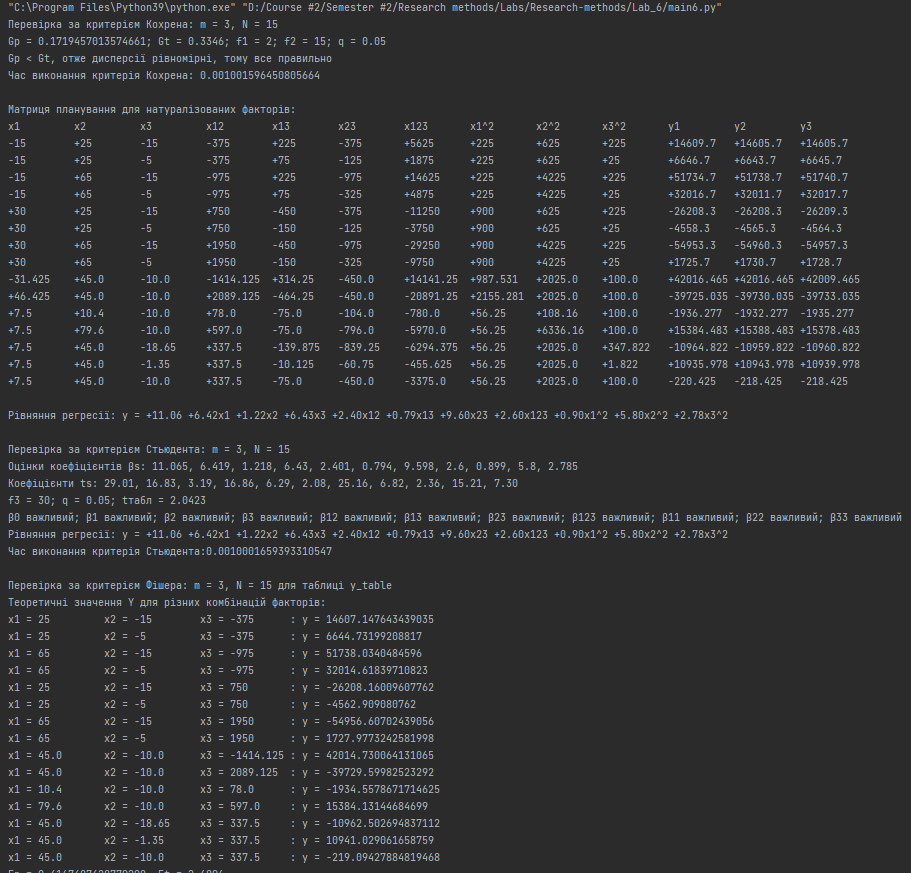
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | x1 | | x2 | | x3 | |
|  | min | max | min | max | min | max |
| **213** | -15 | 30 | 25 | 65 | -15 | -5 |



**Код програми:**

import math  
import random  
import time  
from \_decimal import Decimal  
from functools import reduce  
from itertools import compress  
  
import numpy  
from scipy.stats import f, t  
  
  
def generate\_y(m, factors\_table):  
 return [[round(func(row[0], row[1], row[2]) + random.randint(-5, 5), 3) for \_ in range(m)] for row in factors\_table]  
  
  
def func(x1, x2, x3):  
 coeffs = [6.7, 6.5, 1.3, 6.5, 2.4, 0.8, 9.6, 2.6, 0.9, 5.8, 2.8]  
 return equation\_of\_regression(x1, x2, x3, coeffs)  
  
  
def m\_ij(\*arrays):  
 return numpy.average(reduce(lambda accum, el: accum \* el, list(map(lambda el: numpy.array(el), arrays))))  
  
  
def equation\_of\_regression(x1, x2, x3, cef):  
 factors\_array = [1, x1, x2, x3, x1 \* x2, x1 \* x3, x2 \* x3, x1 \* x2 \* x3, x1 \*\* 2, x2 \*\* 2, x3 \*\* 2]  
 return sum([el[0] \* el[1] for el in zip(cef, factors\_array)])  
  
  
def generate\_factors\_table(raw\_array):  
 raw\_list = [row + [row[0] \* row[1], row[0] \* row[2], row[1] \* row[2], row[0] \* row[1] \* row[2]] + list(  
 map(lambda x: x \*\* 2, row)) for row in raw\_array]  
 return list(map(lambda row: list(map(lambda el: round(el, 3), row)), raw\_list))  
  
