Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

Тема: «Проведення трьохфакторного експерименту при

використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)»

Виконав:

студент групи ІО-93

Ващенко Ірина Олегівна

Номер в списку групи: 4

Перевірив:

ас. Регiда П. Г.

Київ 2021

**Лабораторна робота № 5**

**Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів**

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

**Завдання:**

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 по номеру в списку в журналі викладача.

4. Розрахувати коефіцієнти рівняння регресії і записати його.

5. Провести 3 статистичні перевірки.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варианта | X1 | | X2 | | X3 | |
| min | max | min | max | min | max |
| 304 | -5 | 8 | -1 | 4 | -4 | 2 |

import random  
import sklearn.linear\_model as lm  
from scipy.stats import f, t  
from functools import partial  
from pyDOE2 import \*  
  
x\_range = ((-5, 8), (-1, 4), (-4, 2))  
x\_maxavr = (8 + 4 - 2)/3  
x\_minavr = (-5 - 1 - 4)/3  
  
y\_max = 200 + int(x\_maxavr)  
y\_min = 200 + int(x\_minavr)  
y\_min\_max = [y\_min, y\_max]  
  
def suma(x, b):  
 y = sum([x[i] \* b[i] for i in range(len(x))])  
 return y  
  
  
  
def matrixs(n, m):  
 y = np.random.randint(\*y\_min\_max, size=(n, m))  
 norm\_matrix = ccdesign(3, center=(0, 1))  
 norm\_matrix = np.insert(norm\_matrix, 0, 1, axis=1)  
  
 for i in range(4, 11):  
 norm\_matrix = np.insert(norm\_matrix, i, 0, axis=1)  
  
 l = 1.215  
  
 for i in range(len(norm\_matrix)):  
 for j in range(len(norm\_matrix[i])):  
 if norm\_matrix[i][j] < -1 or norm\_matrix[i][j] > 1:  
 if norm\_matrix[i][j] < 0:  
 norm\_matrix[i][j] = -l  
 else:  
 norm\_matrix[i][j] = l  
  
 def addnums(x):  
 for i in range(len(x)):  
 x[i][4] = x[i][1] \* x[i][2]  
 x[i][5] = x[i][1] \* x[i][3]  
 x[i][6] = x[i][2] \* x[i][3]  
 x[i][7] = x[i][1] \* x[i][3] \* x[i][2]  
 x[i][8] = x[i][1] \*\* 2  
 x[i][9] = x[i][2] \*\* 2  
 x[i][10] = x[i][3] \*\* 2  
 return x  
  
  
 x = np.ones((15, 11,), dtype=np.int64)  
 for i in range(8):  
 for j in range(1, 4):  
 if norm\_matrix[i][j] == -1:  
 x[i][j] = x\_range[j - 1][0]  
 else:  
 x[i][j] = x\_range[j - 1][1]  
  
 for i in range(8, len(x)):  
 for j in range(1, 3):  
 x[i][j] = (x\_range[j - 1][0] + x\_range[j - 1][1]) / 2  
  
 dx = [x\_range[i][1] - (x\_range[i][0] + x\_range[i][1]) / 2 for i in range(3)]  
  
 x[8][1] = l \* dx[0] + x[9][1]  
 x[9][1] = -l \* dx[0] + x[9][1]  
 x[10][2] = l \* dx[1] + x[9][2]  
 x[11][2] = -l \* dx[1] + x[9][2]  
 x[12][3] = l \* dx[2] + x[9][3]  
 x[13][3] = -l \* dx[2] + x[9][3]  
  
 x = addnums(x)  
 print(**"Нормована матриця:"**)  
 for i in norm\_matrix:  
 print([round(x, 2) for x in i])  
 return x, y, norm\_matrix  
  
  
def find\_coef(X, Y, norm=False):  
 skm = lm.LinearRegression(fit\_intercept=False)  
 skm.fit(X, Y)  
 B = skm.coef\_  
  
 if norm == 1:  
 print(**'**\n**Коефіцієнти з нормованими X:'**)  
 else:  
 print(**'**\n**Коефіцієнти рівняння регресії:'**)  
 B = [round(i, 3) for i in B]  
 print(**'**\n**Результат рівняння зі знайденими коефіцієнтами:**\n**'**, np.dot(X, B))  
 return B  
  
def count\_dispersion(y, y\_avarg, n, m):  
 y\_var = np.var(y, axis=1)  
 return y\_var  
  
