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 №2

З дисципліни «Методи наукових досліджень»

За темою:

«ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З

ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:

Студент ІІ курсу ФІОТ

Групи ІВ-91

Гришин О.С.

Номер у списку - 07

ПЕРЕВІРИВ:

асистент

Регіда П.Г.

Київ 2021 р.

**Мета:** Провести двофакторний експеримент, перевірити однорідність дисперсії за критерієм Романовського, отримати коефіцієнти рівняння регресії, провести натуралізацію рівняння регресії.

**Завдання:**

1. Записати лінійне рівняння регресії.

2. Обрати тип двофакторного експерименту і скласти матрицю планування для

нього з використанням додаткового нульового фактору (хо=1).

3. Провести експеримент в усіх точках повного факторного простору (знайти

значення функції відгуку y). Значення функції відгуку задати випадковим

чином у відповідності до варіанту у діапазоні ymin ÷ ymax

ymax = (30 - Nваріанту)\*10,

ymin = (20 - Nваріанту)\*10.

Варіанти обираються по номеру в списку в журналі викладача.

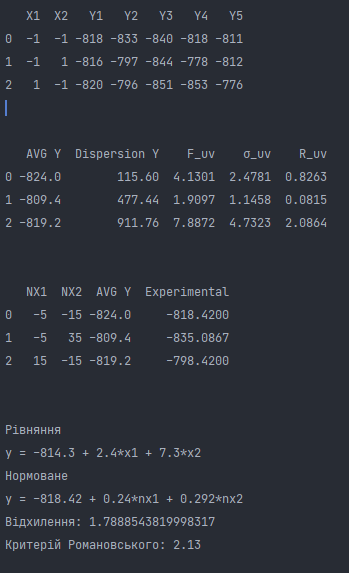
****

**Програмний код**

*if* \_\_name\_\_ == '\_\_main\_\_':  
  
 new\_r = Romanovsky(107, -5, 15, -15, 35)  
  
 new\_r.run()

*from* math *import* sqrt  
*from* random *import* randint  
  
*import* pandas *as* pd  
*from* numpy.linalg *import* det  
  
  
*class* Romanovsky:  
 m = 5  
 average\_y = []  
 dispersion\_y = []  
 f\_uv = []  
 sigma\_uv = []  
 r\_uv = []  
 deviation = 0  
 r\_value = 0  
 romanovsky\_table = {(2, 3, 4): 1.72, (5, 6, 7): 2.13, (8, 9): 2.37, (10, 11): 2.54,  
 (12, 13): 2.66, (14, 15, 16, 17): 2.8, (18, 19, 20): 2.96}  
 x1 = [-1, -1, 1]  
 x2 = [-1, 1, -1]  
  
 *def \_\_init\_\_*(*self*, var, x1\_min, x1\_max, x2\_min, x2\_max):  
 *self*.x2\_max = x2\_max  
 *self*.x2\_min = x2\_min  
 *self*.x1\_max = x1\_max  
 *self*.x1\_min = x1\_min  
 *self*.var = var  
  
 *self*.y\_min = (20 - var) \* 10  
 *self*.y\_max = (30 - var) \* 10  
  
 *self*.nx1 = [x1\_min *if self*.x1[i] == -1 *else* x1\_max *for* i *in range*(3)]  
 *self*.nx2 = [x2\_min *if self*.x2[i] == -1 *else* x2\_max *for* i *in range*(3)]  
  
 *self*.y\_1 = [randint(*self*.y\_min, *self*.y\_max) *for* \_ *in range*(*self*.m)]  
 *self*.y\_2 = [randint(*self*.y\_min, *self*.y\_max) *for* \_ *in range*(*self*.m)]  
 *self*.y\_3 = [randint(*self*.y\_min, *self*.y\_max) *for* \_ *in range*(*self*.m)]  
 *self*.y\_lists = [*self*.y\_1, *self*.y\_2, *self*.y\_3]  
  
