Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

# ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ТА КЛАСТЕРУВАННЯ З ВИКОРИСТАННЯМ БАГАТОВИМІРНОГО СОРТУВАННЯ

**Текстова частина до курсової роботи**
**за спеціальністю „Комп'ютерні науки" 122**

Керівник курсової роботи
к. ф.-м. н. Ющенко Ю.О.
_____
*(підпис)*
"____" _____ 2020 р.

Виконав студент
Крещенко Т.О.
"____" _____ 2020 р.

Київ – 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
доцент, к. ф.-м. н.
_____ Гороховський С.С.
*(підпис)*
“____” _____ 2019 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Крещенку Тарасу Олександровичу факультету інформатики 3-го курсу

ТЕМА Дослідження методів класифікації та кластерування з використанням багатовимірного сортування

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Календарний план виконання роботи
Анотація
Вступ
1 Багатовимірне сортування
2 Корисність використання багатовимірного сортування з методами класифікації або кластерування даних
3 Демонстрація використання багатовимірного сортування з кластеруванням на прикладі розробленого застосунку
Висновки
Список літератури
Додатки

Дата видачі “___” _____ 2019 р. Керівник _____
*(підпис)*
Завдання отримав _____
*(підпис)*

**Тема:** Дослідження методів класифікації та кластерування з використанням багатовимірного сортування

**Календарний план виконання роботи:**

| № п/п | Назва етапу курсової роботи | Термін виконання етапу | Примітка |
|---|---|---|---|
| 1. | Отримання теми курсової роботи | 15.10.2019 | |
| 2. | Перегляд літератури | 15.11.2019 | |
| 3. | Аналіз методів реалізації багатовимірного сортування | 12.01.2020 | |
| 4. | Написання перших двох розділів ТЧ | 01.03.2020 | |
| 5. | Створення практичної реалізації | 15.03.2020 | |
| 6. | Написання третього розділу | 01.04.2020 | |
| 7. | Написання презентації та доповіді | 10.04.2020 | |
| 8. | Корегування роботи згідно із зауваженнями керівника | 10.05.2020 | |

Студент Крещенко Т.О.

Керівник Ющенко Ю.О.

“____” _____

Зміст

Анотація

У роботі розглянуто багатовимірне адресне сортування, запропоновано способи його реалізації. Доведено корисність, насамперед в поєднанні з класифікацією або кластеруванням даних. Розроблено програму, що реалізує кластерування та демонструє наочність та корисність багатовимірного сортування.

Ключові слова: адресне сортування, сортування, впорядкування, IT, індексація, класифікація, кластерування, кластеризація, кластерний аналіз, DBSCAN

# Вступ

Багатовимірне адресне сортування – дуже цікава тема, яка безумовно є недостатньо дослідженою, на відміну від практично будь-яких інших сфер сортування даних. В час, коли дані панують всесвітом, а машинне навчання можна зустріти в будь-якій сфері діяльності, наведена тема є неймовірно актуальною, а вивчення її може бути початком нового відкриття.

Метою курсової роботи є проаналізувати багатовимірне адресне сортування, визначити його переваги та недоліки, запропонувати приклади його реалізації, довести корисність його використання в поєднанні з класифікацією або кластеруванням та показати це на прикладі.

Робота складається з трьох розділів.

У першому розділі проаналізовано та визначено багатовимірне сортування та його переваги і недоліки. Запропоновано декілька структур даних для використання в багатовимірному адресному сортуванні.

В другому розділі доведено та обґрунтовано корисність та зручність використання багатовимірного адресного сортування з класифікацією або кластеруванням.

Третій розділ присвячено розробці програми, що демонструє варіант використання алгоритму кластерування в поєднанні з багатовимірним адресним сортуванням.

# 1 БАГАТОВИМІРНЕ СОРТУВАННЯ

## *1.1 Корисність сортування*

З давніх часів люди шукали способи спростити взаємодію з різноманітними об’єктами та інформацією. Неодмінно, одним з найефективніших з них було впорядкування. Чи це розміщення книг на полиці в бажаному порядку, чи сортування інструментів на столі в тесляра, людині завжди приємніше мати справу з тим, де панує порядок. Зокрема, він пришвидшує пошук елемента, коли той є потрібним.

