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 оптимізації та планування»

ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-93
Красулін Є.С. – 9316
Номер у списку: 12

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.

Варіант завдання:

Варіант	X_1		X_2	
	min	max	min	max
312	-40	20	-70	-10

Y_{min}=(30-12)*10 = 180

Y_{max}=(20-12)*10 = 80

Лістинг програми:

```
import numpy as np


class Exp:
    def __init__(self, X1_range, X2_range, Y_range, m):
        self.R_crit = {5: 2, 6: 2.16, 7: 2.3, 8: 2.43, 9: 2.5}
        self.X1_range = X1_range
        self.X2_range = X2_range
        self.Y_range = Y_range

        self.plan_matr = np.array(
            [np.random.randint(*self.X1_range, size=3),
             np.random.randint(*self.X2_range, size=3)]).T

        self.x0 = [np.mean(self.X1_range), np.mean(self.X2_range)]

        self.norm_matrix = self.norm_plan_matr()

        self.m = m

        self.experiment()

        self.b = self.find_b()
        self.a = self.find_a()

        self.check_b = self.check_b_koefs()
        self.check_a = self.check_a_koefs()

    def experiment(self):
        self.y_matrix = np.random.randint(*self.Y_range, size=(3, self.m))
        self.y_mean = np.mean(self.y_matrix, axis=1)

        self.y_var = np.var(self.y_matrix, axis=1)
        self.sigma = np.sqrt((2 * (2 * self.m - 2)) / (self.m * (self.m - 4)))

        if not self.check_r():
            print(f'\n Дісперсія неоднорідна! Змінимо m={self.m} to m={self.m+1}\n')
            self.m += 1
            self.experiment()

    def norm_plan_matr(self) -> np.array:
```

```
        self.N = self.plan_matr.shape[0]
        self.k = self.plan_matr.shape[1]

        interval_of_change = [self.X1_range[1] - self.x0[0],
                              self.X2_range[1] - self.x0[1]]
        X_norm = [
            [(self.plan_matr[i, j] - self.x0[j]) / interval_of_change[j]
             for j in range(self.k)]
            for i in range(self.N)
        ]
        return np.array(X_norm)

    def check_r(self) -> bool:
        for i in range(len(self.y_var)):
            for j in range(len(self.y_var)):
                if i > j:
                    if self.y_var[i] >= self.y_var[j]:
                        R = (abs((self.m - 2) * self.y_var[i] /
                                 (self.m * self.y_var[j]) - 1) / self.sigma)
                    else:
                        R = (abs((self.m - 2) * self.y_var[j] /
                                 (self.m * self.y_var[i]) - 1) / self.sigma)
                    if R > self.R_crit[self.m]:
                        print('Variance isn\'t stable!')
                        return False
        return True

    def find_b(self) -> np.array:
        mx1 = np.mean(self.norm_matrix[:, 0])
        mx2 = np.mean(self.norm_matrix[:, 1])

        a1 = np.mean(self.norm_matrix[:, 0] ** 2)
        a2 = np.mean(self.norm_matrix[:, 0] * self.norm_matrix[:, 1])
        a3 = np.mean(self.norm_matrix[:, 1] ** 2)

        my = np.mean(self.y_mean)
        a11 = np.mean(self.norm_matrix[:, 0] * self.y_mean)
        a22 = np.mean(self.norm_matrix[:, 1] * self.y_mean)

        b = np.linalg.solve([[1, mx1, mx2],
                             [mx1, a1, a2],
                             [mx2, a2, a3]],
                            [my, a11, a22])
        return b

    def find_a(self) -> np.array:
        delta_x = [abs(self.X1_range[1] - self.X1_range[0]) / 2,
                   abs(self.X2_range[1] - self.X2_range[0]) / 2]
        a = [(self.b[0] - self.b[1] * self.x0[0] / delta_x[0] -
              self.b[2] * self.x0[1] / delta_x[1]),
             self.b[1] / delta_x[0],
             self.b[2] / delta_x[1]]
        return np.array(a)

    def check_b_koefs(self) -> np.array:
        return np.array([
            (self.b[0] + np.sum(self.b[1:3] * self.norm_matrix[i]))
            for i in range(self.N)])

    def check_a_koefs(self) -> np.array:
        return np.array([
            (self.a[0] + np.sum(self.a[1:3] * self.plan_matr[i]))
```

```
            for i in range(self.N)])

    def results(self) -> None:
        print('Матриця планування:\n', self.plan_matr)
        print('\nНормована матриця:\n', self.norm_matrix)
        print('\nМатриця Y:\n', self.y_matrix)
        print('\nНормовані коефіцієнти:     ', self.b)
        print('Натуралізовані коефіцієнти:', self.a)
        print('\nY середнє:                                ', self.y_mean)
        print('Перевірка нормованих коефіцієнтів:       ', self.check_b)
        print('Перевірка натуралізованих коефіцієнтів:  ', self.check_a)


if __name__ == '__main__':
    np.set_printoptions(precision=3)
    m = 6
    x1_minmax = [-40, 20]
    x2_minmax = [-70, -10]
    y_minmax = [80, 180]
    exp = Exp(x1_minmax, x2_minmax, y_minmax, m)
exp.results()
```

Результат виконання роботи:

```
Матриця планування:
 [[-38 -43]
 [-40 -43]
 [-37 -18]]

Нормована матриця:
 [[-0.933 -0.1  ]
 [-1.    -0.1  ]
 [-0.9    0.733]]

Матриця Y:
 [[150 163 164  85  80 117]
 [111 157 165  94 159 134]
 [113 147 178 148 102 147]]

Нормовані коефіцієнти:      [ -13.703 -152.5     21.3  ]
Натуралізовані коефіцієнти: [-36.137  -5.083   0.71 ]

Y середнє:                                [126.5   136.667 139.167]
Перевірка нормованих коефіцієнтів:        [126.5   136.667 139.167]
Перевірка натуралізованих коефіцієнтів:   [126.5   136.667 139.167]

Process finished with exit code 0
```

Висновок:

В даній лабораторній роботі я провів двофакторний експеримент з перевіркою дисперсій на однорідність за критерієм Романовського і отримав коефіцієнти рівняння регресії. Також провів натуралізацію рівняння регресії.

Контрольні питання:

1. Що таке регресійні поліноми і де вони застосовуються?

Регресійні поліноми – це апроксимуючі поліноми, за допомогою яких ми можемо описати функцію. Застосовуються в теорії планування експерименту.

2. Визначення однорідності дисперсії.

Опираючись на вимоги регресивного аналізу достовірне оброблення та використання вихідних даних експериментальних досліджень можливе лише тоді, коли дисперсії вимірювання функцій відгуку в кожній точці експерименту є однаковими. Дана властивість називається однорідністю дисперсії.

3. Що називається повним факторним експериментом?

ПФЕ – багатофакторний експеримент в якому використовуються всі можливі комбінації рівні факторів. NПФЕ = 2k або 3k або 5k