Міністерство освіти і наукии України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Реалізація паралельних алгоритмів кластеризації даних на мовах Python та C++

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Король Д.І. (прізвище та ініціали)

Керівник (підпис) Матійчук Л.П. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль - 2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«08» ___06_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Королю Дмитру Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Реалізація паралельних алгоритмів кластеризації даних на мовах Python та C++

Керівник роботи Матійчук Любомир Павлович, д.е.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «17» 05 2026 року № 4/9-239

2. Термін подання студентом завершеної роботи 21.06.2026 р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Огляд предметної області.
2. Аналіз та реалізація паралельних алгоритмів кластеризації
3. Використання технологій розпаралелювання та порівняльний аналіз продуктивності алгоритмів
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2.Мета, задачі дослідження. 3. Огляд предметної області. 4. Паралельні алгоритми кластеризації. 5. Використовувані програмні засоби та технології. 6. Набори даних для реалізації алгоритмів. 7. Результати реалізації алгоритмів на Python. 8, Вимірювання часу роботи алгоритмів. 9. Вимірювання швидкості роботи алгоритмів K-means++ та DBSCAN з бібліотеки Scikit-learn. 10. Результати реалізації алгоритмів на C++ (вимірювання часу). 11. Паралелізація програм із використанням технології OpenMP 12. Обчислення на GPU з використанням технології CUDA. 13. Порівняльний аналіз продуктивності алгоритмів з різними реалізаціями. 14. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 26.01.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.26	*Виконано*
2.	Підбір джерел про паралельні алгоритми		
	кластеризації даних	27.01 – 03.02.26	*Виконано*
3.	Опрацювання джерел про паралельні алгоритми		
	кластеризації даних	04.02 – 16.02.26	*Виконано*
4.	Виконання дослідження щодо розробки програмного		
	забезпечення	17.02 – 22.04.26	*Виконано*
6.	Оформлення розділу «Огляд предметної області»	24.04 – 10.05.26	*Виконано*
7.	Оформлення розділу «Аналіз та реалізація паралельних		
	алгоритмів кластеризації»	11.05 – 17.05.26	*Виконано*
8.	Оформлення розділу «Використання технологій		
	розпаралелювання та порівняльний аналіз		
	продуктивності алгоритмів»	18.05 – 30.05.26	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	31.05 – 03.06.26	*Виконано*
10.	Оформлення кваліфікаційної роботи	20.05 – 03.06.26	*Виконано*
11.	Нормоконтроль	01.06 – 05.06.26	*Виконано*
12.	Перевірка на плагіат	04.06 – 10.06.26	*Виконано*
13.	Попередній захист кваліфікаційної роботи	01.06 – 18.06.26	*Виконано*
14.	Захист кваліфікаційної роботи	23.06.26	

Студент _____ Король Д.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Реалізація паралельних алгоритмів кластеризації даних на мовах Python та C++ // Король Дмитро Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СН–41 // Тернопіль, 2026 // С. – 54, рис. – 21, табл. – 28, слайдів – 14, бібліогр. – 19.

Ключові слова: кластеризація, продуктивність розпаралелювання, DBSCAN, C-means, CUDA, K-means, OpenMP

Кваліфікаційна робота присвячена дослідженню алгоритмів кластеризації даних (K-mean, C-means, DBSCAN) з подальшою їх програмною реалізацією та оцінкою якості роботи із проведенням порівняльного аналізу приросту швидкості.

У першому розділі викладено основні визначення кластерного аналізу, наведено дві технології для використання паралельних обчислень (OpenMP та CUDA).

У другому розділі виконано докладний аналіз та здійснена реалізація паралельних алгоритмів кластеризації на мовах Python і C++. Проведені вимірювання часу роботи алгоритмів із оцінкою якості.

У третьому розділі проведено порівняльний аналіз приросту швидкості, зроблено висновки про отримані результати. Експериментально показано, що використання технологій організації паралельних обчислень значно прискорює роботу програмного коду.

У четвертому розділі розглянуто питання безпеки життєдіяльності і основ охорони праці.

ANNOTATION

Implementation of parallel data clustering algorithms in Python and C++ // Korol Dmytro // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2026 // P. - 54, Fig. - 21, Table - 28, Slide - 14, References - 19.

Keywords: clustering, performance, parallelization, DBSCAN, C-means, CUDA, K-means, OpenMP

The Thesis deals with the study of data clustering algorithms (K-mean, C-means, DBSCAN) with their subsequent software implementation and quality assessment of work with a comparative analysis of the speed increase.

The first section presents the basic definitions of cluster analysis, two technologies for using parallel computing (OpenMP and CUDA) are presented.

The second section provides a detailed analysis and implementation of parallel clustering algorithms in Python and C++. The running time of the algorithms is measured with a quality assessment.

The third section provides a comparative analysis of the speed increase, and conclusions are drawn on the results obtained. It is experimentally shown that the use of parallel computing technologies significantly speeds up the work of the program code.

The fourth section considers the issues of life safety and the basics of labor protection.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

CPU (англ. Central Processing Unit) – кентральний процесор, електронна схема, що виконує основні арифметичні, логічні команди, обробляє дані та керує роботою операційної системи і програм.

CUDA (англ. Compute Unified Device Architecture) — це програмно-апаратна архітектура паралельних обчислень, що дозволяє використовувати графічні процесори (GPU) для прискорення складних обчислювальних задач, які раніше виконувалися лише центральним процесором (CPU).

DBSCAN (англ. Density Based Spatial Clustering of Applications with Noise) – алгоритм кластеризації даних.

GPU (англ. Graphics Processing Unit) - графічний процесор, спеціалізована електронна схема, розроблена для швидкої паралельної обробки великих масивів даних, рендерингу 2D/3D графіки, відео, а також використання в штучному інтелекті та обчисленнях.

OpenMP (англ. Open Multi-Processing) – набір директив компілятора, бібліотечних процедур та змінних середовища, які призначені для програмування багатопоточних застосунків на багатопроцесорних системах із спільною пам'яттю на мовах C, C++ та Fortran.

Кластеризація – метод машинного навчання без учителя, який групує об'єкти у підмножини (кластери) на основі їх схожості. Елементи в одному кластері максимально схожі між собою, а об'єкти з різних кластерів — максимально відрізняються.

ЗМІСТ

ВСТУП

Актуальність теми. Із року в рік збільшуються обсяги накопиченої інформації. З одного боку, це надає нові можливості для бізнесу та науки, а з іншого — ставить перед аналітиками та дослідниками більш трудомісткі завдання. Завдяки розвитку технологій, ідеї машинного навчання, що зародилися в середині 20-го століття, стали втілюватися в сучасній реальності і вже знайшли практичне застосування в різних сферах людського життя, від автоматизації виробництва, медичної діагностики, фінансового аналізу до розважальної індустрії, творчості та повсякденних побутових потреб. Однак, зі збільшенням складності використовуваних методів і обсягу інформації, що обробляється, зростає обчислювальне навантаження, що тягне за собою витрати ресурсів, у тому числі і тимчасових. Для прискорення обробки великого обсягу даних можна придбати потужніше обладнання, але при цьому накладається висока вартість покупки та обслуговування. Альтернативним підходом є можливість розпаралелювання програмного коду на процеси, які поділяють між собою обчислення, виконуючи їх одночасно та незалежно, що дозволяє значно збільшити продуктивність та скоротити фінансові витрати.

Значної популярність при опрацюванні даних отримала мова Python, для котрої створено великий набір спеціальних бібліотек із вмонтованими алгоритмами машинного навчання. Найчастіше вони повністю або частково реалізовані мовами C/C++, Cython, Fortran і з метою підвищення продуктивності вже містять вбудовані технології для паралельних обчислень. У цьому випадку Python служить як зручний інтерфейс, який дозволяє легко запускати, розробляти та тестувати моделі машинного навчання, не вдаючись у деталі їх низькорівневої реалізації.

Мета роботи – розробка паралельних алгоритмів для кластеризації даних мовами Python і C++ та проведення порівняльного аналізу приросту швидкості виконання програм у процесі їх оптимізації.

Це може мати практичний сенс у ситуаціях, коли потрібна максимальна

продуктивність, контроль над оптимізацією, універсальність та гнучкість інтеграції в інші системи, а також коли алгоритми специфічні для конкретних завдань та не реалізовані в існуючих бібліотеках.

Для досягнення мети виділено ряд завдань:

– ознайомитися із основними визначеннями поняття «кластерного аналізу» (задачами кластеризації, формами кластерів, методами кластеризації, оцінкою її якості);

– вивчити основні поняття та принципи паралельного програмування;

– дослідити технології OpenMP та CUDA;

– провести огляд паралельних алгоритмів кластеризації K-means, C-means, DBSCAN;

– реалізувати ці алгоритми на мовах Python та C++ ;

– застосувати можливості технологій OpenMP та CUDA для організації паралельних обчислень у програмах;

– виконати порівняльний аналіз продуктивності алгоритмів для різних реалізацій.

Практичне значення одержаних результатів. Розробка може бути успішно використана під час виявлення спаму, шахрайських повідомлень, недостовірних новин, а також для таргетування цільової аудиторії та персоналізації продуктів.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Кластеризація

1.1.1 Введення до кластерного аналізу

У міру розвитку методів обробки даних аналізувати окремо взяті елементи стало неефективним і набагато зручніше виявилося об'єднувати їх за будь-якими загальними критеріями — так з'явився напрямок «багатовимірний аналіз», який став новим ступенем математичної статистики (рисунок 1.1).

Рисунок 1.1 – Кластеризація

Це поняття, зі свого боку, розділилося на взаємопов'язані області, такі як:

– кластерний аналіз;
– дискримінантний аналіз;
– кореляційно - регресивний аналіз;
– факторний аналіз;
– метод основних компонентів;
– таксономія;
– розпізнавання образів та ін.

