|  |
| --- |
| МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ **Кафедра програмних систем і технологій**  **Звіт**  **до лабораторної роботи № 3**  з дисципліни  **«Ймовірнісні основи програмної інженерії»**  **Студента групи ІПЗ-22 групи**  **Мукан Дмитра Сергійовича** |

**Мета роботи:**

Навчитись використовувати на практиці набуті знання про міри у двовимірній статистиці.

**Завдання:**

1. Намалюйте діаграму розсіювання для даних. Укажіть, чи існує тренд у даних. Якщо так, то вкажіть, чи є це негативним трендом, чи позитивним.  
2. Знайдіть центр ваги і коваріацію.  
3. Знайти рівняння лініїї регресії y від x.

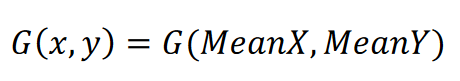
4. Розрахуйте коефіцієнт кореляції між даними.

5. Зробити висновок про залежності.

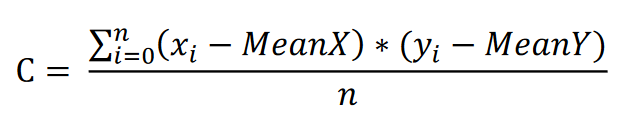
**Математична модель:**

Для побудови діаграми розсіювання та лінії регресії на графіку використовувалась бібліотека matplotlib мови програмування python.

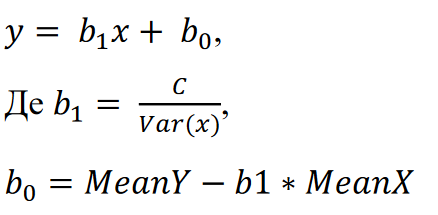
Для визначення центру ваг використовувалась формула:



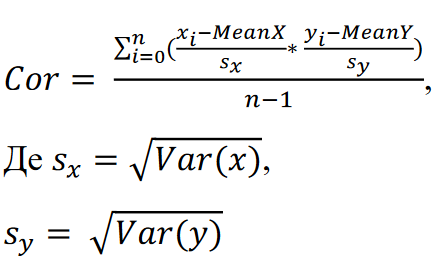
Для визначення коваріації використовувалась формула:



Для визначення рівняння регресії використовувалась формула:



Для визначення коефіцієнта кореляції використовувалась формула:



**Код алгоритму:**

import math  
  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
def all(f, res):  
 integer = []  
 flt = []  
 list = []  
 for i in f:  
 a = i.replace('\t', ' ').replace('\n', '').replace(',', '.').split(' ')  
 list.append(a)  
 for i in list:  
 for j in i:  
 if '.' in j:  
 flt.append(float(j))  
 else:  
 integer.append(int(j))  
  
 dictionary = dict(zip(flt, integer))  
 # res.write(dictionary)  
  
 one(flt, integer)  
 res.write(f'{two(flt, integer)}\n')  
 res.write(f'{three(flt, integer)}\n')  
 res.write(f'{four(flt, integer)}\n')  
  
def grav\_cent(flt: list, integer: list):  
 a = sum(flt) / len(flt)  
 b = sum(integer) / len(integer)  
 gravity\_center = []  
 gravity\_center.append(a)  
 gravity\_center.append(b)  
 return gravity\_center  
def kovariat(flt: list, integer: list):  
 n = len(flt)  
 a = sum(flt) / len(flt)  
 b = sum(integer) / len(integer)  
 sum\_temp = 0  
 for i in range(len(flt)):  
 sum\_temp += (flt[i - 1] \* integer[i - 1]) - (a \* b)  
 return sum\_temp / n  
def despertion(data):  
 n = len(data)  
 mean = sum(data) / n  
 deviations = [(x - mean) \*\* 2 for x in data]  
 variance = sum(deviations) / n  
 return variance  
  
def one(flt: list, integer: list):  
 x = flt  
 y = integer  
 colors = np.random.rand(len(x))  
 plt.scatter(x, y, c=colors, alpha=0.5)  
 plt.title('Діаграма розсіювання з позитивним трендом')  
 plt.show()  
def two(flt: list, integer: list):  
 a = grav\_cent(flt, integer)  
 b = kovariat(flt, integer)  
 return f'Grav Center: {a}\nKovariation: {b}'  
def three(flt: list, integer: list):  
 a = despertion(flt)  
 b = kovariat(flt,integer)  
 def last(flt: list, integer: list):  
 x = sum(flt) / len(flt)  
 y = sum(integer) / len(integer)  
 b1 = b/a  
 b0 = x\*b1-y  
 return f'y={b1:.3f}x-{b0:.3f}'  
  