  
def set\_factors\_table(factors\_table):  
 def x\_i(i):  
 with\_null\_factor = list(map(lambda x: [1] + x, generate\_factors\_table(factors\_table)))  
 res = [row[i] for row in with\_null\_factor]  
 return numpy.array(res)  
  
 return x\_i  
  
  
def find\_coefficients(factors, y\_vals):  
 x\_i = set\_factors\_table(factors)  
 coeffs = [[m\_ij(x\_i(column), x\_i(row)) for column in range(11)] for row in range(11)]  
 y\_numpy = list(map(lambda row: numpy.average(row), y\_vals))  
 free\_values = [m\_ij(y\_numpy, x\_i(i)) for i in range(11)]  
 beta\_coefficients = numpy.linalg.solve(coeffs, free\_values)  
 return list(beta\_coefficients)  
  
  
def print\_matrix(m, N, factors, y\_vals, additional\_text=":"):  
 labels\_table = list(map(lambda x: x.ljust(10),  
 ["x1", "x2", "x3", "x12", "x13", "x23", "x123", "x1^2", "x2^2", "x3^2"] + [  
 "y{}".format(i + 1) for i in range(m)]))  
 rows\_table = [list(factors[i]) + list(y\_vals[i]) for i in range(N)]  
 print("\nМатриця планування" + additional\_text)  
 print(" ".join(labels\_table))  
 print("\n".join([" ".join(map(lambda j: "{:<+10}".format(j), rows\_table[i])) for i in range(len(rows\_table))]))  
 print("\t")  
  
  
def print\_equation(coeffs, importance=[True] \* 11):  
 x\_i\_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23", "x123", "x1^2", "x2^2", "x3^2"], importance))  
 coefficients\_to\_print = list(compress(coeffs, importance))  
 equation = " ".join(  
 ["".join(i) for i in zip(list(map(lambda x: f"{x:+.2f}", coefficients\_to\_print)), x\_i\_names)])  
 print("Рівняння регресії: y = " + equation)  
  
  
def cochran\_criteria(m, N, y\_table):  
 start\_cochrane = time.time()  
  
 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\_\_()  
  
 print("Перевірка за критерієм Кохрена: m = {}, N = {}".format(m, N))  
 y\_variations = [numpy.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)  
 stop = time.time()  
 print(f"Gp = {gp}; Gt = {gt}; f1 = {f1}; f2 = {f2}; q = {q:.2f}")  
 if gp < gt:  
 print("Gp < Gt, отже дисперсії рівномірні, тому все правильно")  
 print(f"Час виконання критерія Кохрена: {str(stop - start\_cochrane)}")  
  
 return True  
 else:  
 print("Gp > Gt, отже дисперсії нерівномірні, тому змінюємо значення m")  
 return False  
  
  
def student\_criteria(m, N, y\_table, beta\_coefficients):  
 start\_student = time.time()  
  
 def get\_student\_value(f3, q):  
 return Decimal(abs(t.ppf(q / 2, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
  
 print(f"\nПеревірка за критерієм Стьюдента: m = {m}, N = {N} ")  
 average\_variation = numpy.average(list(map(numpy.var, y\_table)))  
 variation\_beta\_s = average\_variation / N / m  
 standard\_deviation\_beta\_s = math.sqrt(variation\_beta\_s)  
 t\_i = [abs(beta\_coefficients[i]) / standard\_deviation\_beta\_s for i in range(len(beta\_coefficients))]  
 f3 = (m - 1) \* N  
 q = 0.05  
 t\_our = get\_student\_value(f3, q)  
 importance = [True if el > t\_our else False for el in list(t\_i)]  
 print(f"Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x: str(round(float(x), 3)), beta\_coefficients))))  
 print(f"Коефіцієнти ts: " + ", ".join(list(map(lambda i: "{:.2f}".format(i), t\_i))))  
 print(f"f3 = {f3}; q = {q}; tтабл = {t\_our}")  
 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))  
 print(\*to\_print, sep="; ")  
 print\_equation(beta\_coefficients, importance)  
 stop = time.time()  
 print("Час виконання критерія Стьюдента:" + str(stop - start\_student))  
 return importance  
  