У сучасному світі сортування найчастіше зустрічається, коли ведеться мова про комп’ютерні дані. І це не дивно, оскільки щодня генерується все більше і більше даних в Інтернеті. Тільки за 2016-2017 рр. було згенеровано 90% на той час існуючих Інтернет-даних [1], а чим більше інформації, тим більшою є потреба її впорядкувати. Це досить легко довести, оскільки сортування надає можливість використовувати алгоритм бінарного пошуку. Він спрощує часову складність знаходження елемента в масиві з O(n) до O(log(n)), що на перший погляд може здатися невеликим покращенням, проте коли мова йде про квадрильйони одиниць даних (Big Data), без логарифмічного часу не обійтися.

## *1.2 Загальний опис багатовимірного впорядкування*

Відомо, що в IT задачу сортування досліджено практично досконало. Неодноразово було розглянуто велику кількість алгоритмів впорядкування та доведено складність цієї задачі. Тоді в чому ж полягає проблема такого сортування, до якого всі звикли? Невже прийшов час говорити про квантові комп’ютери та сортування за O(n)? На жаль, ця робота про дещо інше, але не менш суттєве.

У реальному світі інформація про певний об’єкт, як правило, є у вигляді не простих (елементарних) даних, таких як число, рядок або булеан, а у вигляді запису (англ. *record*). В мові програмування Python ця структура більш відома

як кортеж (англ. *tuple*), а саме, статичний масив елементів довільного типу. Багато кому така структура більш знайома як рядок реляції в базі даних, де міститься інформація про певний об’єкт. Коли мова йде про сортування кортежів, то, як правило, це відбувається по деякому елементу цих записів. І з цим виникає декілька проблем.

По-перше, не рідкість, коли доцільно мати можливість сортувати дані не тільки по одному певному елементу, а по багатьох. По-друге, прийнято копіювати або переміщувати дані при сортуванні, що може бути дуже ресурсомістким процесом, коли мова йде про кортежі.

Для вирішення цих проблем пропонується багатовимірне адресне сортування. Ющенко Ю. О. надав таке визначення цьому терміну: «Під багатовимірним впорядкуванням будемо розуміти сортування за багатьма ознаками чи критеріями та збереження цих результатів таким чином, що від будь-якого елемента сукупності можна швидко (миттєво, без використання пошуку та/або сортування) перейти до інформації щодо іншого елемента, який є наступним чи попереднім за однією з ознак/характеристик, за якою було здійснено сортування. Під виміром будемо розуміти ознаку, характеристику сукупності об’єктів, за якою було здійснено адресне сортування.» [2] При цьому створюється нова послідовність адрес даних, яка посилається на початковий масив, не змінюючи його. Таким чином, на виході є будь-яка кількість варіантів упорядкувань, досягнута без копіювання чи переміщення початкових даних. Різні структури даних можуть бути використані для зберігання результатів сортування для подальшого використання.

### *1.3 Розгляд та аналіз різних структур даних для використання у багатовимірному адресному сортуванні*

Найпростіша структура, яка може бути використана для зберігання варіантів впорядкування – це масив індексації. Це є масив вказівників або посилань на об’єкти в потрібному порядку. На рисунку 1.3.1 наведено приклад початкового масиву об’єктів та результат тривимірного сортування.

