|  |  |  |  |
| --- | --- | --- | --- |
| МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка  ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  **Кафедра програмних систем і технологій**  Дисципліна  **«Ймовірнісні основи програмної інженерії»**  **Лабораторна робота № 2**  **«Лінійне перетворення та Графічне зображення даних»** | | | |
| **Виконав:** | Бойко Костянтин Богданович | **Перевірила**: |  |
| Група | ІПЗ-24(1) | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2022 | | | |

**Мета**: Навчитись використовувати на практиці набуті знання про лінійні перетворення та графічне зображення даних.

**Завдання**

1. Знайдіть Q(1), Q(3) та P(90).

2. Знайдіть середнє та стандартне відхилення цих оцінок.

3. Через незадоволення низькими оцінками викладач вирішив використати шкалу форми y = ax + b, щоб відредагувати оцінки. Він хотів, щоб середнє значення масштабних оцінок становило 95, а оцінка 100, щоб залишалася рівною 100.

4. Показати дані за допомогою діаграми "стовбур – листя".

5. Відобразити дані за допомогою коробкового графіка.

1. Знайти Q(1), Q(3) та P(90).

*Математична модель*

Для розрахунку 1-ого квартилю(25 персантиль), 3-ого(75 персантилю) та 90-ого персантилю будемо використовувати формулу:

n = k 100 (N + 1) , де

k – номер персантилю

N – кількість елементів

n – номер елементу, який є персантилем

Так ми знайшли номер елемента(індекс), який є персантилем. Тепер щоб знайти його значення нам потрібно використати наступну формулу:

Pn = xn + xдроб ∗ (xn+1 − xn), де

Pn - значення n-ого персантилю

xn – Значення елемента з вибірки, який стоїть за номером цілої частини n

xдроб – значення дробової частини n

xn+1– Значення елемента з вибірки, який стоїть за номером цілої частини n +1

*Псевдокод алгоритму*

def CalculationOfPercentile(x, SortedData): #обчислення персантилів

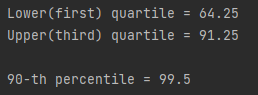
index = x \* (len(SortedData) + 1) - 1

Percentile = SortedData[int(index)] + (index % int(index)) \* (SortedData[int(index) + 1] - SortedData[int(index)])

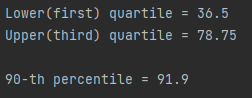
return Percentile

*Результат на консолі та випробування алгоритму*

Для 10 елементів

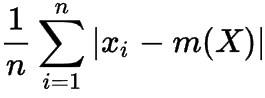


Для 100 елементів



1. Потрібно знайти *середнє відхилення*.

Для цього використаємо формулу:

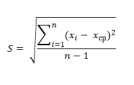
, де

n – кількість елементів

xi – поточний елемент

m(x) – середнє значення з вибірки

Також щоб знайти *стандартне відхилення* використаємо:

, де

n - кількість елементів

xi – поточний елемент

хср– середнє значення з вибірки

S – стандарте відхилення

*Псевдокод алгоритму*

def CalculateAverageDeviation(data, midleX): #cереднє відхилення

totalSum = 0

for i in range(len(data)):

totalSum += abs(data[i] - midleX)

return totalSum/(len(data))

def CalculateStandartDeviation(data, midleX): #стандартне відхилення

totalSum = 0

for i in range(len(data)):

totalSum += (data[i] - midleX)\*\*2

return np.sqrt(totalSum/(len(data)-1))

*Результат на консолі та випробування алгоритму*

Для 10 елементів

**

Для 100 елементів



1. Для виконання завдання потрібно знайти значення a та b, розвязавши систему рівнянь

, де

Max – Значення максимальної оцінки

Aver - значення бажаної середньої оцінки

x̅ – середнє значення оцінок

a і b – значення невідомих, які необхідно знайти

Тепер щоб змінити значення оцінок нам потрібно підставити a і b в формулу:

y = ax + b

де:

x – значення поточної оцінки

у – значення оцінки в кінцевому результаті

*Псевдокод алгоритму*

def CalculationOfGrades(data, midleX): #обчислення оцінок за допомогою лінійних рівнянь

ResultArray = []

GradeA = np.array([[100\*1, 1, ], [1\*midleX, 1, ]])

GradeB = np.array([100, 95])

GradeX = solve(GradeA, GradeB)

GradeA = round(GradeX[0], 2)

GradeB = round(GradeX[1], 2)

print("GradeA = " + str(GradeA) + "\nGradeB = " + str(GradeB))

File.write("GradeA = " + str(GradeA) + "\nGradeB = " + str(GradeB))

for i in range(len(data)):

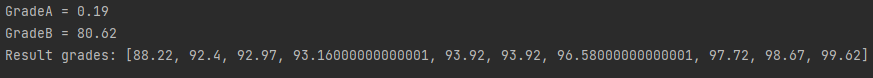
ResultArray.append(GradeA\*data[i]+GradeB)

print("Result grades: " + str(ResultArray) + "\n\n")