  
def fisher\_criteria(m, N, d, x\_table, y\_table, b\_coefficients, importance):  
 start\_fisher = time.time()  
  
 def get\_fisher\_value(f3, f4, q):  
 return Decimal(abs(f.isf(q, f4, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
  
 f3 = (m - 1) \* N  
 f4 = N - d  
 q = 0.05  
 theoretical\_y = numpy.array([equation\_of\_regression(row[0], row[1], row[2], b\_coefficients) for row in x\_table])  
 average\_y = numpy.array(list(map(lambda el: numpy.average(el), y\_table)))  
 s\_ad = m / (N - d) \* sum((theoretical\_y - average\_y) \*\* 2)  
 y\_variations = numpy.array(list(map(numpy.var, y\_table)))  
 s\_v = numpy.average(y\_variations)  
 f\_p = float(s\_ad / s\_v)  
 f\_t = get\_fisher\_value(f3, f4, q)  
 theoretical\_values\_to\_print = list(  
 zip(map(lambda x: f"x1 = {x[1]:<10} x2 = {x[2]:<10} x3 = {x[3]:<10}", x\_table), theoretical\_y))  
 print(f"\nПеревірка за критерієм Фішера: m = {m}, N = {N} для таблиці y\_table")  
 print(f"Теоретичні значення Y для різних комбінацій факторів:")  
 print("\n".join([f"{el[0]}: y = {el[1]}" for el 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 "Fp > Ft, отже модель неадекватна")  
 stop = time.time()  
 print(f"Час виконання критерія Фішера: {str(stop - start\_fisher)}")  
 return True if f\_p < f\_t else False  
  
  
def main():  
 xmin = [-15, 25, -15]  
 xmax = [30, 65, -5]  
 norm\_plan\_raw = [[-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],  
 [+1.73, 0, 0],  
 [0, -1.73, 0],  
 [0, +1.73, 0],  
 [0, 0, -1.73],  
 [0, 0, +1.73]]  
  
 x0 = [(xmax[\_] + xmin[\_]) / 2 for \_ in range(3)]  
 dx = [xmax[\_] - x0[\_] for \_ in range(3)]  
  
 natur\_plan\_raw = [[xmin[0], xmin[1], xmin[2]],  
 [xmin[0], xmin[1], xmax[2]],  
 [xmin[0], xmax[1], xmin[2]],  
 [xmin[0], xmax[1], xmax[2]],  
 [xmax[0], xmin[1], xmin[2]],  
 [xmax[0], xmin[1], xmax[2]],  
 [xmax[0], xmax[1], xmin[2]],  
 [xmax[0], xmax[1], xmax[2]],  
 [-1.73 \* dx[0] + x0[0], x0[1], x0[2]],  
 [1.73 \* dx[0] + x0[0], x0[1], x0[2]],  
 [x0[0], -1.73 \* dx[1] + x0[1], x0[2]],  
 [x0[0], 1.73 \* dx[1] + x0[1], x0[2]],  
 [x0[0], x0[1], -1.73 \* dx[2] + x0[2]],  
 [x0[0], x0[1], 1.73 \* dx[2] + x0[2]],  
 [x0[0], x0[1], x0[2]]]  
  
 m = 3  
 N = 15  
 natural\_plan = generate\_factors\_table(natur\_plan\_raw)  
 y\_arr = generate\_y(m, natur\_plan\_raw)  
 while not cochran\_criteria(m, N, y\_arr):  
 m += 1  
 y\_arr = generate\_y(m, natural\_plan)  
  
 print\_matrix(m, N, natural\_plan, y\_arr, " для натуралізованих факторів:")  
 coefficients = find\_coefficients(natural\_plan, y\_arr)  
 print\_equation(coefficients)  
 importance = student\_criteria(m, N, y\_arr, coefficients)  
 d = len(list(filter(None, importance)))  
 fisher\_criteria(m, N, d, natural\_plan, y\_arr, coefficients, importance)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результати роботи програми:**





**Висновок:**

При виконанні даної лабораторної роботи було змодельовано трьохфакторний експеримент з квадратичними членами. В процесі було обчислено рівняння регресії, адекватне об’єкту, складено матрицю планування, проведено 3 статичні перевірки, знайдено коефіцієнти рівняння регресії, для кожної перевірки виведено час її виконання.

**Кінцева мета досягнута!**