  
def check\_criter\_Cochran(y, y\_avarg, n, m):  
 S\_kv = count\_dispersion(y, y\_avarg, n, m)  
 Gp = max(S\_kv) / sum(S\_kv)  
 print(**'**\n**Перевірка за критерієм Кохрена'**)  
 return Gp  
  
  
def cohren(f1, f2, q=0.05):  
 q1 = q / f1  
 fisher\_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) \* f2)  
 return fisher\_value / (fisher\_value + f1 - 1)  
  
  
def bs(x, y\_avarg, n):  
 res = [sum(1 \* y for y in y\_avarg) / n]  
 for i in range(3):  
 b = sum(j[0] \* j[1] for j in zip(x[:, i], y\_avarg)) / n  
 res.append(b)  
 return res  
  
def s\_kv(y, y\_aver, n, m):  
 res = []  
 for i in range(n):  
 s = sum([(y\_aver[i] - y[i][j]) \*\* 2 for j in range(m)]) / m  
 res.append(round(s, 3))  
 return res  
  
def kriteriy\_studenta(x, y, y\_aver, n, m):  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 s\_kv\_aver = sum(S\_kv) / n  
  
 s\_Bs = (s\_kv\_aver / n / m) \*\* 0.5  
 Bs = bs(x, y\_aver, n)  
 ts = [round(abs(B) / s\_Bs, 3) for B in Bs]  
 return ts  
  
  
def adequacy\_Fishera(y, y\_avarg, y\_new, n, m, d):  
 S\_ad = m / (n - d) \* sum([(y\_new[i] - y\_avarg[i])\*\*2 for i in range(len(y))])  
 S\_kv = count\_dispersion(y, y\_avarg, n, m)  
 S\_kv\_aver = sum(S\_kv) / n  
 return S\_ad / S\_kv\_aver  
  
#Перевiрка  
def check(X, Y, B, n, m):  
 f1 = m - 1  
 f2 = n  
 f3 = f1 \* f2  
 q = 0.05  
  
 student = partial(t.ppf, q=1 - q)  
 t\_student = student(df=f3)  
  
 G\_kr = cohren(f1, f2)  
  
 y\_aver = [round(sum(i) / len(i), 3) for i in Y]  
  
 d\_p = s\_kv(Y, y\_aver, n, m)  
 print(**f'Дисперсія y:** {d\_p}**'**)  
  
 Gp = check\_criter\_Cochran(Y, y\_aver, n, m)  
 print(**f'Gp =** {Gp}**'**)  
 if Gp < G\_kr:  
 print(**f'Дисперсії однорідні** {1 - q}**'**)  
 else:  
 print(**"Збільшити кількість дослідів"**)  
 m += 1  
 main(n, m)  
  
 ts = kriteriy\_studenta(X[:, 1:], Y, y\_aver, n, m)  
 print(**f'Критерій Стьюдента:** {ts}**'**)  
 res = [t for t in ts if t > t\_student]  
 k = [B[i] for i in range(len(ts)) if ts[i] in res]  
 print(**'**\n**Коефіцієнти статистично незначущі {} '**.format(  
 [round(i, 3) for i in B if i not in k]))  
  
 y\_new = []  
 for j in range(n):  
 y\_new.append(suma([X[j][i] for i in range(len(ts)) if ts[i] in res], k))  
  
 print(**f'**\n**Значення "y" з коефіцієнтами** {k}**'**)  
 print(y\_new)  
  
 d = len(res)  
 if d >= n:  
 print(**'**\n**F4 <= 0'**)  
 print(**''**)  
 return  
 f4 = n - d  
  
 F\_p = adequacy\_Fishera(Y, y\_aver, y\_new, n, m, d)  
  
 fisher = partial(f.ppf, q=0.95)  
 f\_t = fisher(dfn=f4, dfd=f3)  
 print(**'**\n**Перевірка адекватності за критерієм Фішера'**)  
 print(**'Fp ='**, F\_p)  
 print(**'F\_t ='**, f\_t)  
 if F\_p < f\_t:  
 print(**'Математична модель є адекватна'**)  
 else:  
 print(**'Математична модель не є адекватною'**)  
  
  
def main(n, m):  
 X5, Y5, X5\_norm = matrixs(n, m)  
  
 y5\_aver = [round(sum(i) / len(i), 3) for i in Y5]  
 B5 = find\_coef(X5, y5\_aver)  
  
 check(X5\_norm, Y5, B5, n, m)  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main(15, 3)

Text

Description automatically generated with medium confidenceText

Description automatically generated with medium confidence

**Висновок:**

В процесі лабораторної роботи було проведено трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Також було знайдено рівняння регресії, яке повинно бути адекватним для опису об'єкту.