НаціональнийтехнічнийуніверситетУкраїни

«КиївськийполітехнічнийінститутіменіІгоряСікорського»

Факультет інформатики та обчислювальноїтехніки

Кафедра обчислювальноїтехніки

**Методи оптимізації та планування експерименту**

Лабораторна робота №6:

«Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:

студент групи ІВ-92

Копайло Ярослав

Залікова книжка № 9213

Перевірив Регіда П. Г.

Київ 2021р.

**Лабораторна робота №6**

**Тема:** Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами.

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

композиційний план.

**Виконання:**

Варіант – 211.



1. Лістинг програми:

import random as r  
import numpy as np  
import pprint  
from scipy.stats import t**,** f  
import sklearn.linear\_model as lm  
from functools import partial  
  
x\_range = [[**10, 60**]**,** [-**35, 15**]**,** [**10, 15**]]  
x\_sered\_max = sum([x[**1**] for x in x\_range]) / **3**x\_sered\_min = sum([x[**0**] for x in x\_range]) / **3**x01 = (x\_range[**0**][**1**] - x\_range[**0**][**0**]) / **2**x02 = (x\_range[**1**][**1**] - x\_range[**1**][**0**]) / **2**x03 = (x\_range[**2**][**1**] - x\_range[**2**][**0**]) / **2**delta\_x1 = x\_range[**0**][**1**] - x01  
delta\_x2 = x\_range[**1**][**1**] - x02  
delta\_x3 = x\_range[**2**][**1**] - x03  
  
y\_max = **200** + x\_sered\_max  
y\_min = **200** + x\_sered\_min  
  
  
def create\_plan\_matrix(n**,** m):  
 x\_matrix\_norm = [  
 [**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, 1, 1, 1, 1, 1, 1, 1**]**,** [**1,** -**1.73, 0, 0, 0, 0, 0, 0, 2,9929, 0, 0**]**,** [**1, 1.73, 0, 0, 0, 0, 0, 0, 2,9929, 0, 0**]**,** [**1, 0,** -**1.73, 0, 0, 0, 0, 0, 0, 2,9929, 0**]**,** [**1, 0, 1.73, 0, 0, 0, 0, 0, 0, 2,9929, 0**]**,** [**1, 0, 0,** -**1.73, 0, 0, 0, 0, 0, 0, 2,9929**]**,** [**1, 0, 0, 1.73, 0, 0, 0, 0, 0, 0, 2,9929**]**,** [**1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0**]  
 ]  
 print('\nНормована матриця:')  
 pprint.pprint(x\_matrix\_norm)  
  
 x\_matrix = [[] for x in range(n)]  
 for i in range(len(x\_matrix)):  
 if i < **8**:  
 x1 = x\_range[**0**][**0**] if x\_matrix\_norm[i][**1**] == -**1** else x\_range[**0**][**1**]  
 x2 = x\_range[**1**][**0**] if x\_matrix\_norm[i][**2**] == -**1** else x\_range[**1**][**1**]  
 x3 = x\_range[**2**][**0**] if x\_matrix\_norm[i][**3**] == -**1** else x\_range[**2**][**1**]  
 else:  
 x1 = x\_matrix\_norm[i][**1**] \* delta\_x1 + x01  
 x2 = x\_matrix\_norm[i][**2**] \* delta\_x2 + x02  
 x3 = x\_matrix\_norm[i][**3**] \* delta\_x3 + x03  
 x\_matrix[i] = [**1,** float(format(x1**,** '.2f'))**,** float(format(x2**,** '.2f'))**,** float(format(x3**,** '.2f'))**,** float(format(x1 \* x2**,** '.2f'))**,** float(format(x1 \* x3**,** '.2f'))**,** float(format(x2 \* x3**,** '.2f'))**,** float(format(x1 \* x2 \* x3**,** '.2f'))**,** float(format(x1 \*\* **2,** '.2f'))**,** float(format(x2 \*\* **2,** '.2f'))**,** float(format(x3 \*\* **2,** '.2f'))]  
 print('\nНатуралізована матриця: ')  
 pprint.pprint(x\_matrix)  
 y = np.zeros(shape=(n**,** m))  
 for i in range(n):  
 for j in range(m):  
 y[i][j] = count\_y(x\_matrix[i])  
 print("y = 3,8+6,4\*x1+4,8\*x2+6,9\*x3+9,0\*x1\*x1+0,2\*x2\*x2+5,2\*x3\*x3+2,6\*x1\*x2+1,0\*x1\*x3+0,6\*x2\*x3+1,8\*x1\*x2\*x3 + random(10) - 5")  
 print('Y :')  
 pprint.pprint(y)  
  
 y\_avr = np.zeros(n)  
 for i in range(len(y)):  
 for j in range(len(y[**0**])):  
 y\_avr[i] += y[i][j] / m  
 return [x\_matrix\_norm**,** x\_matrix**,** y**,** y\_avr]  
  