 return f'{last(flt, integer)}'  
def four(flt: list, integer: list):  
 sx = math.sqrt(despertion(flt))  
 sy = math.sqrt(despertion(integer))  
 n = len(flt)  
 x = sum(flt)/len(flt)  
 y = sum(integer)/len(integer)  
 temp = 0  
 for i in range(len(flt)):  
 temp += ((flt[i-1]-x)/sx)\*((integer[i-1]-y)/sy)  
 r = temp/(n-1)  
 return f'Koef regretion: {r}'  
  
  
print('1(end)/10/100 ?:', end='')  
q = int(input())  
while q != 1:  
 if q == 10:  
 print('10 is done')  
 f = open('input\_10.txt', 'r')  
 res = open('res\_10.txt', 'w')  
  
 all(f, res)  
  
 f.close()  
 res.close()  
 elif q == 100:  
 print('100 is done')  
 f = open('input\_100.txt', 'r')  
 res = open('res\_100.txt', 'w')  
  
 all(f, res)  
  
 f.close()  
 res.close()  
 else:  
 print('I don`t know this type')  
 print('1(end)/10/100 ?:', end='')  
 q = int(input())

Цикл взяття з файлу інформації

integer = []  
flt = []  
list = []  
for i in f:  
 a = i.replace('\t', ' ').replace('\n', '').replace(',', '.').split(' ')  
 list.append(a)  
for i in list:  
 for j in i:  
 if '.' in j:  
 flt.append(float(j))  
 else:  
 integer.append(int(j))  
  
dictionary = dict(zip(flt, integer))

def grav\_cent(flt: list, integer: list):  
 a = sum(flt) / len(flt)  
 b = sum(integer) / len(integer)  
 gravity\_center = []  
 gravity\_center.append(a)  
 gravity\_center.append(b)  
 return gravity\_center

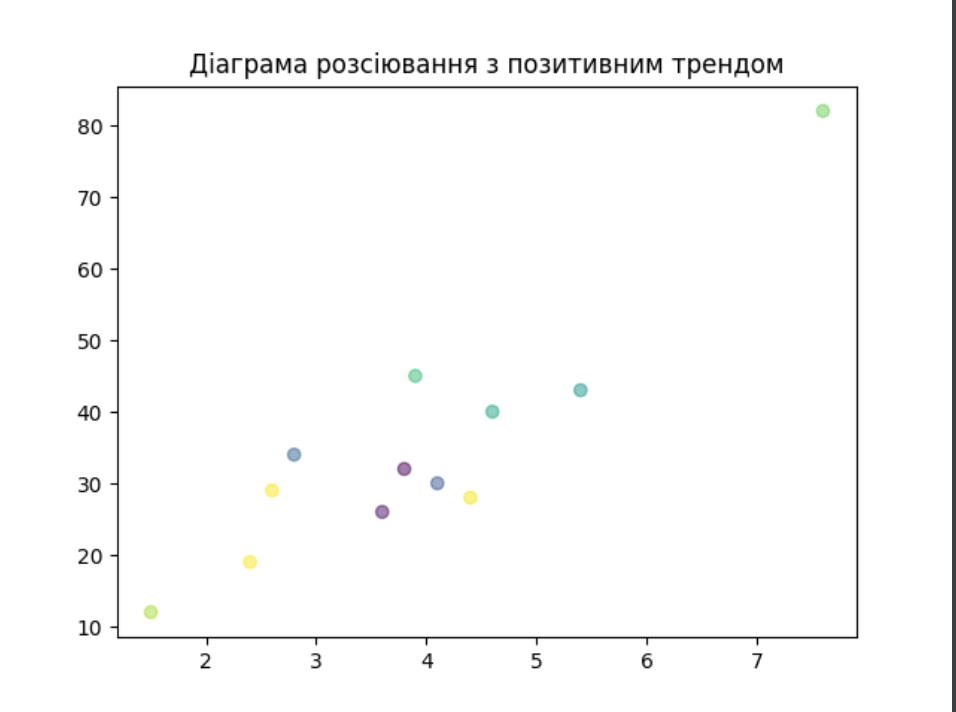
def kovariat(flt: list, integer: list):  
 n = len(flt)  
 a = sum(flt) / len(flt)  
 b = sum(integer) / len(integer)  
 sum\_temp = 0  
 for i in range(len(flt)):  
 sum\_temp += (flt[i - 1] \* integer[i - 1]) - (a \* b)  
 return sum\_temp / n