Відмінності між ними можуть полягати в методах, цілях, результатах та можливостях для використання. У цій роботі увага приділена області, яка

знайшла широке застосування в машинному навчанні та вирішує завдання, що відноситься до «методу навчання без вчителя». Мова йде про кластерний аналіз (рисунок 1.2), який є застосуванням різних метрик і методів для виділення структури даних з метою угруповання об'єктів на основі їх подібності, а також оцінку та інтерпретацію результатів [1-5].

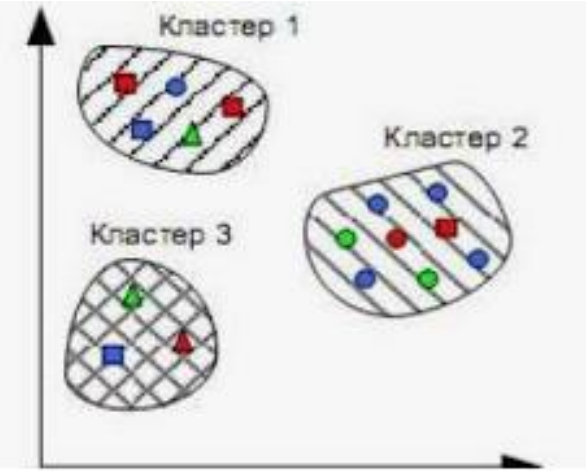

Рисунок 1.2 – Графічне відображення кластерного аналізу

1.1.2 Визначення задач кластеризації

Задача кластерного аналізу називається кластеризацією. Її постановка може відрізнятися залежно від конкретних методів та цілей. Однак, у загальному вигляді її суть полягає у поділі неоднорідної множини на визначене число умовно однорідних підмножин (кластерів). Рішенням є знайдене розбиття, яке задовольняє певну задану умову оптимальності.

На відміну від завдання класифікації, тут не використовується навчальна вибірка з відомими мітками класів, їх потрібно виявити на одному наборі даних за допомогою спеціальних алгоритмів.

1.1.3 Різноманітність задач кластеризації

Кластеризація застосовується скрізь, де потрібно виявити групи даних. Нижче наведено деякі приклади сфер та завдань, які можна вирішувати завдяки кластерному аналізу.

Маркетинг та торгівля:

– виділення цільових аудиторій серед клієнтів;
– виявлення груп товарів, які можуть купуватись разом.

Фінансові послуги:

– пошук аномалій серед транзакцій;
– групування клієнтів за ризиками неплатежів.

Освіта:

– кластеризація студентів за успішністю;
– групування курсів та програм.

Психологія:

– виявлення типу особистості з урахуванням подібних показників;
– аналіз поведінкових шаблонів.

Соціальні мережі та Інтернет:

– виявлення груп користувачів за інтересами;
– поліпшення рекомендацій на основі переваг та взаємодій.

Обробка природної мови:

– групування текстів новин за категоріями;
– виявлення плагіату.

Біологія та медицина:

– аналіз генів та виділення їх груп зі схожими функціями;
– групування пацієнтів за симптомами.

Аналіз зображень:

– виділення різноманітних об'єктів;
– розподіл зображень на основі колірних структур.

1.1.4 Форми кластерів

Кластери можуть бути різних форм, розмірів і щільностей. Найчастіше виділяють такі:

– сферичні;
– лінійні;
– вкладені;
– плоскі;

– змішані;
– щільні або розріджені;
– перехресні;
– спотворені, неправильні та незвичайні форми.

1.1.5 Методи кластеризації

Існує безліч методів для кластеризації та їх вибір залежить від особливостей конкретних даних, цілей і завдань. Вони можуть бути як самостійними, і узагальненнями чи модифікаціями інших. Використовуючи різні підходи, можна отримати різні рішення для тих самих даних.

Алгоритми кластеризації бувають різні. Варто навести деяку їх умовну класифікацію:

– за способом обробки даних: ієрархічні (агломеративні та дивізійні) або розділяючі;
– за способом аналізу даних: точні та нечіткі («жорстка» і «м'яка» кластеризація);
– залежні від шуму та викидів або ні;
– параметричні чи непараметричні;
– засновані на відстанях чи зв'язках;
– передбачають до роботи певну форму кластерів чи залежні від цього;
– за використовуваними метриками.

1.1.6 Оцінка якості кластеризації

Як було згадано у підрозділі 1.1.2, у задачах кластеризації відсутні справжні мітки, тому оцінка якості відбувається дещо складніше, ніж для класифікації чи регресії. У загальному випадку всі методи, призначені для цих цілей, можна розділити на внутрішні і зовнішні метрики.

Перші з них проводять оцінку якості кластеризації шляхом порівняння отриманих результатів між собою. Прикладом тієї, що найчастіше використовується, є індекс (коефіцієнт) силуету. Його суть полягає у вимірювання подібності об'єктів усередині одного кластера та їх відмінностей з

об'єктами з інших кластерів. Існують і інші метрики, включаючи: коефіцієнт Девіда-Болдуїна, індекс Данна, індекс Халінскі, індекс середнього діаметра. Усі вони володіють певними своїми особливостями та обмеженнями. Зокрема, у роботі [6] представлені результати дослідженням властивостей для 11 внутрішніх метрик оцінки якості кластеризації, що застосовуються достатньо широко, та наведені висновки щодо їх роботи із різноманітними даними.

Зовнішні метрики – це такі способи оцінки якості кластеризації, при яких перевіряється, наскільки отримані значення збігаються з деякими відомими мітками кластерів. Наприклад, алгоритми можна протестувати на синтетичних даних із заздалегідь відомою структурою або провести порівняння з наявними результатами, отриманими іншими методами.

1.2 Пришвидшення обчислень за допомогою паралельного програмування

1.2.1 Основні поняття та принципи паралельного програмування

У більшості класичних підручників із програмування наводяться приклади послідовного коду, в якому команди виконуються одна за одною. Такий спосіб є традиційним, простим, передбачуваним і легко піддається налагодженню, однак він має свої обмеження, особливо при вирішенні завдань, що потребують значних обчислювальних ресурсів, наприклад, при обробці великих даних. Альтернативним підходом є паралельне програмування, у якому задача поділяється на підзадачі і виконуються вони одночасно. Це дозволяє ефективно та оптимально використовувати обчислювальні системи.

Існує безліч інструментів, бібліотек, фреймворків та моделей, які дозволяють перейти від послідовного коду до паралельного.

1.2.2 Технологія OpenMP

Одним із популярних способів організувати паралельні області у програмах є використання, яка призначена для роботи з мовами C/C++ та Fortran. За допомогою технології OpenMP спеціальних директив, допоміжних функцій та

змінних середовища розробник може явно вказати місця в коді, де основний потік створить набір підлеглих, які розподілять між собою завдання. Для цього спочатку програма досліджується, в ній виявляються ділянки, які підлягають паралелізації і потім обробляються інкременальним підходом (рисунок 1.3).

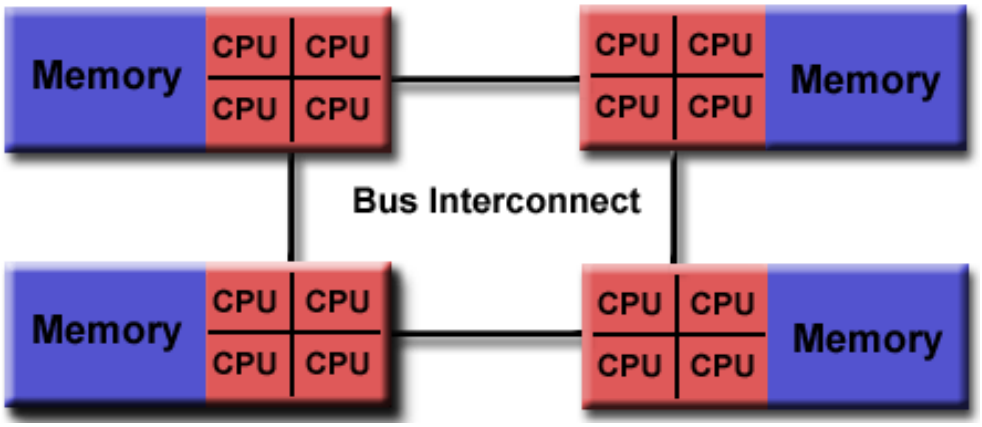

Рисунок 1.3 – Програмна модель OpenMP

Найчастіше практично розпаралелюють цикли, якщо між їх ітераціями відсутня інформаційна залежність. Дуже важливо при цьому рівномірно розподілити роботу між нитками, а також передбачити синхронізацію, щоб уникнути конфліктів роботи потоків та неправильного звернення їх до пам'яті з даними [7].

1.2.3 Технологія CUDA

Ще одним способом значно підвищити продуктивність роботи програм є використання відеокарт компанії NVIDIA, які, окрім обробки графіки, здатні взастосовуватися як математичний співпроцесор для CPU. Таку можливість надає технологія CUDA, яка є частиною класу GPGPU і призначена для реалізації паралельних обчислень на ядрах пристрою (рисунок 1.4). Вона підтримується мовами C/C++, Python і Fortran і дозволяє за допомогою спеціального синтаксису доповнити послідовну програму таким чином, щоб організувати пересилання даних із центрального процесора на відеокарту, прискорено зробити безліч нескладних операцій і повернути отриманий результат назад.