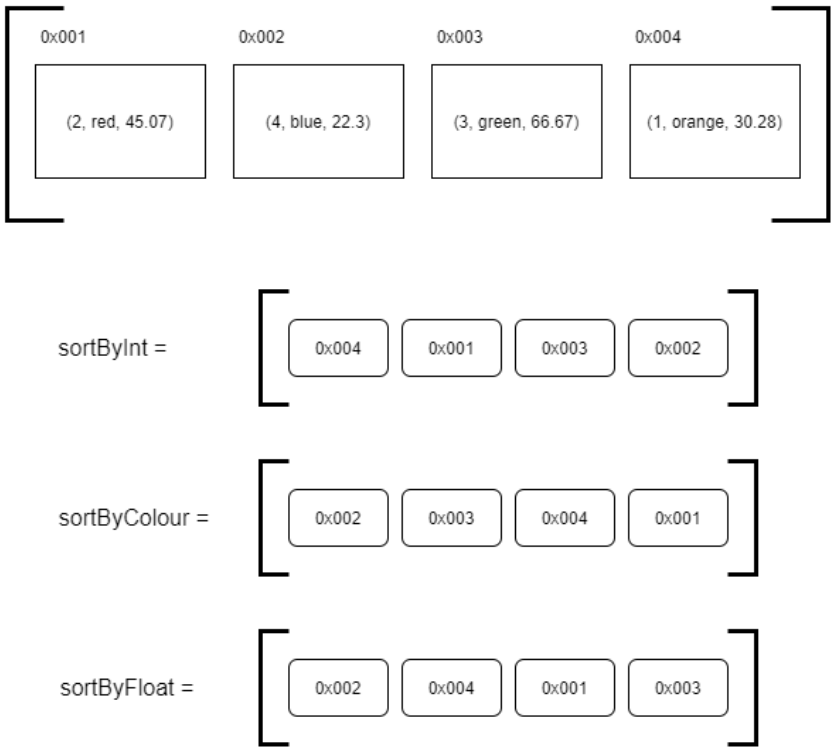


*Рисунок 1.3.1 – Візуалізація багатовимірного сортування з використанням масивів індексації*

Масиви індексації використані для збереження впорядкувань елементів. Головна перевага такої реалізації – це те, що часова складність отримання n-го елемента – O(1). Також це найпростіша з точки зору реалізації та використання структура. Недоліком є те, що перехід від елемента до йому попереднього (наступного) займає O(log(n)), якщо невідомий індекс елемента, що є цілком можливим, насамперед коли змінюється варіант сортування.

Така структура як двобічно зв’язаний список може бути корисною для переходу від певного об’єкта до наступного або попереднього по будь-якому із сортувань з часовою складністю O(1), тобто моментально. На рисунку 1.3.2 зображено UML-діаграма класу вузла, що може бути використаний для реалізації такого двозв’язного списку.

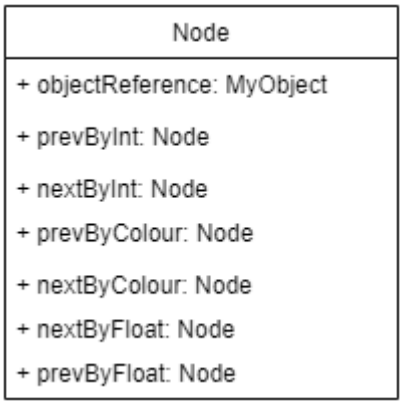


*Рисунок 1.3.2 – Приклад реалізації вузла двобічно зв'язаного списку*

У вузлі міститься відсилка на сам об'єкт, а також вказівники на наступні та попередні об'єкти для кожного з сортувань. У випадку, якщо цей об'єкт є першим або останнім, то відповідний вказівник та наступний або попередній елемент буде мати значення null. Таким чином досягається складність O(1) для переходу до наступних (попередніх) елементів. Головним недоліком такої реалізації є те, що часова складність отримання елемента за індексом – O(n).

Ще одна можлива реалізація збереження результатів багатовимірного сортування – це використання зовнішніх ключів у реляційних базах даних. На рисунку 1.3.3 зображено одну з можливих реалізацій сутності вузла.

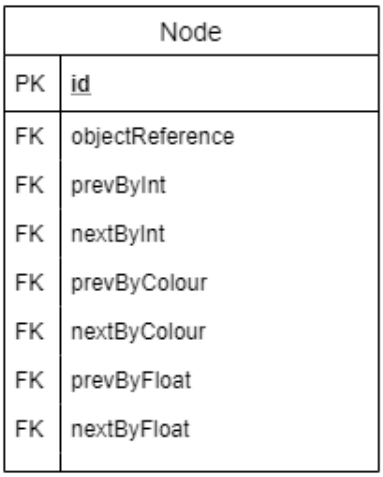


*Рисунок 1.3.3 – Приклад сутності в реляційній базі даних*

Така реалізація є дуже схожою на двобічно зв'язаний список, проте є декілька відмінностей. По-перше, наявність унікального номера в якості Головного Ключа (англ. Primary Key, PK) забезпечує можливість знаходження елемента

за індексом за O(log(n)), що є краще, ніж O(n). Перехід до наступних (попередніх) елементів все ще займає O(1). По-друге, збереження сортувань в базі даних унеможливлює втрату даних при зупинці програми, що інакше змусило б знову виконати сортування елементів. Недоліком такої реалізації, на думку автора, є можливість надмірного навантаження бази даних, у випадку якщо вона також використовується для інших потреб.