 @staticmethod  
 *def* make\_df(f\_names, rows):  
 n\_df = {i: [] *for* i *in* f\_names}  
 *for* row *in* rows:  
 *for* i, val *in enumerate*(n\_df):  
 n\_df[val].append(row[i])  
  
 df = pd.DataFrame(data=n\_df)  
 *print*(df)  
  
 *def* \_\_dispersion\_calc(*self*, y\_list, y\_avg) -> *float*:  
 *return sum*([(i - y\_avg) \*\* 2 *for* i *in* y\_list]) / *self*.m  
  
 *def* get\_avarage\_y(*self*) -> *list*:  
 *return* [*sum*(*self*.y\_1) / *self*.m, *sum*(*self*.y\_2) / *self*.m, *sum*(*self*.y\_3) / *self*.m]  
  
 *def* get\_dispersion\_y(*self*) -> *list*:  
 *return* [*round*(*self*.\_\_dispersion\_calc(*self*.y\_lists[i], *self*.get\_avarage\_y()[i]), 4) *for* i *in range*(3)]  
  
 *def* get\_deviation(*self*) -> *float*:  
 *return* sqrt((2 \* (2 \* *self*.m - 2)) / *self*.m \* (*self*.m - 4))  
  
 *def* get\_f\_uv(*self*):  
 uv = [[*self*.dispersion\_y[0], *self*.dispersion\_y[1]], [*self*.dispersion\_y[1], *self*.dispersion\_y[2]],  
 [*self*.dispersion\_y[2], *self*.dispersion\_y[0]]]  
 *return* [*round*(*max*(uv[i]) / *min*(uv[i]), 4) *for* i *in range*(3)]  
  
 *def* get\_sigma(*self*):  
 *return* [*round*(((*self*.m - 2) / *self*.m \* f), 4) *for* f *in self*.f\_uv]  
  
 *def* get\_r\_uv(*self*):  
 *return* [*round*((*abs*(sigma - 1) / *self*.deviation), 4) *for* sigma *in self*.sigma\_uv]  
  
 *def* romanovsky\_criterion(*self*) -> *bool*:  
 *self*.average\_y = *self*.get\_avarage\_y()  
  
 *self*.dispersion\_y = *self*.get\_dispersion\_y()  
  
 *self*.deviation = *self*.get\_deviation()  
  
 *self*.f\_uv = *self*.get\_f\_uv()  
  
 *self*.sigma\_uv = *self*.get\_sigma()  
  
 *self*.r\_uv = *self*.get\_r\_uv()  
  
 *for* key *in self*.romanovsky\_table.keys():  
 *if self*.m *in* key:  
 *self*.r\_value = *self*.romanovsky\_table[key]  
 *break  
 return max*(*self*.r\_uv) <= *self*.r\_value  
  
 *def* run(*self*):  
  
 *while not self*.romanovsky\_criterion():  
 *for* i *in self*.y\_lists:  
 i.append((randint(*self*.y\_min, *self*.y\_max)))  
 *self*.m += 1  
  
 mx1, mx2, my = *sum*(*self*.x1) / 3, *sum*(*self*.x2) / 3, *sum*(*self*.average\_y) / 3  
 a1 = *sum*([i \*\* 2 *for* i *in self*.x1]) / 3  
 a2 = *sum*([*self*.x1[i] \* *self*.x2[i] *for* i *in range*(3)]) / 3  
 a3 = *sum*([i \*\* 2 *for* i *in self*.x2]) / 3  
  
 a11 = *sum*([*self*.x1[i] \* *self*.average\_y[i] *for* i *in range*(3)]) / 3  
 a22 = *sum*([*self*.x2[i] \* *self*.average\_y[i] *for* i *in range*(3)]) / 3  
  