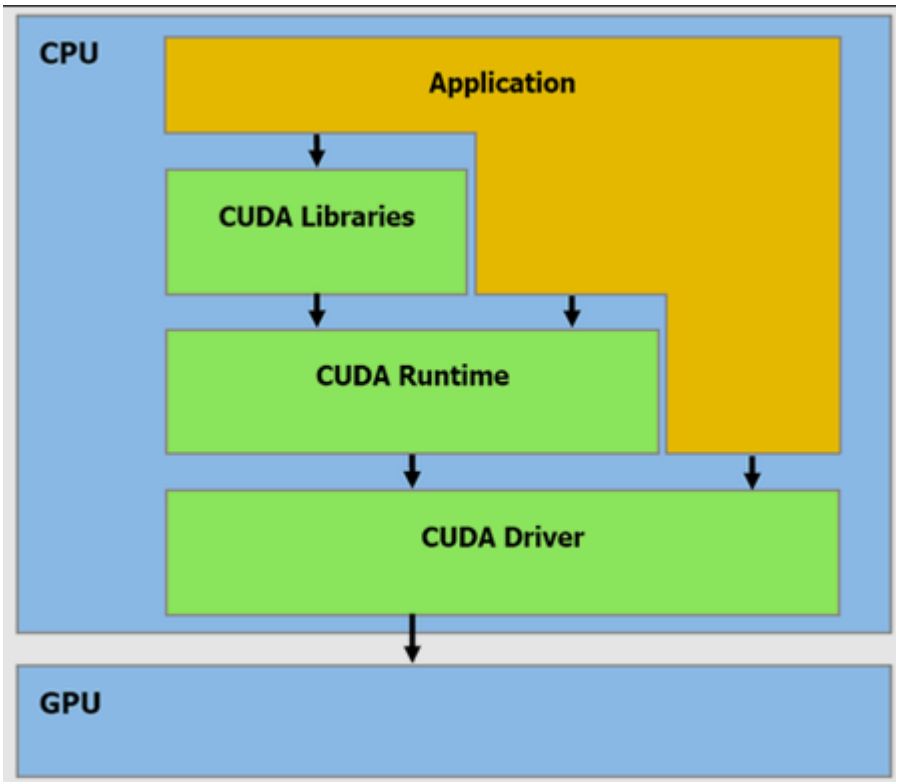

Рисунок 1.4 – Взаємодія CPU та GPU

Всі процеси, що запускаються на відеокарті, видаються такою архітектурою: нитки формуються в блоки, комплектуючись по 32 одиниці в варпи, а блоки утворюють сітку.

Кількість потоків та його розподіл розробник задає явно під час написання функцій-ядер, які виконуватимуться у пристрої [8, 9].

РОЗДІЛ 2. АНАЛІЗ ТА РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНИХ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ

2.1 Огляд алгоритмів K-means, C-means, DBSCAN

Для дослідження та реалізації програм обрано три популярні методи кластеризації:

– K-means;

– Fuzzy C-means;

– DBSCAN.

Розглянемо кожен із них окремо.

2.1.1 Алгоритм K-means /К-середніх (алгоритм Ллойда)

Є одним із найпростіших, але таких, що застосовуються у практиці методів. Його суть полягає в ітеративному переміщенні координат центрів у просторі шляхом перегляду належності точок даних до кластерів доти, доки не виконається умова зупинки. Кількість кластерів необхідно спочатку задати, від правильності вибору цього параметра залежить якість результату. Якщо ця кількість невідома заздалегідь, то існують способи їх визначення, наприклад, «метод ліктя».

Алгоритм K-means відноситься до методів точної кластеризації, нетерпимий до викидів, збіжність залежить від початкового положення центроїд. Найкраще працює зі сферичними кластерами подібних розмірів з яскраво вираженим розподілом Гауса.

Існують різні модернізації цього алгоритму, у тому числі його покращена версія K-means++, в якій тільки перший центр вибирається випадково, а наступні обчислюються, створюючи їх рівномірний розподіл по всьому простору [10].

Кроки алгоритму K-means:

1. Випадковим чином вибираються початкові центри $\{C_1, \ldots, C_k\}$.
2. Кожна точка набору даних присвоюється найближчому центру кластера на основі обраної метрики відстані:

$$j = argmin\|x_i - C_j\|, \quad (2.1)$$

де i - індекс точки; j - індекс найближчого кластера.

3. Центри перераховуються як середнє значення координат усіх точок, окремо для кожного кластера:

$$C_j = \frac{1}{S_i}\sum_{x_i \in S_i} x_i \,, \quad (2.2)$$

де S_i - множина всіх точок, що належать кластеру з індексом j.

Кроки 2 і 3 повторюються до того часу, аж поки відстань між обчисленими новими та попередніми центрами не виявиться меншою, ніж задане порогове значення, або поки не буде досягнуто максимальної кількості ітерацій.

На рисунку 2.1 представлено процес роботи алгоритму від початкової ініціалізації до отримання результату кластеризації.

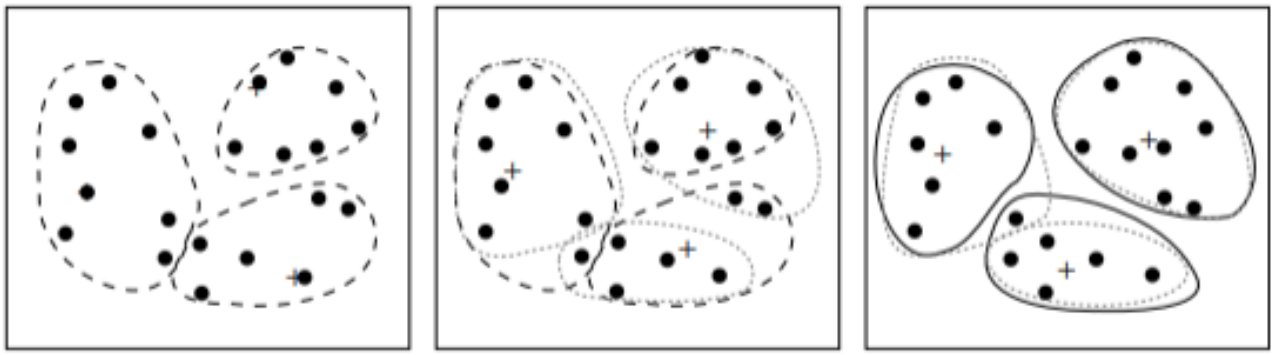

Рисунок 2.1 – Процес роботи алгоритму K-means

2.1.2 Fuzzy C-means / Алгоритм нечіткої кластеризації С-середніх

Цей метод дуже схожий на розглянутий вище K-means, винятком є той факт, що в результаті для кожної точки визначаються ступеня її приналежності до всіх кластерів. Це значення лежить в інтервалі від 0 до 1 і не є ймовірністю. Близькість до одиниці означає високий рівень належності точки до певного кластеру, а наближення до нуля свідчить про її віддаленість [11].

Для реалізації є два підходи.

Перший варіант - можна спочатку випадковим чином ініціалізувати матрицю ступенів приналежності, а потім на її основі порахувати центри. Цей спосіб має додаткове обчислювальне навантаження, але його корисно використовувати, якщо потрібно створити плавніший перехід між кордонами.

Другий варіант — спочатку ініціалізувати випадково перші центри, а вже на їх основі розрахувати матрицю приналежності. У цьому випадку переходи будуть трохи менш розмиті, а продуктивність вища.

Кроки алгоритму для другого способу:

1. Випадковим чином вибираються початкові центри $\{C_1, \ldots, C_k\}$.

2. Розраховується матриця ступенів належності на основі поточних центрів:

$$W_{ij} = \frac{1}{\sum_{j=1,l=1}^{k} \left(\frac{\|x_i - c_j\|}{\|x_i - c_l\|} \right)^{\frac{2}{m-1}}}. \quad (2.3)$$

3. Здійснюється перерахунок центрів кластерів на основі поточної матриці ступенів приналежності:

$$C_j = \frac{\sum_{i=1}^{n} W_{i,j}^m x_i}{\sum_{i=1}^{n} W_{i,j}^m}, \quad (2.4)$$

де $i = 1, \ldots, n$ - індекс точки; $j, l = 1, \ldots, к$ - індекси центрів; m - експоненційна вага (відображає ступінь розмитості, стандартно задається значенням 2),

$W_{ij} \in [0,1], \forall\, i, j.$

Кроки 2 і 3 повторюються, поки відстань між обчисленими новими та попередніми центрами не виявиться меншою, ніж задане порогове значення, або поки не буде досягнуто максимальної кількості ітерацій (рис. 2.2).

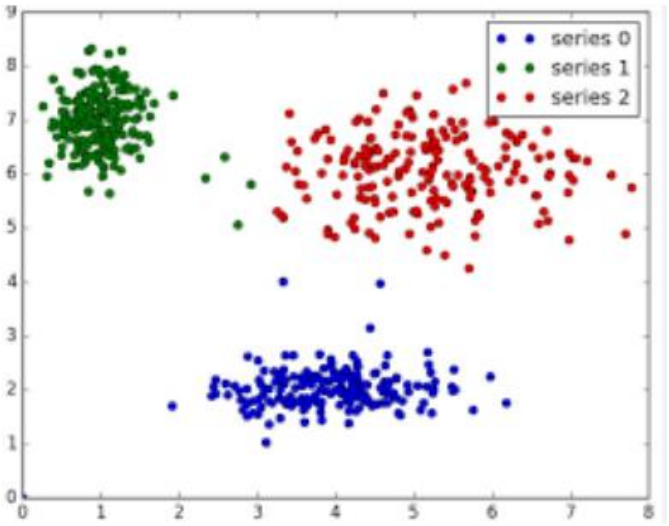

Рисунок 2.2 – Робота алгоритму Fuzzy C-means

2.1.3 DBSCAN

Цей алгоритм відрізняється від двох розглянутих раніше. Він ґрунтується на щільності і призначений для виявлення кластерів довільних форм, самостійно визначаючи їх кількість. DBSCAN не є чутливим до викидів.

У процесі роботи алгоритму виділяється три види точок: ядерні, граничні та шумові. Спочатку потрібно визначити два параметри: s (радіус околу) та MinPts (мінімальна кількість точок в околі). Їх можна підібрати із кластера з найменшою щільністю [12].