Отже, багатовимірне адресне сортування розв’язує проблему копіювання даних під час їх сортування, а також надає можливість зберігати декілька варіантів впорядкування коли це може бути потрібно. Слід зазначити, що таке сортування є зручним та корисним не тільки для обробки даних та насамперед розширення функціоналу програмного забезпечення, а також і для покращення зручності інтерфейсу користувача.

# 2 КОРИСНІСТЬ ВИКОРИСТАННЯ БАГАТОВИМІРНОГО СОРТУВАННЯ З МЕТОДАМИ КЛАСИФІКАЦІЇ АБО КЛАСТЕРУВАННЯ ДАНИХ

Машинне навчання стає все більш широковживаним. Стрімкий розвиток інформаційних технологій у XXI столітті зробив можливим застосування методів машинного навчання практично до будь-якої галузі. У випадку застосування багатовимірного сортування має сенс поєднати його з використанням методів групування об’єктів.

Серед них було розглянуто задачу класифікації, в якій відомо назви класів та потрібно визначити який об’єкт до якого класу належить; та задачу кластерування, в якій не відомо назви класів, і потрібно створити кластери об’єктів (згрупувати об’єкти) залежно від насамперед об’єктів.

Кластерування також відоме як кластерний аналіз, має за мету знаходження груп схожих об’єктів у вибірці, що зазвичай використовується для подальшого аналізу кожного кластеру. Одна з найбільших переваг багатовимірного сортування – це можливість зручно переглядати наступні та попередні по різних ознаках об’єкти відносно якогось об’єкту. Це може бути доволі корисно саме в аналізі вже створених кластерів, оскільки надається можливість аналізувати різниці в ознаках схожих об’єктів в одному кластері та наочно бачити їх близькість.

# 3 ДЕМОНСТРАЦІЯ ВИКОРИСТАННЯ БАГАТОВИМІРНОГО СОРТУВАННЯ З КЛАСТЕРУВАННЯМ НА ПРИКЛАДІ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

## *3.1 Суть проекту*

Програма полягає в кластеруванні країн світу, в яких було виявлено випадки вірусу COVID-19, а також виконання багатовимірного сортування та демонстрація її роботи.

## *3.2 Використані технології та бібліотеки*

Програму було написано на мові Python з використанням бібліотек numpy, pandas, seaborn, matplotlib, а також scikit-learn. Було взято інформацію про кількість випадків на мільйон чоловік, кількість смертей на мільйон чоловік, а також загальну кількість випадків, смертей та одужань у кожній країні, та збережено у CSV-файлі. Використано реалізацію алгоритму DBSCAN з бібліотеки scikit-learn.

## *3.3 Демонстрація роботи програми*

Спершу, під’єднується датасет (масив даних) та виводиться його опис на екран (див. Рисунок 3.3.1)

| | casesPerMil | deathsPerMil | cases | deaths | recovered | mortality |
|---|---|---|---|---|---|---|
| **count** | 211.000000 | 211.000000 | 2.110000e+02 | 211.000000 | 211.000000 | 211.000000 |
| **mean** | 937.943128 | 44.824645 | 1.760906e+04 | 1238.985782 | 5692.161137 | 0.121588 |
| **std** | 1979.824898 | 131.529691 | 9.164096e+04 | 6337.780464 | 20711.818746 | 0.173489 |
| **min** | 1.000000 | 0.000000 | 1.000000e+00 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 40.500000 | 0.000000 | 9.450000e+01 | 2.000000 | 26.000000 | 0.019608 |
| **50%** | 196.000000 | 4.000000 | 7.290000e+02 | 14.000000 | 232.000000 | 0.063258 |
| **75%** | 990.500000 | 24.000000 | 5.207500e+03 | 109.500000 | 1618.000000 | 0.142857 |
| **max** | 17554.000000 | 1222.000000 | 1.243456e+06 | 72800.000000 | 165659.000000 | 1.000000 |

*Рисунок 3.3.1 – Опис датасету*

На рисунку 3.3.2 можна більш детально побачити записи про кожну країну в датасеті.

