МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)

Лабораторная работа №5
дисциплина «Системный анализ»
по теме «Уравнение регрессии»

Выполнил: студент группы ВТ-31 Макаров Д.С.
Проверил: Полунин А. И.

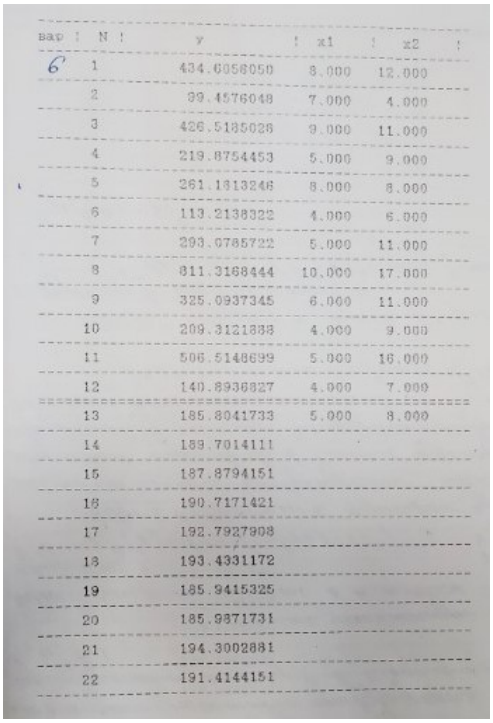
Белгород 2020

Лабораторная работа №5

«Уравнение регрессии»

Цель работы:определить функцию зависящую от 2 аргументов, найти ее коэффициенты, оценить коэффициенты и шум.

Задание: Исследуемый процесс определяется системой дифференциальных уравнений.



вар	N	y	x1	x2
6	1	434.6056050	8.000	12.000
	2	89.4576048	7.000	4.000
	3	426.5185026	9.000	11.000
	4	219.8754453	5.000	9.000
	5	261.1313246	8.000	8.000
	6	113.2138322	4.000	6.000
	7	293.0785722	5.000	11.000
	8	811.3168444	10.000	17.000
	9	325.0937345	6.000	11.000
	10	209.3121838	4.000	9.000
	11	506.5148699	5.000	16.000
	12	140.8936827	4.000	7.000
	13	185.8041733	5.000	8.000
	14	189.7014111		
	15	187.8794151		
	16	190.7171421		
	17	192.7927908		
	18	193.4331172		
	19	185.9415325		
	20	185.9871731		
	21	194.3002881		
	22	191.4144151		

Рис. 1: Задание к работе

В таблице находятся результаты эксперимента для определения функции, зависящей от двух аргументов. Эксперименты 1 – 12 использовать для оценки коэффициентов математической модели a, эксперименты 13 – 22 использовать для оценки дисперсии погрешности измерений σ_ϵ.

Ход работы

1. 4.3395 x_1x_2 + 12.63661 x_2 + -14.85473 x_1 Rr: 0.9844168175198087 Rост: 0.9935220024997524
2. 7.5885 x_1^2 + 40.28835 x_2 + -66.12757 x_1 Rr: 0.9951115915334204 Rост: 0.991580466562496
3. 2.59578 x_2^2 + -14.00299 x_2 + 26.50111 x_1 Rr: 0.9277305511370146 Rост: 0.9728269166592233
4. -3.24687 x_1^2 + 6.2375 x_1x_2 + 7.85926 x_1 Rr: 0.9840643713939677 Rост: 0.9891928211951432
5. 1.07002 x_2^2 + 2.89199 x_1x_2 + 0.95272 x_1 Rr: 1.0085387173824405 Rост: 0.9993237710303345
6. 1.93084 x_2^2 + 2.68162 x_1^2 + -2.90833 x_1 Rr: 1.0033622103033342 Rост: 0.9889365337929491
7. -2.0437 x_1^2 + 5.43998 x_1x_2 + 5.05622 x_2 Rr: 0.9667963894671152 Rост: 0.9911920896761623
8. 0.99591 x_2^2 + 2.9445 x_1x_2 + 1.14978 x_2 Rr: 0.9987387871299906 Rост: 0.9994723448920507
9. 1.89924 x_2^2 + 2.36179 x_1^2 + 0.16008 x_2 Rr: 0.962241868194272 Rост: 0.9883981051932157
10. 1.04104 x_2^2 + -0.02687 x_1^2 + 3.03552 x_1x_2 Rr: 1.025175381449643 Rост: 0.9992270864215934