DBSCAN є ітеративним алгоритмом, в якому основна дія, що повторюється - це порівняння відстаней від досліджуваної точки до інших з даних.

Опис способу реалізації:

1. Ініціалізація індексу кластера є початковим значенням.
2. Прохід за допомогою циклу по кожній точці даних.
3. Якщо точка ще була відвідана, починається пошук її сусідів у заданої ε-околу.
4. Якщо кількість сусідів менша, ніж задане значення параметра minPts, тоді призначається мітка «шум» і здійснюється перехід до наступної.
5. Якщо така кількість більша чи дорівнює, точка вважається ядерною, їй присвоюється мітка поточного кластера і від неї починається його

розширення.

6. Після завершення розширення в кінці циклу значення кластера збільшується на одиницю.

Процес розширення кластерів показаний на рисунку 2.3.

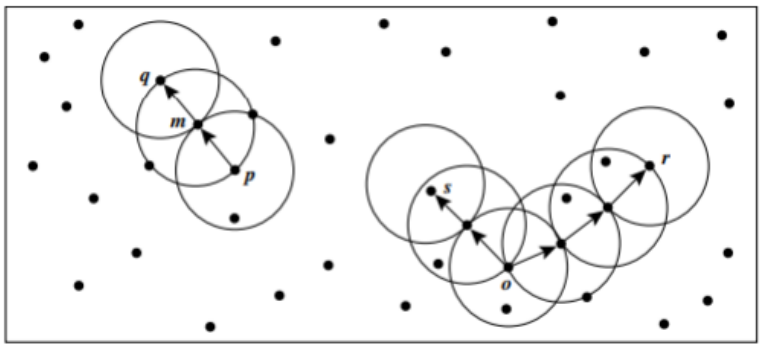

Рисунок 2.3 – Процес роботи алгоритму DBSCAN

2.2 Характеристики обчислювальної техніки

У таблиці 2.1 наведено характеристики ноутбука, який використовувався для реалізації програм, обчислень та вимірювання часу.

Таблиця 2.1 – Характеристики ноутбука

Характеристика	Значення
Модель ноутбука	Lenovo ThinkPad T450
Операційні системи	Windows 10, Linux Ubuntu 22.04.4 LTS
Процесор	Intel Core i5-5300U CPU @ 2.30GHz
Архітектура	x86_64
Кількість ядер	2
Потоків на ядро	2
Тактова частота	від 500 MHz до 2900 MHz
Оперативна пам'ять	11 GBt
Тип пам'яті	DDR3
Жорсткий диск	223.6 GBt

2.3 Реалізація алгоритмів на Python

2.3.1 Дослідження та вимірювання

Перші дослідження та вимірювання проведемо для програм, написаних мовою Python у середовищі розробки PyCharm на операційній системі Linux Ubuntu. Для реалізації були під'єднані такі бібліотеки: NumPy для роботи з математичними функціями та Matplotlib для побудови візуалізацій у вигляді графіків.

Вимірювання відстані тут і далі здійснюється за допомогою евклідової метрики:

$$\sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}, \tag{2.5}$$

де $p = (p_1, \ldots, p_n)$, $q = (q_1, \ldots, q_n)$, - точки в n-вимірному просторі,
$k \in (1,..., n)$ – номер координати.

2.3.2 Функції та їх параметри

Для алгоритму K-means написано декілька функцій:

– starting_centers - для завдання випадкових центрів;

– cluster_distibution— для розподілу точок даних за кластерами;

– relaculation_centers - для перерахунку центрів кластерів;

– k-means – основна функція роботи алгоритму, яка повертає двовимірний список із кластерами.

Використані наступні параметри:

– k – кількість кластерів;

– epsilon – порогове значення відстаней між новими та старими центрами для зупинення алгоритму;

– max_iteration – максимальна кількість ітерацій основного циклу

алгоритму.

Для вибору випадкових центрів starting_centers була використана функція random.uniform() з бібліотеки NumPy, яка генерує випадкові числа з рівномірним розподілом на заданому інтервалі. У параметри передано мінімальне та максимальне значення по першій координаті, знайдені серед усіх точок даних.

Визначення точки до кластера функції cluster_distibution відбувається за допомогою вибору її мінімальної відстані з обчислених до кожного центру.

Нові центри перераховуються у функції relaculation_centers як середнє значення всіх точок, що належать кластеру.

Основна логіка описана у функції k-means. Повторення кроків алгоритму здійснюється всередині циклу до тих пір, поки кількість ітерацій не досягне заданого числа max_iteration, або відстань між старими і новими центрами не буде меншою за ε.

Для алгоритму C-means написані такі функції:

– starting_centers - аналогічна тій, що була для K-means;

– update_membership - для розрахунку матриці ступенів належності на основі поточних центрів;

– update_centers - для перерахунку центрів на основі поточної матриці ступенів належності;

– c-means - основна функція алгоритму, яка повертає підсумкові центри та матрицю ступенів належності.

Параметри такі:

– k – кількість кластерів;

– exponential_weight – експоненційна вага;

– epsilon – граничне значення відстані між старими та новими центрами для зупинення роботи алгоритму.

Початкові центри вибираються тим самим способом, що у розглянутому раніше алгоритмі.

Основна логіка описана у функції c_means, яка містить цикл while, що працює до виконання критерію зупинки.

Для реалізації алгоритму DBSCAN написано такі функції:

– distance - для обчислення евклідової відстані;
– fing_neighbours - для знаходження сусідів для точки;
– expand_cluster - для розширення кластера;
– dbscan - основна функція для роботи алгоритму.

Параметри алгоритму: epsilon, minPts підбираються для кожного набору даних.

Обчислення евклідової відстані між двома точками відбувається в окремій функції distance, яка приймає як параметри координати двох точок.

Знаходження сусідів у fing_neighbours здійснюється за допомогою умови, яка перевіряє, що знайдена відстань між досліджуваною точкою та будь-якою іншою з набору даних менше або дорівнює заданому радіусу ε -околу.

У expand_cluster спочатку заводиться стек, у який поміщується індекс початкової точки, переданий у параметри функції. Розширення кластера відбувається завдяки послідовному вилученню в циклі while індексу точки зі стека, додавання його в словник (ключ - індекс кластера, значення - індекс точки) і дослідження точки на наявність сусідів. Якщо в точки-сусіда вистачає сусідів, щоб вважатися ядерною, то її індекс додається в стек. Після вилучення з стека всіх точок та їх обробки кластер буде вважатися завершеним.

Основна логіка описана у функції dbscan. У ній оголошуються структури зберігання даних, такі як: словник для кластерів, множина для індексів відвіданих точок, список для індексів шумових точок, а потім ініціалізується початкове значення кластера і відбувається обхід усіх точок та дослідження їхніх сусідів. Якщо кількість сусідів точки менша, ніж minPts, її індекс поміщається до списку для шумових точок, а якщо більше або дорівнює, то від неї починається розширення кластера. Функція повертає словник кластерів і список шумових точок.

2.3.3 Перевірка алгоритмів

Для перевірки точності взятий штучно сформований масив з розмірністю (100, 3), що містить координати 100 точок в тривимірному просторі. Значення зґенеровані за допомогою функції randn() модуля random як випадкові дійсні

числа, розкидані навколо центрів з координатами (0, 0, 0), (3, 3, 3), (-3, -3, -3) згідно із законом стандартного нормального розподілу. Точки утворюють три сферичні кластери: у першому – 30, у другому – 30, у третьому – 40. Знаючи точні значення центрів можна оцінити відхилення від них у обчислених.

Результати вимірювань точності роботи алгоритму K-means наведено у таблиці 2.2.

Таблиця 2.2 - Відхилення обчислених центрів алгоритму K- means

Epsilon = 0.001			**Epsilon = 0.0001**			**Epsilon = 0.00001**		
Центр 1	Центр 2	Центр 3	Центр 1	Центр 2	Центр 3	Центр 1	Центр 2	Центр 3
0.4340	0.3388	0.2480	0.3388	0.3273	0.2423	0.2480	0.4340	0.3388

У таблиці 2.3 представлено результати вимірювань точності роботи алгоритму C- means.

Таблиця 2.3 - Відхилення обчислених центрів алгоритму C- means

Epsilon = 0.001			**Epsilon = 0.0001**			**Epsilon = 0.00001**		
Центр 1	Центр 2	Центр 3	Центр 1	Центр 2	Центр 3	Центр 1	Центр 2	Центр 3
0.4242	0.2887	0.3288	0.4245	0.3288	0.2886	0.4245	0.2886	0.3288

Значення можна трактувати як хорошу роботу алгоритмів, враховуючи, що розмах всім даних становить 6 одиниць у тривимірному просторі і результати вийшли значно менші, ніж стандартне відхилення.

Для достовірності перевірки, була виміряна точність обчислених центрів для цього набору даних у вбудованого в бібліотеку scikit-learn алгоритму K-means ++ з встановленими параметрами n_dusters=3 і tol для відхилень. Отримані результати наведено у таблиці 2.4.

Таблиця 2.4 - Відхилення обчислених центрів алгоритму K- means++ з бібліотеки scikit-learn

Epsilon = 0.001			**Epsilon = 0.0001**			**Epsilon = 0.00001**		
Центр 1	Центр 2	Центр 3	Центр 1	Центр 2	Центр 3	Центр 1	Центр 2	Центр 3
0.3273	0.3388	0.2423	0.3388	0.2423	0.3273	0.3388	0.2480	0.4340

Значення вийшли схожими. Це означає, що написані програми для K-means та C- means обчислюють центри відповідно до очікуваної роботи методів.

Алгоритм DBSCAN швидше оцінити безпосередньо за графіками, результатами та кількістю «шумових» точок. Найкращий результат на тих же даних показали параметри: epsilon = 1.43, minPts = 3. Загальні результати вийшов наведено у таблиці. 2.5.