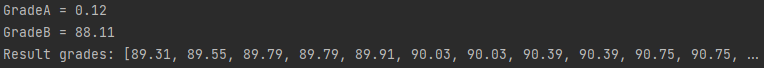
File.write("\nResult grades: " + str(ResultArray) + "\n\n")

*Результат на консолі та випробування алгоритму*

Для 10 елементів



Для 100 елементів

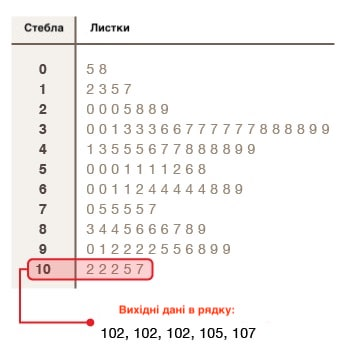


1. Створити діаграму стовбур-листя

Для створення діаграми *стовбур-листя* потрібно знайти максимальний і мінімальні елементи, після чого перший стовбчик заповнюємо значеннями цих елементів від мінімального до максимального. Зауважимо, що елементи підставляються з кроком в 10, береться тільки 1 частина числа у перший стовбчик та 2 частина числа у другий стовбчик, однак не тільки одного, а й усіх чисел в яких рівна перша частина.

Також шукаємо ключ, це буде значення першого елемента, після якого кожному наступному елементу в 1 стовпчику відповідає масив значень у другому. Для наведеного нижче прикладу ключем буде значення 0|5 = 05

Приклад того, як буде виглядати діаграма:



*Псевдокод алгоритму*

def CreateSteamLeafDiagram(Data): # діаграма "стовбур-листя"

strnumbers = ""

leaves = []

key = 0

for i in range(len(Data)):

strnumbers += str(Data[i] % 10) + " "

FDigit = int(CalculateFirstDigit(Data[i]))

j = i

if (i == len(Data)-1):

leaves.append(strnumbers)

print(CalculateFirstDigit(Data[i]) + " \t| \t" + strnumbers)

File.write("\n" + CalculateFirstDigit(Data[i]) + " \t| \t" + strnumbers)

break

elif (CalculateFirstDigit(Data[i]) != CalculateFirstDigit(Data[i + 1])):

leaves.append(strnumbers)

print(CalculateFirstDigit(Data[i]) + " \t| \t" + strnumbers)

File.write("\n" + CalculateFirstDigit(Data[i]) + " \t| \t" + strnumbers)

strnumbers = ""

FDigit += 1

while (FDigit != int(CalculateFirstDigit(Data[i + 1]))):

print(str(FDigit) + " \t| \t" + strnumbers)

File.write("\n" + str(FDigit) + " \t| \t" + strnumbers)

FDigit += 1

while (j != len(Data) and key == 0):

if (int(CalculateFirstDigit(Data[i])) == int(CalculateFirstDigit(Data[j]))-1):

key = Data[i]

break

else:

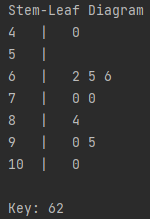
j += 1

print("\nKey: " + str(key))

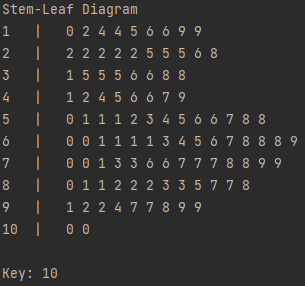
File.write("\n\nKey: " + str(key))

*Результат на консолі та випробування алгоритму*

Для 10 елементів



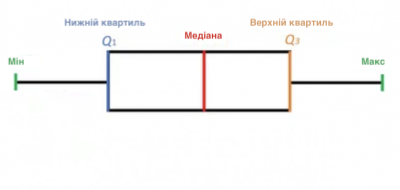
Для 100 елементів



1. Для побудови коробкового графіка нам потрібно застосувати знайдені значення у минулих завданнях, а саме:

* Медіану
* Перший та третій квартилі
* Верхня та нижня границя(мінімальне та максикальне значення вибірки)

Ось як буде зображатись графік:



**(Зауважимо,** що для побудови графіків використовуєтсья бібліотека pyplot, де за допомогою функції boxplot() передаються дані і автоматично визначаються персантилі, медіана. За допомогою цього можна перевірити правильність розрахунків у минулих завданнях, порівняти їх точність**)**

*Псевдокод алгоритму*

def GenerateBoxDiagram(Data): #коробкова діаграма

plt.figure(figsize=(10, 7))