  
def find\_coefs(x**,** y):  
 skm = lm.LinearRegression(fit\_intercept=False) # знаходимо коефіцієнти рівняння регресії  
 skm.fit(x**,** y)  
 B = skm.coef\_  
 print('Коефіціенти: ')  
 print(B)  
 return B  
  
  
def count\_y(x\_arr):  
 return **3.8** + **6.4** \* x\_arr[**1**] + **4.8** \* x\_arr[**2**] + **6.9** \* x\_arr[**3**] + **9.0** \* x\_arr[**8**] + **0.2** \* x\_arr[**9**] + **5.2** \* x\_arr[**10**] + **2.6** \* x\_arr[**4**] + **1.0** \* x\_arr[**5**] + **0.6** \* x\_arr[**6**] + **1.8** \* x\_arr[**7**] + r.randint(**0, 10**) -**5**def perevirka(x**,** y**,** b):  
 y\_pract = np.zeros(len(y))  
 for i in range(len(x)):  
 for j in range(len(x[**0**])):  
 y\_pract[i] += b[j] \* x[i][j]  
 print("\nПеревірка:")  
 print("\ny - real :"**,** y)  
 print('\ny - found:'**,** y\_pract)  
  
  
def get\_new\_y(x**,** b):  
 y\_pract = np.zeros(len(y))  
 for i in range(len(x)):  
 for j in range(len(x[**0**])):  
 y\_pract[i] += b[j] \* x[i][j]  
 return y\_pract  
  
  
def get\_cohren\_critical(prob**,** f1**,** f2):  
 f\_crit = f.isf((**1** - prob) / f2**,** f1**,** (f2 - **1**) \* f1)  
 return f\_crit / (f\_crit + f2 - **1**)  
  
  
def cohren\_crit(y**,** n**,** m):  
 y\_var = [np.var(i) for i in y]  
 Gp = max(y\_var) / sum(y\_var)  
 Gt = get\_cohren\_critical(**0.95,** m - **1,** n)  
 if (Gp < Gt):  
 print("\nДисперсії однорідні")  
 return True  
 else:  
 print("\nДисперсії не однорідні")  
 return False  
  
  
fisher\_teor = partial(f.ppf**,** q=**1** - **0.05**)  
  
student\_teor = partial(t.ppf**,** q=**1** - **0.025**)  
  
  
def kriteriy\_studenta(x**,** y**,** y\_aver**,** n**,** m**,** B):  
 d = **0** y\_var = [np.var(i) for i in y]  
 s\_kv\_aver = sum(y\_var) / n  
 s\_aver = (s\_kv\_aver / (n \* m)) \*\* **0.5** b = np.zeros(len(x[**0**]))  
 for i in range(len(x[**0**])):  
 for j in range(len(y\_aver)):  
 b[i] += y\_aver[j] \* x[j][i] / n  
 ts = []  
 for bi in b:  
 ts.append(abs(bi) / s\_aver)  
 Stud\_teor = student\_teor(df=(m - **1**) \* n)  
 for i in range(len(ts)):  
 if ts[i] < Stud\_teor:  
 B[i] = **0** else:  
 d += **1** print("\nКоефіціенти після перевірки нуль гіпотези: ")  
 print(B)  
 return [B**,** d]  
  
  
def kriteriy\_fishera(m**,** n**,** d**,** new\_y\_pract**,** y\_avr**,** y):  
 f4 = n - d  
 f3 = (m - **1**) \* n  
 y\_var = [np.var(i) for i in y]  
 Sa = (sum(y\_var) / n)  
 Sad = (m / (n - d)) \* sum([(new\_y\_pract[i] - y\_avr[i]) \*\* **2** for i in range(len(y\_avr))])  
 pract = Sad / Sa  
 teor = fisher\_teor(dfn=f4**,** dfd=f3)  
 if pract > teor:  
 print("\nПрактичне значення:"**,** pract)  
 print("Теоретичне значення:"**,** teor)  
 print("\nРівняння регресії неадекватне")  
 return [False**,** False]  
 else:  
 print("\nРівняння регресії адекватне")  
 return [True**,** True]  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 odnorid = False  
 adekvat = False  
 n = **15** m = **3** while not adekvat:  
 while not odnorid:  
 x\_matrix\_norm**,** x\_matrix**,** y**,** y\_avr = create\_plan\_matrix(n**,** m)  
 odnorid = cohren\_crit(y**,** n**,** m)  
 if odnorid == False:  
 m += **1** B = find\_coefs(x\_matrix**,** y\_avr)  
 perevirka(x\_matrix**,** y\_avr**,** B)  
 new\_B**,** d = kriteriy\_studenta(x\_matrix\_norm**,** y**,** y\_avr**,** n**,** m**,** B)  
 new\_y\_pract = get\_new\_y(x\_matrix**,** new\_B)  
 adekvat**,** odnorid = kriteriy\_fishera(m**,** n**, 4,** new\_y\_pract**,** y\_avr**,** y)

1. Результат виконання роботи програми:

Нормована матриця:

[[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, 1, 1, 1, 1, 1, 1, 1],

[1, -1.73, 0, 0, 0, 0, 0, 0, 2, 9929, 0, 0],

[1, 1.73, 0, 0, 0, 0, 0, 0, 2, 9929, 0, 0],

[1, 0, -1.73, 0, 0, 0, 0, 0, 0, 2, 9929, 0],

[1, 0, 1.73, 0, 0, 0, 0, 0, 0, 2, 9929, 0],

[1, 0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2, 9929],

[1, 0, 0, 1.73, 0, 0, 0, 0, 0, 0, 2, 9929],

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

Натуралізована матриця:

[[1, 10.0, -35.0, 10.0, -350.0, 100.0, -350.0, -3500.0, 100.0, 1225.0, 100.0],

[1, 10.0, -35.0, 15.0, -350.0, 150.0, -525.0, -5250.0, 100.0, 1225.0, 225.0],

[1, 10.0, 15.0, 10.0, 150.0, 100.0, 150.0, 1500.0, 100.0, 225.0, 100.0],

[1, 10.0, 15.0, 15.0, 150.0, 150.0, 225.0, 2250.0, 100.0, 225.0, 225.0],

[1,

60.0,

-35.0,

10.0,

-2100.0,

600.0,

-350.0,

-21000.0,

3600.0,

1225.0,

100.0],

[1,

60.0,

-35.0,

15.0,

-2100.0,

900.0,

-525.0,

-31500.0,

3600.0,

1225.0,

225.0],

[1, 60.0, 15.0, 10.0, 900.0, 600.0, 150.0, 9000.0, 3600.0, 225.0, 100.0],

[1, 60.0, 15.0, 15.0, 900.0, 900.0, 225.0, 13500.0, 3600.0, 225.0, 225.0],

[1, -35.55, 25.0, 2.5, -888.75, -88.88, 62.5, -2221.87, 1263.8, 625.0, 6.25],

[1, 85.55, 25.0, 2.5, 2138.75, 213.88, 62.5, 5346.88, 7318.8, 625.0, 6.25],

[1, 25.0, 42.3, 2.5, 1057.5, 62.5, 105.75, 2643.75, 625.0, 1789.29, 6.25],

[1, 25.0, 7.7, 2.5, 192.5, 62.5, 19.25, 481.25, 625.0, 59.29, 6.25],

[1,

25.0,

25.0,

-19.12,

625.0,

-478.12,

-478.12,

-11953.12,

625.0,

625.0,

365.77],

[1, 25.0, 25.0, 24.12, 625.0, 603.12, 603.12, 15078.12, 625.0, 625.0, 582.02],

[1, 25.0, 25.0, 2.5, 625.0, 62.5, 62.5, 1562.5, 625.0, 625.0, 6.25]]

y = 3,8+6,4\*x1+4,8\*x2+6,9\*x3+9,0\*x1\*x1+0,2\*x2\*x2+5,2\*x3\*x3+2,6\*x1\*x2+1,0\*x1\*x3+0,6\*x2\*x3+1,8\*x1\*x2\*x3 + random(10) - 5

Y :

array([[ -5686.2 , -5684.2 , -5681.2 ],

[ -8206.7 , -8206.7 , -8210.7 ],

[ 4949.8 , 4949.8 , 4953.8 ],

[ 7080.3 , 7079.3 , 7082.3 ],

[ -9412.2 , -9421.2 , -9415.2 ],

[-27439.7 , -27435.7 , -27431.7 ],

[ 52728.8 , 52720.8 , 52728.8 ],

[ 61848.3 , 61854.3 , 61850.3 ],

[ 5080.734, 5079.734, 5082.734],

[ 82156.784, 82155.784, 82153.784],

[ 14036.648, 14034.648, 14032.648],

[ 7327.168, 7332.168, 7333.168],

[-12850.732, -12855.732, -12855.732],

[ 38954.34 , 38962.34 , 38956.34 ],

[ 10624.05 , 10622.05 , 10620.05 ]])

Дисперсії однорідні

Коефіціенти:

[7.45488471 6.33918977 4.54893332 6.49766177 2.60498993 1.00639706

0.62057197 1.79963722 8.99998815 0.20214411 5.19618882]

Перевірка:

y - real : [ -5683.86666667 -8208.03333333 4951.13333333 7080.63333333

-9416.2 -27435.7 52726.13333333 61850.96666667

5081.06733333 82155.45066667 14034.648 7330.83466667

-12854.06533333 38957.67333333 10622.05 ]

y - found: [ -5683.18228072 -8208.81575384 4953.0873495 7081.68993009

-9415.44956304 -27436.30948434 52724.22551038 61853.06694774

5080.65785708 82155.56999801 14034.8549366 7329.04126835

-12853.68211173 38956.5223374 10621.44839186]

Коефіціенти після перевірки нуль гіпотези:

[7.45488471 6.33918977 4.54893332 6.49766177 2.60498993 1.00639706

0.62057197 1.79963722 8.99998815 0.20214411 5.19618882]

Рівняння регресії адекватне

**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!