| | country | casesPerMil | deathsPerMil | cases | deaths | recovered | mortality |
|---|---|---|---|---|---|---|---|
| 0 | Abkhazia | 12 | 4 | 3 | 1 | 2 | 0.333333 |
| 1 | Afghanistan | 105 | 3 | 3392 | 104 | 458 | 0.185053 |
| 2 | Albania | 292 | 11 | 832 | 31 | 570 | 0.051581 |
| 3 | Algeria | 113 | 11 | 4997 | 476 | 2197 | 0.178077 |
| 4 | Andorra | 9685 | 593 | 751 | 46 | 514 | 0.082143 |
| 5 | Angola | 1 | 0 | 36 | 2 | 11 | 0.153846 |
| 6 | Anguilla | 202 | 0 | 3 | 0 | 3 | 0.000000 |
| 7 | Antigua and Barbuda | 249 | 31 | 24 | 3 | 11 | 0.214286 |
| 8 | Argentina | 111 | 6 | 5007 | 264 | 1459 | 0.153221 |
| 9 | Armenia | 941 | 16 | 2782 | 47 | 1111 | 0.040587 |
| 10 | Artsakh | 47 | 0 | 7 | 0 | 0 | 0.000000 |
| 11 | Aruba | 899 | 18 | 101 | 2 | 89 | 0.021978 |
| 12 | Australia | 264 | 4 | 6794 | 97 | 5980 | 0.015962 |
| 13 | Austria | 1751 | 68 | 15589 | 608 | 13639 | 0.042676 |
| 14 | Azerbaijan | 205 | 3 | 2127 | 28 | 1536 | 0.017903 |
| 15 | Bahamas | 231 | 29 | 89 | 11 | 26 | 0.297297 |
| 16 | Bahrain | 2410 | 5 | 3842 | 8 | 1860 | 0.004283 |
| 17 | Bangladesh | 70 | 1 | 11719 | 186 | 1403 | 0.117055 |
| 18 | Barbados | 286 | 24 | 82 | 7 | 47 | 0.129630 |
| 19 | Belarus | 2047 | 12 | 19255 | 112 | 4388 | 0.024889 |
| 20 | Belgium | 4403 | 723 | 50781 | 8339 | 12731 | 0.395776 |
| 21 | Belize | 44 | 5 | 18 | 2 | 16 | 0.111111 |
| 22 | Benin | 8 | 0 | 96 | 2 | 50 | 0.038462 |
| 23 | Bermuda | 1796 | 109 | 115 | 7 | 54 | 0.114754 |
| 24 | Bhutan | 9 | 0 | 7 | 0 | 5 | 0.000000 |
| 25 | Bolivia | 157 | 7 | 1802 | 86 | 187 | 0.315018 |
| 26 | Bosnia and Herzegovina | 590 | 24 | 1987 | 86 | 928 | 0.084813 |
| 27 | Botswana | 10 | 0 | 23 | 1 | 8 | 0.111111 |
| 28 | Brazil | 550 | 38 | 121600 | 8022 | 48221 | 0.142631 |
| 29 | British Virgin Islands | 200 | 33 | 6 | 1 | 3 | 0.250000 |

*Рисунок 3.3.2 – Перші рядки датасету*

Далі, виконується попередня обробка даних. Кластерування відбуватиметься по кількості випадків на мільйон осіб (casesPerMil) та по смертності (mortality). Створюється копія датасету, але без зайвих ознак.

На рисунку 3.3.3 можна бачити, що розподіл даних обох ознак є логарифмічним. Тому функція логарифму застосовується до датасету щоб позбутися цієї проблеми, та надалі порівнювати країни у логарифмічному співвідношенні. На рис. 3.3.4 наведено розподіл після застосування функції.