 determinant = det([[1, mx1, mx2], [mx1, a1, a2], [mx2, a2, a3]])  
 b0 = det([[my, mx1, mx2], [a11, a1, a2], [a22, a2, a3]]) / determinant  
 b1 = det([[1, my, mx2], [mx1, a11, a2], [mx2, a22, a3]]) / determinant  
 b2 = det([[1, mx1, my], [mx1, a1, a11], [mx2, a2, a22]]) / determinant  
  
 delta\_x1 = *abs*(*self*.x1\_max - *self*.x1\_min) / 2  
 delta\_x2 = *abs*(*self*.x2\_max - *self*.x2\_min) / 2  
 x\_10 = (*self*.x1\_max + *self*.x1\_min) / 2  
 x\_20 = (*self*.x2\_max + *self*.x2\_min) / 2  
  
 nb0 = b0 - b1 \* (x\_10 / delta\_x1) - b2 \* (x\_20 / delta\_x2)  
 nb1 = b1 / delta\_x1  
 nb2 = b2 / delta\_x2  
  
 f\_names = ['X1', 'X2', \*[f"Y{i}" *for* i *in range*(1, *self*.m + 1)]]  
 rows = [[*self*.x1[i], *self*.x2[i], \**self*.y\_lists[i]] *for* i *in range*(*len*(*self*.y\_lists))]  
 *self*.make\_df(f\_names, rows)  
  
 *print*('\n')  
  
 f\_names = ['AVG Y', 'Dispersion Y', 'F\_uv', 'σ\_uv', 'R\_uv']  
 rows = [[*self*.average\_y[i], *self*.dispersion\_y[i], *self*.f\_uv[i], *self*.sigma\_uv[i], *self*.r\_uv[i]] *for* i *in  
 range*(*len*(*self*.y\_lists))]  
 *self*.make\_df(f\_names, rows)  
  
 *print*('\n')  
  
 f\_names = ['NX1', 'NX2', 'AVG Y', 'Experimental']  
 rows = [[*self*.nx1[i], *self*.nx2[i], *self*.average\_y[i], *round*(nb0 + a1 \* *self*.nx1[i] + a2 \* *self*.nx2[i], 4)] *for* i  
 *in  
 range*(*len*(*self*.y\_lists))]  
 *self*.make\_df(f\_names, rows)  
  
 *print*('\n')  
  
 *print*('Рівняння')  
 *print*(f"y = {*round*(b0, 4)} + {*round*(b1, 4)}\*x1 + {*round*(b2, 4)}\*x2")  
  
 *print*('Нормоване')  
 *print*(f"y = {*round*(nb0, 3)} + {*round*(nb1, 3)}\*nx1 + {*round*(nb2, 3)}\*nx2")  
  
 *print*(f"Відхилення: {*self*.deviation}")  
 *print*(f"Критерій Романовського: {*self*.r\_value}")

**Результати роботи програми**



**Контрольні запитання:**

1. **В теорії планування експерименту найважливішою частиною є оцінка результатів вимірів. При цьому використовують апроксимуючі поліноми, за допомогою яких ми можемо описати нашу функцію. В ТПЕ ці поліноми отримали спеціальну назву - регресійні поліноми, а їх знаходження та аналіз - регресійний аналіз.**
2. **Обирають так названу «довірчу ймовірність» p – ймовірність, з якою вимагається підтвердити гіпотезу про однорідність дисперсій. У відповідності до p і кількості дослідів m обирають з таблиці критичне значення критерію. Кожне експериментальне значення Ruv критерію Романовського порівнюється з Rкр. (значення критерію Романовського за різних довірчих ймовірностей p) і якщо для усіх кожне Ruv < Rкр., то гіпотеза про однорідність дисперсій підтверджується з ймовірністю p.**
3. **Для знаходження коефіцієнтів у лінійному рівнянні регресії застосовують повний факторний експеримент (ПФЕ). Якщо в багатофакторному експерименті використані всі можливі комбінації рівнів факторів, то такий експеримент називається повним факторним експериментом**