Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №6**

З дисципліни «Методи оптимізації та планування»

**Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами**

ВИКОНАв:

Студент ІІ курсу ФІОТ

Групи ІО-93

Мудрий Юрій

ПЕРЕВІРИВ:

асистент

Регіда П.Г.

Київ 2021 р.

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

**Варіант завдання:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Варіант | Х1 | | Х2 | | Х3 | |
| min | max | min | max | min | max |
| 112 | -40 | 20 | -35 | 15 | 20 | 25 |

f(х1, х2, х3)=2,2+1,6\*x1+9,2\*x2+9,5\*x3+0,8\*x1\*x1+0,7\*x2\*x2+6,5\*x3\*x3+ +0,2\*x1\*x2+0,9\*x1\*x3+8,7\*x2\*x3+9,1\*x1\*x2\*x3

**Лістинг програми:**

import math  
import random  
from \_decimal import Decimal  
from itertools import compress  
from scipy.stats import f, t  
import numpy  
from functools import reduce  
  
def regression\_equation(x1, x2, x3, coeffs, importance=[True] \* 11):  
 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 compress(zip(coeffs, factors\_array), importance)])  
  
def func(x1, x2, x3):  
 coeffs = [2.2, 1.6, 9.2, 9.5, 0.8, 0.7, 6.5, 0.2, 0.9, 8.7, 9.1]  
 return regression\_equation(x1, x2, x3, coeffs)  
  
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]]  
  
natur\_plan\_raw = [[-40, -35, 20],  
 [-40, -35, 25],  
 [-40, 15, 20],  
 [-40, 15, 25],  
 [20, -35, 20],  
 [20, -35, 25],  
 [20, 15, 20],  
 [20, 15, 25],  
 [-61.9, 10, 22.5],  
 [-41.9, 10, 22.5],  
 [-10, -41.9, 22.5],  
 [-10, 61.9, 22.5],  
 [-10, 10, -29.4],  
 [-10, 10, 74.4],  
 [-10, 10, 22.5]]  
  
  
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 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 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: "{:+.2f}".format(x), coefficients\_to\_print)), x\_i\_names)])  
 print("Рівняння регресії: y = " + equation)  
  
  
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 m\_ij(\*arrays):  
 return numpy.average(reduce(lambda accum, el: accum \* el, list(map(lambda el: numpy.array(el), arrays))))  
  
  
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 cochran\_criteria(m, N, y\_table):  
 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 рівномірності дисперсій 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)  
 print("Gp = {}; Gt = {}; f1 = {}; f2 = {}; q = {:.2f}".format(gp, gt, f1, f2, q))  
 if gp < gt:  
 print("Gp < Gt => дисперсії рівномірні - все правильно")  
 return True  
 else:  
 print("Gp > Gt => дисперсії нерівномірні - треба ще експериментів")  
 return False  
  
  
def student\_criteria(m, N, y\_table, beta\_coefficients):  
 def get\_student\_value(f3, q):  
 return Decimal(abs(t.ppf(q / 2, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
  
 print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {}, N = {} ".format(m, 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 = numpy.array([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 result data  
 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("f3 = {}; q = {}; tтабл = {}".format(f3, q, 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)  
 return importance  
  
  
def fisher\_criteria(m, N, d, x\_table, y\_table, b\_coefficients, importance):  
 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([regression\_equation(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: "x1 = {0[1]:<10} x2 = {0[2]:<10} x3 = {0[3]:<10}".format(x), x\_table), theoretical\_y))  
 print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N = {} для таблиці y\_table".format(m, N))  
 print("Теоретичні значення y для різних комбінацій факторів:")  
 print("\n".join(["{arr[0]}: y = {arr[1]}".format(arr=el) for el in theoretical\_values\_to\_print]))  
 print("Fp = {}, Ft = {}".format(f\_p, f\_t))  
 print("Fp < Ft => модель адекватна" if f\_p < f\_t else "Fp > Ft => модель неадекватна")  
 return True if f\_p < f\_t else False  
  
  
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)

**Висновок:**

В даній лабораторній роботі я провела трьохфакторний експеримент і отримала адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.