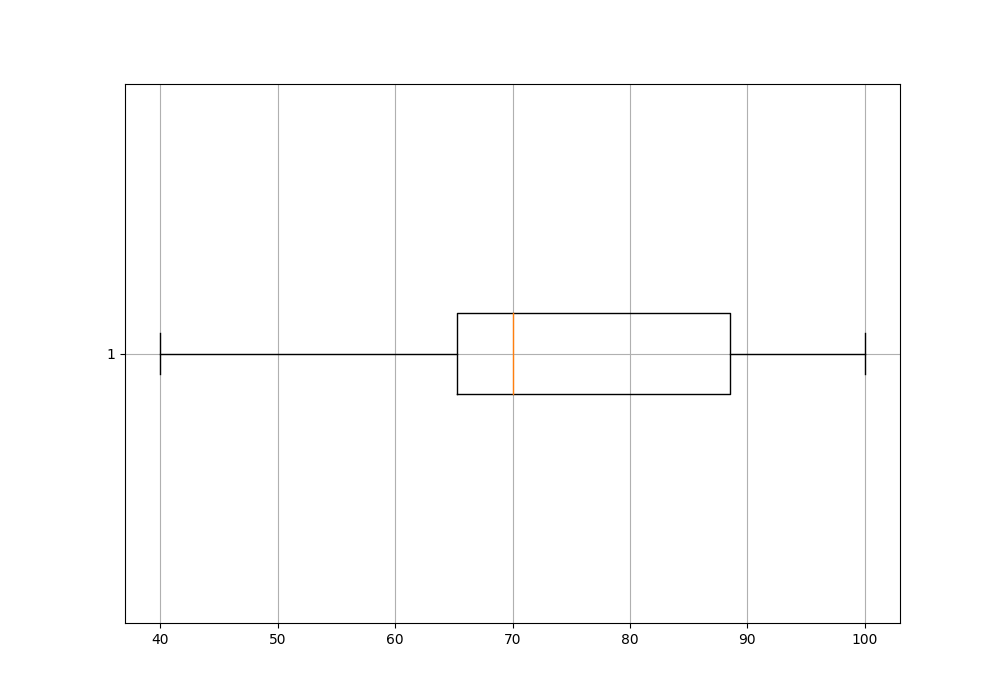
plt.boxplot(Data, vert = 0)

plt.grid()

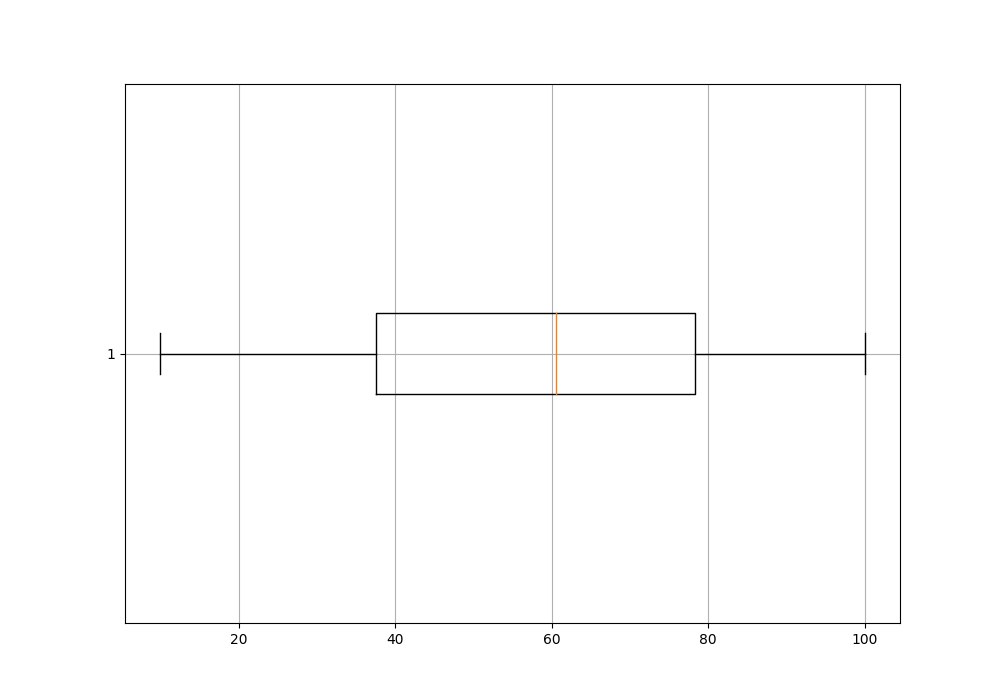
plt.show()

*Результат на консолі та випробування алгоритму*

Для 10 елементів



Для 100 елементів



**Висновок:**

Під час виконання лабораторної роботи ми навчились використовувати набуті знання про лінійні перетворення та зображення даних. Опанували навички знаходження квартилів, персантилів, стандартного та середнього відхилення. Також навчились будувати коробкові графіки та діаграму «стовбур-листя».