Таблиця 2.5 - Вивід результату роботи алгоритму DBSCAN на масиві з розмірністю (100,3) та параметрами: epsilon = 1.43, minPts = 3

Кількість знайдених кластерів: 3
Індекси точок за кластерами
0 : [0, 1, 2, 3, 5, 6, 7, 8, 9, 11, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 73]
1 : [31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]
2 : [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 96, 97, 99]
Кількість шумових точок: 9
Індекси шумових точок: [4, 10, 12, 14, 30, 45, 78, 91, 98]

При подальших реалізаціях результати оцінки якості будуть порівнюватися з отриманими цьому етапі для того, щоб зміни у програмах впливали на коректність роботи алгоритмів.

Далі подивимося на візуальне втілення. Для цього штучно створено три набори даних з дійсними числами, що представляють координати точок у двовимірному просторі.

Перший масив з даними містить 150 точок, випадково розкиданих навколо центрів з координатами (0, 0, 0), (3, 3, 3), (-3, -3, -3) за стандартним нормальним розподілом, і являє собою три сферичні кластери (рисунок 2.3).

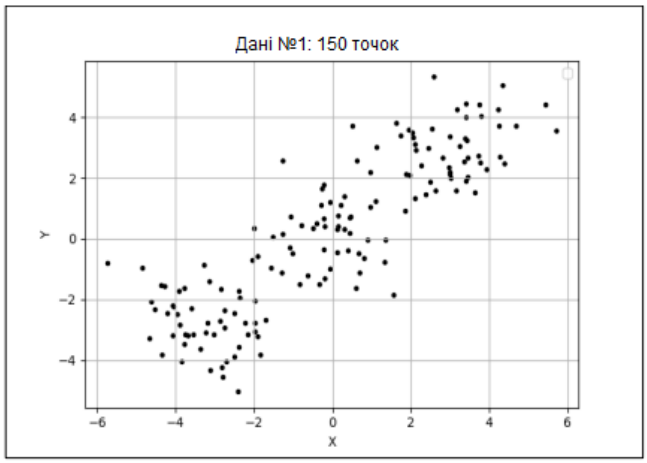

Рисунок 2.3 – Три сферичні кластери

Другий масив даних складається з 400 точок і містить три кластери довільної форми, а також шумові точки (рисунок 2.4).

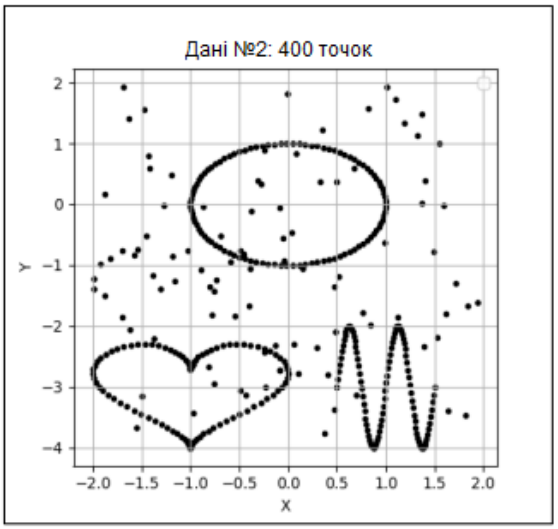

Рисунок 2.4 – Три кластери довільних форм та шумові точки

Третій масив є плоским кластером і містить 500 точок, випадково і

рівномірно розкиданих на двовимірному обмеженому просторі (рисунок 2.5).

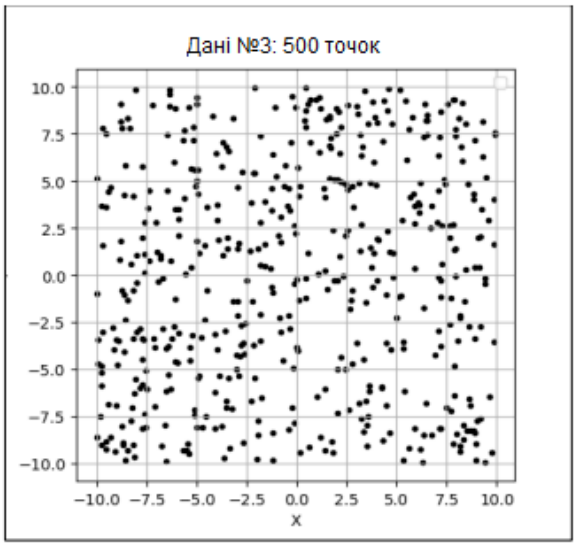

Рисунок 2.5 – Плоский кластер з рівномірним випадковим розподілом точок

Спочатку подивимося, як працює метод K-means на першому наборі даних. Нижче наведено результат у вигляді графіка, де приналежність точки до кластера виділяється кольором, а знайдені центри позначені «хрестиком» (рисунок 2.6). Встановлені параметри: k = 3, epsilon = 1e-4. Алгоритм зійшовся на 4-ій ітерації.

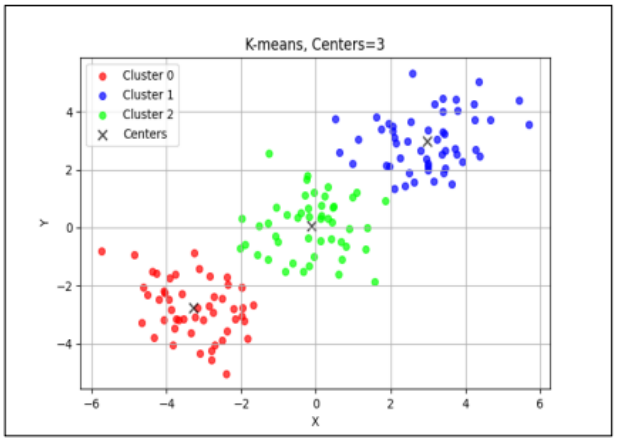

Рисунок 2.6 – Робота алгоритму K-means на першому наборі даних

Очікувано, K-means правильно розділив точки даних, оскільки він добре справляється з кластерами. Їхнє число було відомо заздалегідь, тому алгоритм відпрацював дуже добре.

Розглянемо графік роботи методу C-means (рисунок 2.7). Чіткість кольору відбиває ступінь належності точки до кластера. Параметри: k = 3, epsilon = 1e-4, m = 2. Алгоритм зійшовся на 14-ій ітерації.

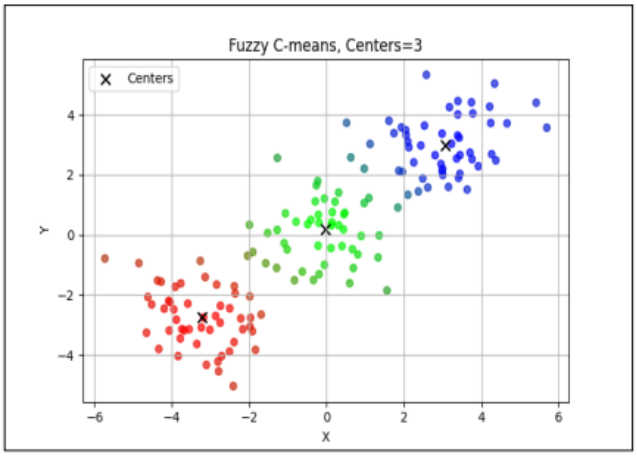

Рисунок 2.7 – Робота алгоритму C-means на першому наборі даних

За графіком можна бачити розмитість кордонів на переходах між кластерами. Алгоритм відпрацював добре і корисний, якщо не потрібен суворий поділ.

Тепер проаналізуємо візуалізацію DBSCAN (рисунок 2.8). Найкращий результат було отримано з параметрами epsilon=0.8, minPts=5.

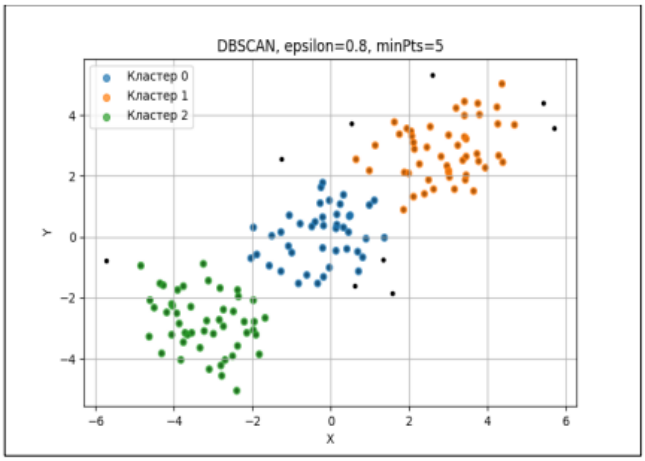

Рисунок 2.8 – Робота алгоритму DBSCAN на першому наборі даних

Точки, що залишилися незафарбованими, алгоритм відокремив як «шум», а кольором виділив найщільніші частини кластерів. Очікувано, DBSCAN теж

добре впорався із цим набором даних.

Тепер перейдемо до кластерів довільних форм та оцінимо, як у цьому випадку відпрацювали алгоритми. Першим розглянемо K- means (рисунок 2.9).

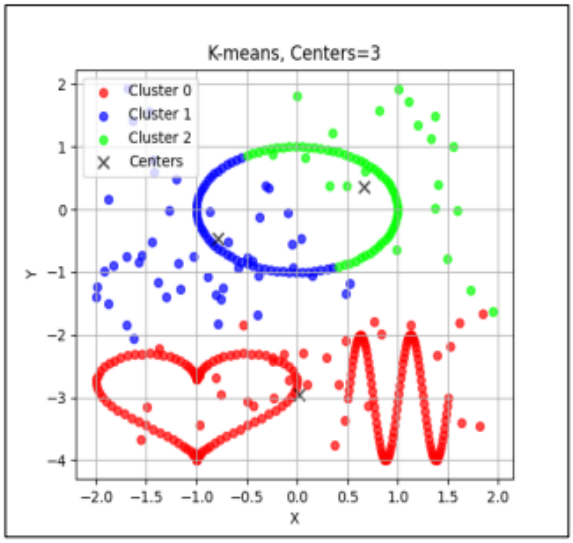

Рисунок 2.9 – Робота алгоритму K- means на другому наборі даних

Кластеризація вийшла погана. Хоча при різних запусках іноді траплялося, що обчислені центри потрапляли в середину кластерів, результат все одно не був коректним.