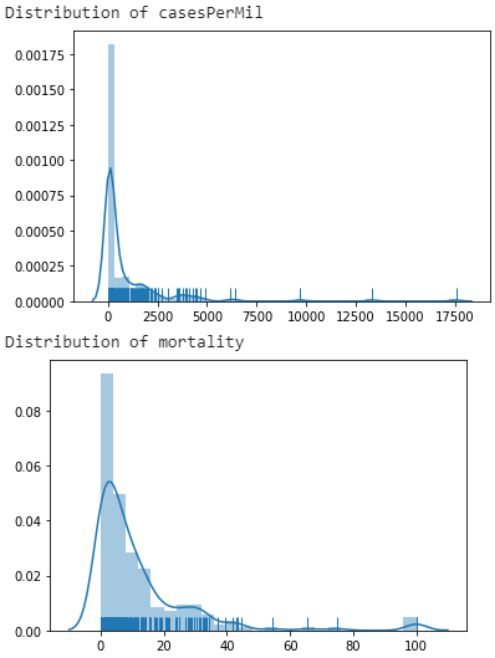


*Рисунок 3.3.3 – Розподіл даних по обраним ознакам*

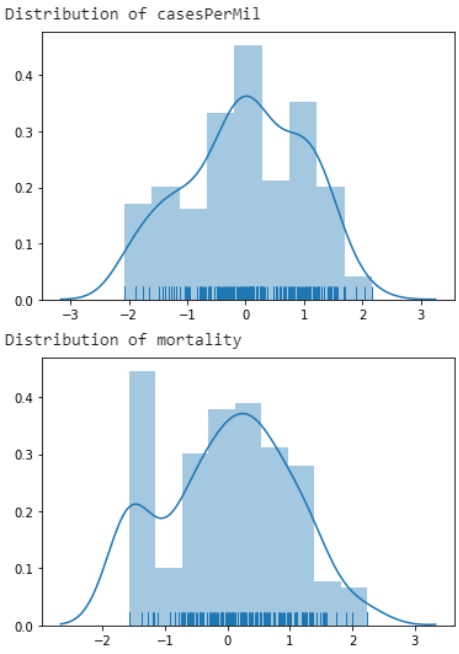


*Рисунок 3.3.4 – Розподіл даних по обраним ознакам після застосування до них функції*

Далі виконується алгоритм кластерування DBSCAN, та виводиться результат у вигляді графіка, де кожна точка має дві раніше обрані ознаки: смертність по вісі Oy та кількість випадків на мільйон осіб по вісі Ox (див. Рисунок 3.3.5). Різними кольорами позначено різні кластери.

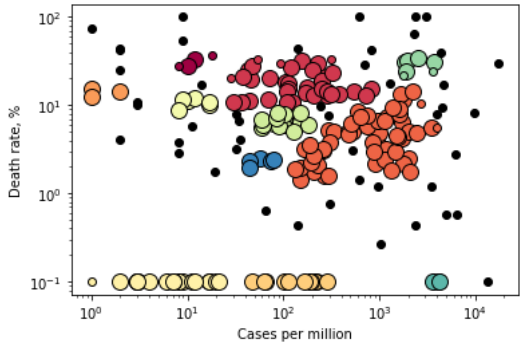


*Рисунок 3.3.5 – Графік створених кластерів*

На рисунку 3.3.6 наведено вивід перших кількох кластерів на екран, а також середню смертність кожного з них.

```
Cluster 0: ['Abkhazia' 'Chad' 'DR Congo' 'Sudan']
Average number of cases per million: 12.000
Average mortality: 31.558%
----------------------------------------------------------------
Cluster 1: ['Afghanistan' 'Algeria' 'Antigua and Barbuda' 'Argentina' 'Bahamas'
 'Bangladesh' 'Barbados' 'Belize' 'Bolivia' 'Brazil'
 'British Virgin Islands' 'Bulgaria' 'Colombia' 'Congo'
 'Dominican Republic' 'Egypt' 'Equatorial Guinea' 'Eswatini' 'Guatemala'
 'Guyana' 'Hungary' 'India' 'Indonesia' 'Jamaica' 'Japan' 'Lebanon'
 'Liberia' 'Mali' 'Mexico' 'Northern Mariana Islands' 'Philippines'
 'Poland' 'Romania' 'Sierra Leone' 'Somalia' 'Turks and Caicos Islands'
 'Ukraine']
Average number of cases per million: 197.027
Average mortality: 18.800%
----------------------------------------------------------------
Cluster 2: ['Albania' 'Armenia' 'Aruba' 'Australia' 'Austria' 'Azerbaijan' 'Belarus'
 'Bermuda' 'Bosnia and Herzegovina' 'Canada' 'Cape Verde' 'Cayman Islands'
 'Chile' 'Costa Rica' 'Croatia' 'Cyprus' 'Czech Republic' 'Denmark'
 'Finland' 'Georgia' 'Germany' 'Guam' 'Guernsey' 'Guinea' 'Iran' 'Israel'
 'Jersey' 'Kazakhstan' 'Kosovo' 'Kuwait' 'Kyrgyzstan' 'Latvia'
 'Liechtenstein' 'Lithuania' 'Malaysia' 'Maldives' 'Mauritius' 'Moldova'
 'Monaco' 'Montserrat' 'New Zealand' 'North Macedonia' 'Northern Cyprus'
 'Norway' 'Peru' 'Russia' 'Saudi Arabia' 'Serbia' 'Slovakia' 'South Korea'
 'Switzerland' 'Turkey' 'U.S. Virgin Islands' 'United Arab Emirates'
 'Uruguay']
Average number of cases per million: 1097.745
Average mortality: 5.004%
----------------------------------------------------------------
Cluster 3: ['Angola' 'Burundi' 'Mauritania']
Average number of cases per million: 1.333
Average mortality: 14.057%
```