Лучшие полиномы:

R^2 через QR: $x_2^2 + x_1^2 + x_1x_2$

R^2 через Qstop: $x_2^2 + x_1x_2 + x_2$

Рис. 2: Подбор уравнения регрессии при помощи программы

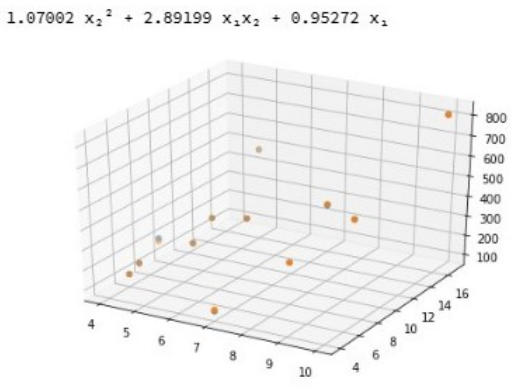


Рис. 3: Полученное уравнение регрессии

Приложение

Содержимое файла lab5regression.py

```
from itertools import combinations

import matplotlib.pyplot as plt
import numpy as np

def get_polynom_list(polynom_parts, polynom_parts_count=3):
    polynoms = []
    combination_list = combinations(range(len(polynom_parts)), polynom_parts_count)
    for combination in combination_list:
        combination = sorted(combination, reverse=True)
        result_element = []
        for part_num in combination:
            result_element.append(polynom_parts[part_num])
        polynoms.append(result_element)
    return polynoms


def get_functions_labels(functions):
    labels = []
    for func in functions:
        labels.append(func.__name__)
    return labels


def get_all_F_matrices(polynoms_list, x_vectors):
    F_matrices = []
    for polynom in polynoms_list:
        F_matrix = []
        for x_vector in x_vectors:
            F_matrix_str = []
            for polynom_part in polynom:
                F_matrix_str.append(polynom_part(x_vector[0], x_vector[1]))
            F_matrix.append(F_matrix_str)
        F_matrices.append(np.array(F_matrix))
    return F_matrices


def get_all_a_vectors(F_matrices, Y_vector):
    a_vectors = []
    for F_matrix in F_matrices:
        a_vector = np.linalg.inv((F_matrix.T.dot(F_matrix))).dot(
            F_matrix.T.dot(Y_vector)
        )
        a_vectors.append(a_vector)
    return a_vectors


def get_best_polynom(F_matrices, a_vectors, Y_vector,polynoms):
    y_cap_vectors = []
    for F, a in zip(F_matrices, a_vectors):
        y_cap_vectors.append(F.dot(a))

    R2_1_list = []
    R2_2_list = []
    Q = sum(map(lambda y: (y - np.mean(Y_vector)) ** 2, Y_vector))
    for index,y_cap_vector in enumerate(y_cap_vectors):
```

```
        QR = sum(map(lambda y_cap_i: (y_cap_i - np.mean(Y_vector)) ** 2, y_cap_vector))
        Qstop = sum(
            map(lambda y_cap_i, Y_i: (Y_i - y_cap_i) ** 2, y_cap_vector, Y_vector)
        )
        R2_1 = QR / Q
        R2_2 = 1 - Qstop / Q
        R2_1_list.append(R2_1)
        R2_2_list.append(R2_2)
        print('{}.
        ↪  '.format(index+1),get_regression_equation(get_functions_labels(polynoms[index]),a_vectors[i
        ↪  ' Rr:' ,R2_1,'Rocт: ',R2_2)
    return (R2_1_list.index(max(R2_1_list)), R2_2_list.index(max(R2_2_list)))


def get_regression_equation(polynom_labels, a_vector):
    result_str = ""
    for index, label in enumerate(polynom_labels, start=1):
        result_str += str(round(a_vector[index - 1],5))
        result_str += " " + label + " + "
    return result_str[:-3]
```