Подивимося, як відпрацював C-means (рисунок 2.10).

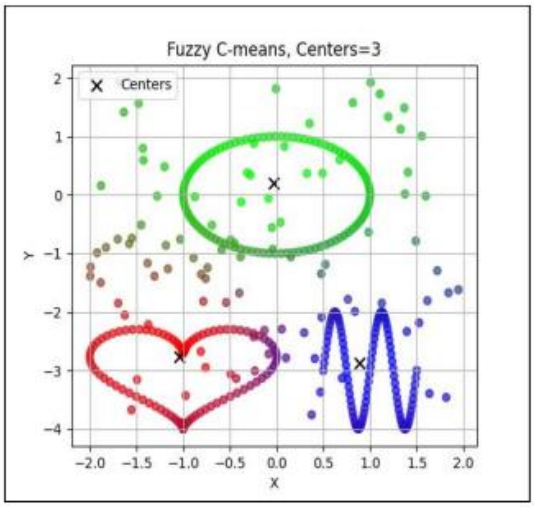

Рисунок 2.10 – Робота алгоритму C- means на другому наборі даних

При багаторазових запусках алгоритм завжди схоже визначав центри.

Зумовлено це специфікою цих даних. Можна бачити, що K- means та C- means не призначені для даних специфічних форм із шумом та викидами.

Далі подивимося, як пройшла кластеризація цих даних за допомогою методу DBSCAN (рисунок 2.11). Підібрані параметри: epsilon =0.15, minPts =3.

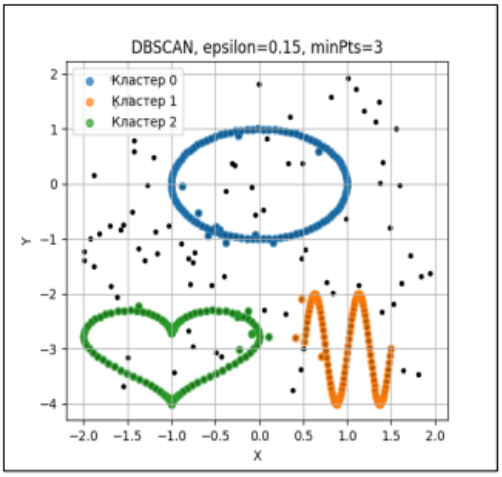

Рисунок 2.11 – Робота алгоритму DBSCAN на другому наборі даних

Результат вийшов достатньо добрим. Алгоритм успішно виділив кластери, відокремивши шумові точки. Відомо, що DBSCAN добре справляється з кластерами специфічних форм, однак має свої недоліки, пов'язані з обчислювальною складністю і підбором параметрів.

Тепер перейдемо до третього набору даних, який є плоским кластером. Почнемо з методу K-means (рисунок 2. 12). Алгоритм зупинився на 9-ій ітерації.

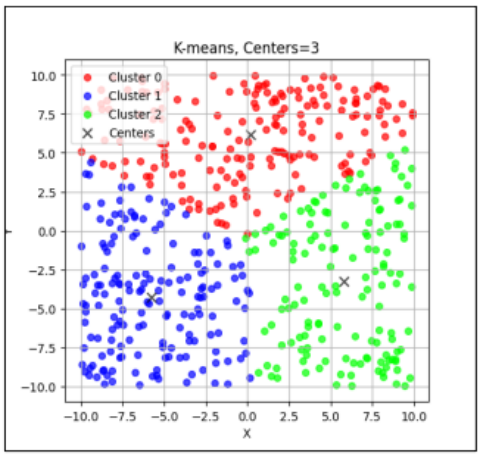

Рисунок 2.12 – Робота алгоритму K- means на третьому наборі даних

За цим прикладом можна наочно зрозуміти принцип роботи K- means - весь простір він поділив на три рівні частини з чіткими межами залежно від того, куди потрапили центри.

Поглянемо на роботу C- means (рисунок 2.13). Алгоритм зійшовся на ітерації 88 - це свідчить, що центри могли довго переміщатися малому інтервалі, перескакуючи заданий пороговий мінімум відхилення.

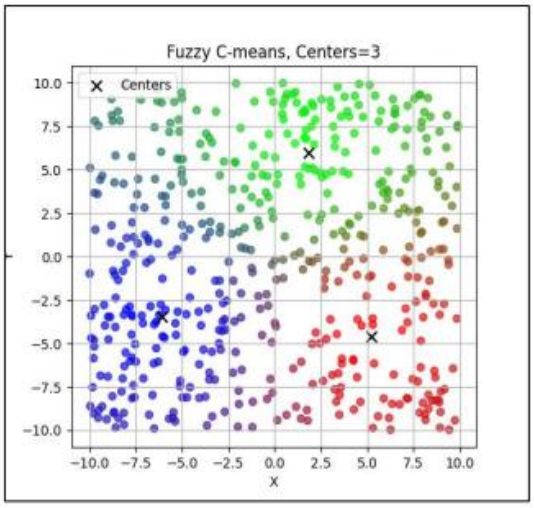

Рисунок 2.13 – Робота алгоритму C- means на третьому наборі даних

Отримали очікуваний результат, C- means рівновіддалено визначив центри по всьому просторі і можна побачити плавний перехід межі умовного поділу.

Подивимося, як із цим набором впорався алгоритм DBSCAN (рисунок 2.14). Встановлені параметри: epsilon = 1, minPts = 3.

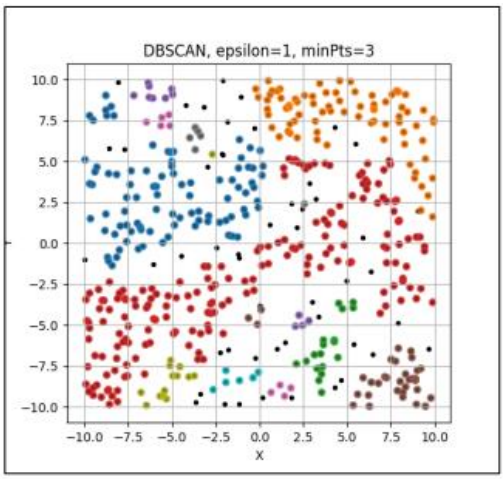

Рисунок 2.14 – Робота алгоритму DBSCAN на третьому наборі даних

За графіком видно, що DBSCAN визначив у загальні групи щільності скупчення точок. Незважаючи на те, що на цьому наборі даних свідомо не створювалися кластери, їх алгоритм непогано виділив на заданих параметрах.

В результаті перевірки реалізовані алгоритми кластеризації показали очікувану від них роботу. Вони коректно працюють і з даними інших розмірів. Отже, можна перейти до вимірювання часу.

2.3.4 Вимірювання часу роботи алгоритмів

Для наступних експериментів сформовані масиви з розмірностями (10000, 3), (100000, 3), (1000000, 3), що є координатами 10000, 100000 та 1000000 точок у тривимірному просторі. Дані згенеровані як випадкові дійсні числа, розподілені навколо центрів з координатами (0, 0, 0), (3, 3, 3), (-3, -3, -3) згідно із законом стандартного нормального розподілу. Завдання полягає в тому, щоб виміряти час роботи алгоритмів на масивах з даними великих розмірностей та провести порівняльний аналіз швидкості у міру оптимізації програм. І тут розподіл точок і якість кластеризації не є важливими.

Для вимірювання часу у програм, реалізованих мовою Python, використано бібліотеку time. Для методів K-means та C-means встановлено параметр max_iteration=10, що означає фіксовану максимальну кількість ітерацій у основного циклу, та прихована умова перевірки збіжності. Для коректності результатів вимірювання проводилися 100 разів і потім усереднювалися.

Далі представлені результати вимірювань часу роботи алгоритмів у секундах на трьох масивах для методів K-means (таблиця 2.6) та C-means (таблиця 2.7) із заокругленням дробової частини до четвертого знаку.

Таблиця 2.6 – Результати вимірювання часу для алгоритму K-means мовою Python

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	2.6457	29.0940	284.9458

Таблиця 2.7 – Результати вимірювання часу для алгоритму C-means мовою Python

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	3.7095	37.2543	372.5580

Для алгоритму DBSCAN вимірювання часу проводилися на масивах з розмірами (10000, 3) та (100000, 3). Встановлені параметри epsilon=1.43, minPts=3 вибрані виключно як загальні для всіх вимірювань. У випадку масиву на 10000 точок вимірювання проводилися 10 разів з подальшим їх усередненням. Для масиву на 100000 пікселів замір проводився один раз. Масив на 1000000 точок був виключений з дослідження через передбачувану тривалу роботу, пов'язану з дуже великою обчислювальною складністю.

Результати обчислень для DBSCAN наведено у таблиці 2.8.

Таблиця 2.8 – Результати вимірювання часу для алгоритму DBSCAN мовою Python

Розмірність масиву	(10000, 3)	(100000, 3)
Результати в секундах	739.1143	72554.6706

DBSCAN - трудомісткий алгоритм, особливо у випадку дуже великої кількості значень. З масивом на 100000 пікселів до отримання результату він відпрацював 20 годин.

Отримані значення будуть використані як початкові для подальшого покращення та відстеження приросту швидкості.

Крім цього, було виміряно швидкість роботи алгоритмів K-means++ (таблиця 2.9) та DBSCAN (таблиця 2.10) з бібліотеки Scikit-learn.