*Рисунок 3.3.6 – Вивід перших чотирьох кластерів на екран*

Далі було реалізовано багатовимірне сортування. Створено шість масивів для сортування по кожній ознаці, кожен з яких містить масиви індексації для кожного кластера. Для цього було використано функцію pandas.Index.argsort() (див. Рисунок 3.3.7). На рисунку 3.3.8 зображено виведення результату запиту про країни з восьмого кластеру, впорядковані по смертності.

```
sorted_casesPerMil = []
sorted_deathsPerMil = []
sorted_cases = []
sorted_deaths = []
sorted_recovered = []
sorted_mortality = []
for i in range(n_clusters_):
  sorted_casesPerMil.append(pd.Index(data.get('casesPerMil').to_numpy()[labels == i]).argsort())
  sorted_deathsPerMil.append(pd.Index(data.get('deathsPerMil').to_numpy()[labels == i]).argsort())
  sorted_cases.append(pd.Index(data.get('cases').to_numpy()[labels == i]).argsort())
  sorted_deaths.append(pd.Index(data.get('deaths').to_numpy()[labels == i]).argsort())
  sorted_recovered.append(pd.Index(data.get('recovered').to_numpy()[labels == i]).argsort())
  sorted_mortality.append(pd.Index(data.get('mortality').to_numpy()[labels == i]).argsort())

print("Countries from cluster 8 sorted by mortality:")
clus = 8
c = data.to_numpy()[labels == clus][sorted_mortality[clus]]
for i in range(len(sorted_mortality[clus])):
  print("{}: {:5.3f}%".format(c[i][0], c[i][6]*100))
```

*Рисунок 3.3.7 – Реалізація багатовимірного сортування*

```
Countries from cluster 8 sorted by mortality:
Panama: 21.851%
Sint Maarten: 24.138%
Italy: 24.147%
United States: 30.529%
Ecuador: 31.367%
France: 32.350%
Portugal: 34.408%
```

*Рисунок 3.3.8 – Результат запиту після виконання сортування*

Щоб переглянути виконану роботу онлайн див. Додаток А.

# Висновки

Багатовимірне адресне сортування в поєднанні з кластеруванням – це чудовий та ефективний спосіб наочно показати результат кластерного аналізу. Розроблена програма досить гарно демонструє потенціал використання такого сортування на практиці.

В перспективі є розробка програмного продукту з графічним інтерфейсом, який може показати наскільки зручним у використанні може бути багатовимірне сортування з точки зору інтерфейсу користувача. Можливе створення такої реалізації не на площині, а в просторі, використовуючи окуляри віртуальної чи доповненої реальності.

# Список Використаної Літератури

1. IBM Marketing Cloud. 10 key marketing trends for 2017 and ideas for exceeding customer expectations. [Електронний ресурс] – Режим доступу : https://paulwriter.com/wp-content/uploads/2017/10/10-Key-Marketing-Trends-for-2017.pdf
2. Ющенко Ю. О. Багатовимірне впорядкування та його використання для вдосконалення інтерфейсу користувачів інформаційних систем [Електронний ресурс] / Ю. О. Ющенко – Режим доступу : http://ekmair.ukma.edu.ua/bitstream/handle/123456789/14638/Yushchenko_Bahatovymirne_vporiadkuvannia_ta_yoho_vykorystannia.pdf

Додаток А

Посилання на практичну роботу

https://colab.research.google.com/drive/15ngRaaCOMJ7woa6bTpL0Iyhdv16GNPZd?usp=sharing

Додаток Б

Перелік прийнятих скорочень

| | |
|---|---|
| IT | – інформаційні технології; |
| UML | – unified modeling language; |
| CSV | – comma separated values; |
| DBSCAN | – Density-based spatial clustering of applications. |