def despertion(data):  
 n = len(data)  
 mean = sum(data) / n  
 deviations = [(x - mean) \*\* 2 for x in data]  
 variance = sum(deviations) / n  
 return variance

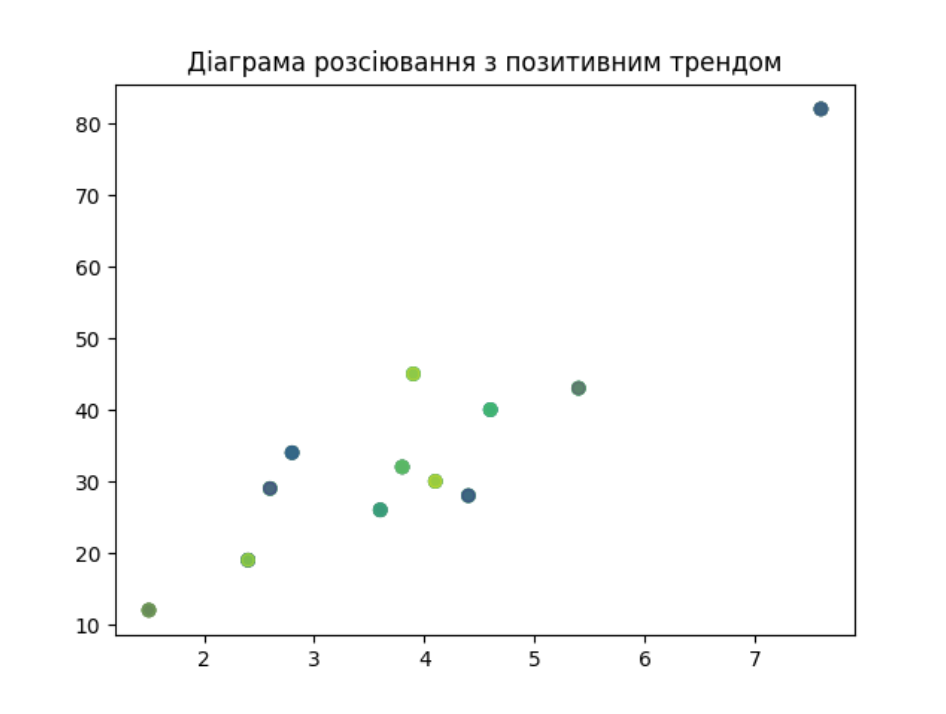
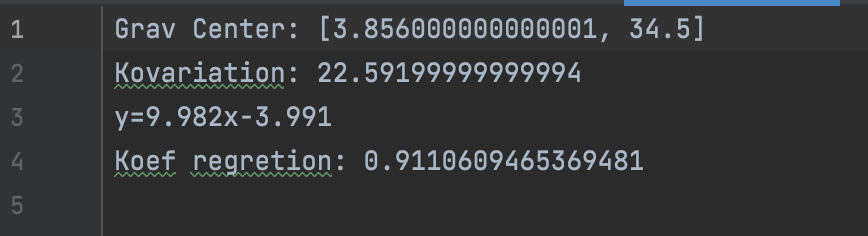
**Випробування алгоритму:**

10:

Изображение выглядит как текст

Автоматически созданное описание

100:



**Висновок:**

Протягом даної лабораторної роботи було ознайомлено з двовимірною статистикою. Було побудовано діаграму розсіювання даних, лінію регресії та розраховано центр ваги для неї за вказаними даними. Також на основі цих даних було знайдено коваріантність та коефіцієнт кореляції. Після виконання основних було підведено підсумки. Результати записані у вихідний файл і всі вимоги виконання були дотримані.