Таблиця 2.9 – Результати вимірювання часу для алгоритму K- means ++ з бібліотеки Scikit-learn

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.0134	0.1105	0.4544

Таблиця 2.10 – Результати вимірювання часу для алгоритму DBSCAN з бібліотеки Scikit-learn

Розмірність масиву	(10000, 3)	(100000, 3)
Результати в секундах	0.4478	-

Робота вбудованих у бібліотеку алгоритмів значно перевершує за швидкістю написані вручну програми на Python, оскільки там вже є паралельні обчислення. Початкові центри у k-means++ вибираються не випадково, що додатково впливає на зростання швидкості. Однак, при спробі виміряти час роботи DBSCAN зі Scikit-learn на масиві з розмірністю (100000, 3) процес завершився примусовою зупинкою через надмірне переповнення вільної пам'яті на процесорі у обчислювальної машини. І тут реалізована вручну програма виявилася кориснішою, оскільки попри тривалість, вона відпрацювала до кінця.

2.4 Реалізація алгоритмів на C++

На цьому етапі алгоритми, що досліджуються, переписані на мову C++ з використанням динамічного виділення пам'яті. Вимірювання виконані на операційній системі Windows 10 у середовиші розробки Visual Studio. Усі функції виконують ту ж обчислювальну роботу, що й у випадку реалізації мовою Python. Відмінності полягають у особливостях синтаксису і специфіці мови.

Вимірювання точності роботи K-means та C-means у порівнянні з програмами на Python показали такі ж результати, а алгоритм DBSCAN на C++ зробив розрахунки аналогічні до тих, що були на Python.

Час виміряно за допомогою бібліотеки chrono на тих самих параметрах алгоритмів. Для K-means та C-means вимірювання проводилися 100 разів, для DBSCAN 10 разів з подальшим усередненням отриманих значень.

Нижче наведено результати на мові C++ для трьох алгоритмів (K-means – таблиця 2.11, C-means- таблиця 2.12, DBSCAN – таблиця 2.13).

Таблиця 2.11 – Результати вимірювання часу для алгоритму K-means мовою C++

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.0762	0.7869	7.8177

Таблиця 2.12 – Результати вимірювання часу для алгоритму C-means мовою C++

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати за секунди	0.2842	2.8612	28.9263

Таблиця 2.13 – Результати вимірювання часу для алгоритму DBSCAN мовою C++

Розмірність масиву	(10000, 3)	(100000, 3)
Результати за секунди	6.0596	569.1956

C++ більш низькорівнева мова, ніж Python, тому обчислення нею відбуваються швидше, що можна помітити за отриманими значеннями. Робота алгоритму K-means прискорилася приблизно в 36 разів, C-means почав працювати швидше приблизно в 12 разів, а DBSCAN отримав приріст швидкості приблизно в 127 разів.

РОЗДІЛ 3. ВИКОРИСТАННЯ ТЕХНОЛОГІЙ РОЗПАРАЛЕЛЮВАННЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОДУКТИВНОСТІ АЛГОРИТМІВ

3.1 Паралелізація програм із використанням технології OpenMP

На цьому етапі приступимо до оптимізації процесу обчислень для програм, написаних мовою C++. Для цього можна скористатися технологією OpenMP, під’єднавши бібліотеку omp.h і відзначивши у налаштуваннях Visual Studio відповідний прапорець, щоб компілятор зміг розпізнати прописані директиви.

У наявних програм найбільш трудомісткими місцями є цикли for, які проходять по всіх точках даних від початку до кінця масиву, виконуючи при цьому однакові операції. Для них можна застосувати директиву #pragma omp parallel for, яка створить паралельні потоки на процесорі, кожен із яких візьме на себе частину ітерацій і виконає їх незалежно та одночасно з іншими. Це дозволить прискорити процес обчислень, однак, потрібно мати на увазі, що застосування директив OpenMP створює додаткові витрати на створення, управління і синхронізацію процесів, тому потрібно грамотно підбирати їх написання.

У коді програми K-means такі цикли присутні у функціях clusterDistribution і relaculationCenters. У C-means такі ділянки коду є у функціях updateMembership та updateCenters. Для DBSCAN директиви можна додати у функціях findNeighbours і expandCluster. Залежно від вмісту циклу для директив вказуються параметри відповідно до синтаксису.

Після цього знову було проведено вимірювання часу за допомогою бібліотеки chrono. Нижче наведено результати вимірювань для K-means (таблиця 3.1), C-means (таблиця 3.2) та DBSCAN (таблиця 3.3).

Таблиця 3.1 – Результати вимірювання часу для алгоритму K-means мовою C++ з використанням технології OpenMP

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.0565	0.4527	4.2568

Таблиця 3.2 – Результати вимірювання часу для алгоритму C-means мовою C++ з використанням технології OpenMP

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.1845	0.8760	8.4661

Таблиця 3.3 – Результати вимірювання часу для алгоритму DBSCAN мовою C++ з використанням технології OpenMP

Розмірність масиву	(10000, 3)	(100000, 3)
Результати в секундах	3.2358	330.3530

Отримані значення показують, що застосування директив OpenMP дозволило отримати приріст швидкості всіх програм приблизно в 1.7-1.8 разів.

3.2 Обчислення на GPU з використанням технології CUDA

Тепер перейдемо до способу організації паралельних обчислень у програмах із використанням відеокарти NVIDIA та технології CUDA. Для цього можна підключитися до графічного прискорювача за допомогою хмарних сервісів. Подальші дослідження проведено у Google Colab.

На цьому етапі програми, написані мовою C++, були доповнені та адаптовані для використання відеокарти. Для цього деякі функції в алгоритмів були переписані як CUDA -ядра зі специфікатором _global_ - вони викликатимуться з хоста і запускатимуться на пристрої.

У K-means такими функціями стали: clusterDistrCUDA - для заповнення кластерів, recalculCentersCUDA - для перерахунку центрів, у C-means: updateMembershipCUDA - для перерахунку матриці ступенів приналежності, updateCentersCUDA - для перерахунку сусідів.

Деякі операції, призначені для однієї нитки (наприклад, обнулення лічильників) були винесені в окремі ядра. Також додано виділення пам'яті на пристрої, пересилання даних з подальшим запуском ядер та обчисленням, отримання результату та вивільнення пам'яті.

Після перевірки коректності роботи програм було організовано вимірювання часу з допомогою вбудованого механізму подій (CUDA event). Вимірювання проведені для:

- виділення пам'яті на пристрої;
- пересилання даних із хоста на пристрій;
- роботи ядер на пристрої;
- копіювання результату з пристрою на хост;
- звільнення пам'яті на пристрої.

Для фінального результату всі значення підсумовувалися. Вимірювання проводилися на 10-ти запусках із наступним усередненням.

Результати для K-means, C-means та DBSCAN (для обчислень на відеокарті в Google Colab) у секундах із заокругленням до 4-го знака дробової частини наведені нижче у таблицях 3.4, 3.5 та 3.6.

Таблиця 3.4 — Результати вимірювання часу для алгоритму K-means мовою C++ з використанням технології CUDA

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.002	0.0149	0.1380

Таблиця 3.5 — Результати вимірювання часу для алгоритму C -means мовою C++ з використанням технології CUDA

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.005	0.0342	0.2248

Таблиця 3.6 - Результати вимірювання часу для алгоритму DBSCAN на мові C++ з використанням технології CUDA

Розмірність масиву	(10000, 3)	(100000, 3)
Результати в секундах	0.7720	49.1274

Крім цього, у середовищі Google Colab були запущені програми для вимірювання часу вихідних програм на C++ для того, щоб отримати прискорення щодо інших одиниць вимірювання (таблиці 3.7, 3.8, 3.9).

Таблиця 3.7 - Результати вимірювання часу для алгоритму K-means

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.0376	0.3795	4.0712

Таблиця 3.8 — Результати вимірювання часу для алгоритму C -means

Розмірність масиву	(10000, 3)	(100000, 3)	(1000000, 3)
Результати в секундах	0.0842	0.9156	8,9386

Таблиця 3.9 — Результати вимірювання часу для алгоритму DBSCAN

Розмірність масиву	(10000, 3)	(100000, 3)
Результати в секундах	3.7035	376.977

3.3 Порівняльний аналіз продуктивності алгоритмів з різними реалізаціями

Тепер узагальнимо усі отримані результати. Нижче наведено підсумкові таблиці з результатами всіх вимірювань часу, проведених для алгоритмів K-means, C- means і DBSCAN (таблиці 3.10, 3.11, 3.12). Значення наведені в секундах.

Таблиця 3.10 - Підсумкова таблиця з усіма вимірюваннями часу для алгоритму K-means

Розмірність масиву	Scikit-learn K-means++	Python	C++	C++ Omp	C++ (Google Colab)	CUDA C++ (Google Colab)
(10000, 3)	0.0134	2.6457	0.0762	0.0565	0.0376	0.002
(100000, 3)	0.1105	29.0940	0.7869	0.4527	0.3795	0.0149
(1000000, 3)	0.454	284.945	7.8177	4.2568	4.0712	0.1380

Таблиця 3.11 - Підсумкова таблиця з усіма вимірюваннями часу для алгоритму C-means

Розмірність масиву	Python	C++	C++ Omp	C++ (Google Colab)	CUDA C++ (Google Colab)
(10000, 3)	3.7095	0.2034	0.1845	0.0842	0.005
(100000, 3)	37.2543	2.0930	0.8760	0.9156	0.0342
(1000000, 3)	372.5580	20.9263	8.4661	8,9386	0.2248

Таблиця 3.12 — Підсумкова таблиця з усіма вимірюваннями часу для алгоритму DBSCAN, значення наведені в секундах

Розмірність масиву	Scikit-learn DBDCAN	Python	C++	C++ Omp	C++ (Google Colab)	CUDA C++ (Google Colab)
(10000, 3)	0.4478	739.1143	6.0596	3.2358	3.7035	0.7720
(100000, 3)	-	72554.6706	569.1956	330.3530	376.977	49.1274

За отриманими даними розрахуємо приріст швидкості. Як одиниці беруться найповільніші результати - це час роботи для алгоритмів, написаних мовою Python. Щодо них розрахуємо прискорення та проведемо порівняльний аналіз продуктивності. У наведених нижче розрахунках пропорційно врахована різниця обчислень між ноутбуком та середовищем Google Colab. Можлива деяка похибка через округлення.

Результати розрахунків прискорення розглянутих алгоритмів наведені у таблицях 3.13 – 3.15.

Таблиця 3.13 - Розрахунок прискорення для алгоритму K-means

Розмірність масиву	Scikit-learn K-means++	Python	C++	C++ Omp	CUDA C++
(10000, 3)	197.4403	1	34.7204	46.8265	652,7435
(100000, 3)	263.2941	1	36.9729	64.2677	941,7013
(1000000, 3)	627.6339	1	36.4488	66.9390	1075,31749

Таблиця 3.14 - Розрахунок прискорення для алгоритму C-means

Розмірність масиву	Python	C++	C++ Omp	CUDA C++
(10000, 3)	1	18.2374	20,1057	307,11595
(100000, 3)	1	17.7995	42.5277	476.5330
(1000000, 3)	1	17.8033	44.0059	707.9093

Таблиця 3.15 - Розрахунок прискорення для алгоритму DBSCAN

Розмірність масиву	Scikit-learn DBSCAN	Python	C++	C++ Omp	CUDA C++
(10000, 3)	1650.5456	1	121.9741	228.4178	933.5989
(100000, 3)	-	1	127.4688	219.6277	978.1262

На основі табл. 3.13 – 3.15 побудовані графіки, що демонструють приріст швидкості програм з різними реалізаціями, для K-means (рисунок 3.1), C-means (рисунок 3.2) і DBSCAN (рисунок 3.3).

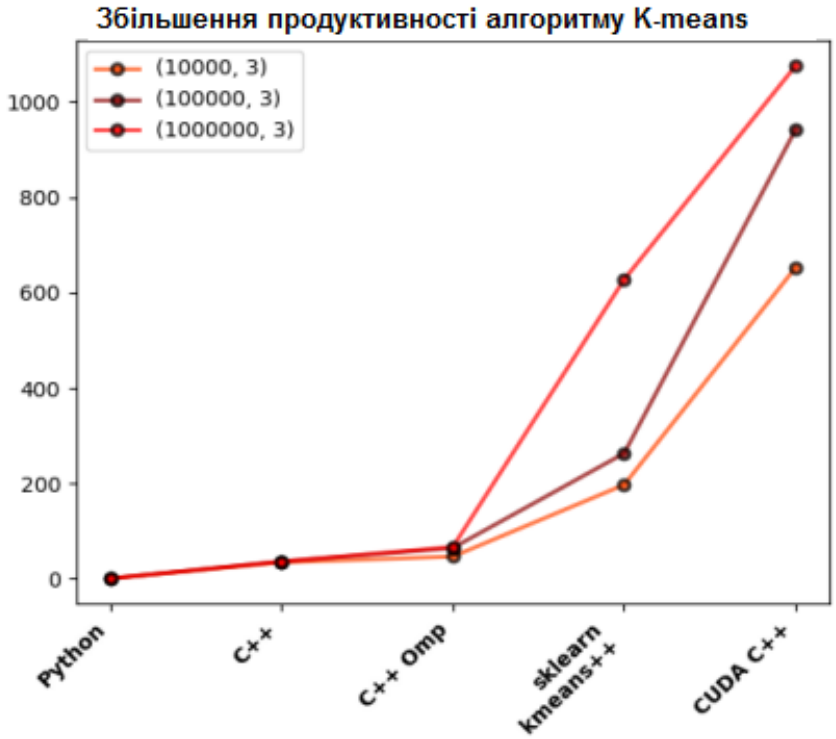

Рисунок 3.1 – Графік із прискоренням роботи алгоритму K-means

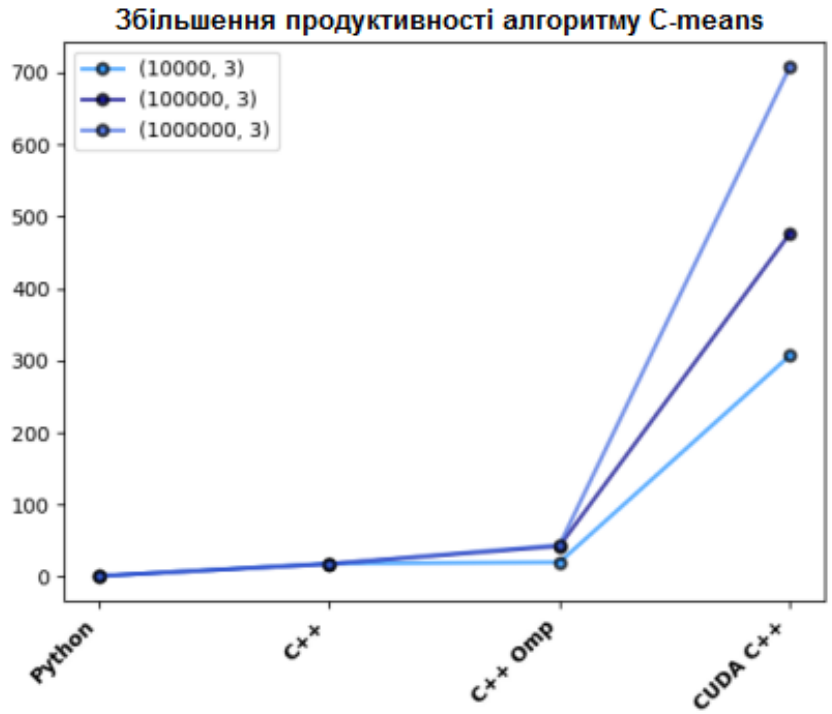

Рисунок 3.2 – Графік із прискоренням роботи алгоритму C-means

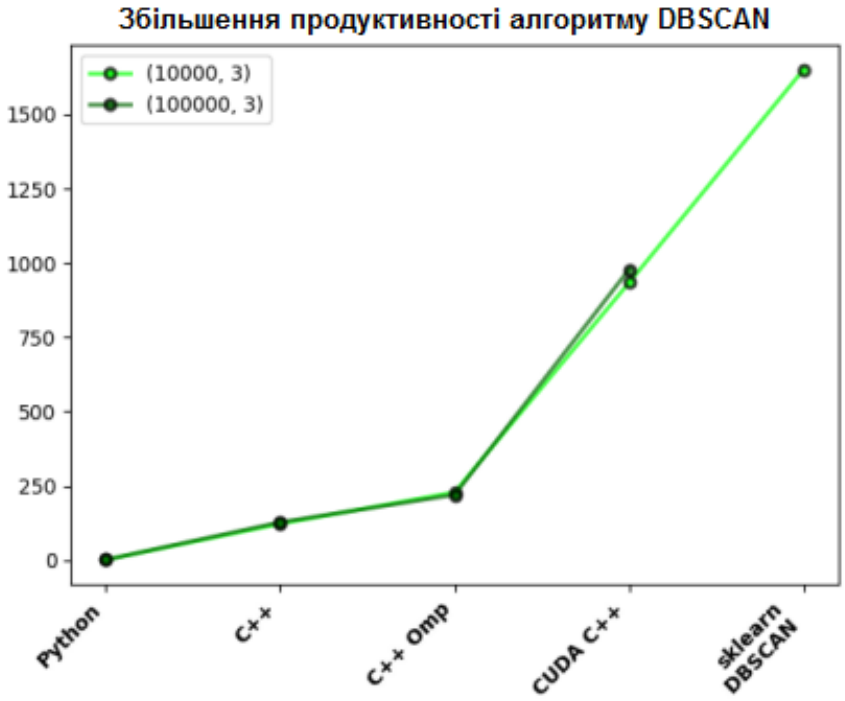

Рисунок 3.3 – Графік із прискоренням роботи алгоритму DBSCAN

Отримані значення та візуалізації показують, у скільки разів алгоритми різних реалізацій перевищують за швидкістю ті, що написані мовою Python. Найчастіше розумніше використовувати готові, вбудовані в бібліотеку Scikit-

learn. Однак, для роботи з великими даними власні програми можуть мати значну перевагу, оскільки є можливість більшого контролю над пам'яттю та організації процесу на власний розсуд. Можна бачити, що найкращі результати алгоритми K-means та C-means показали у варіанті з обчисленнями на відеокарті. Помітною є перевага використання мови C++. Завдяки тому, що вона є більш низькорівневою та компільованою, швидкість роботи програм на ній значно швидше. Особливо це помітно на алгоритмі DBSCAN, у якого послідовний код, написаний мовою C++, виявився більш ніж у 120 разів швидшим у роботі, ніж його реалізація з аналогічними обчисленнями Python.

Програми у розглянутих алгоритмів можна поліпшити, оскільки у них залишилися області, де можна організувати паралельні обчислення. Реалізації, написані на C++, підлягають подальшій оптимізації для прагнення більш вдосконалених версій.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ураженні електричним струмом

Удар електричним струмом є поширеною травмою і часто може закінчитися летальним випадком. Ураження електричним струмом відбувається підчас контакту тіла людини з джерелом електричної енергії під напругою. Це реакція організму людини на проходження електричного струму через тіло. Вона проявляється по різному, від легких потрясінь до небезпечних здоров'ю травм, які можуть вплинути на тканини в організмі.

Шкода завдана електричним струмом залежать від кількох факторів: наскільки висока була напруга, яка область тіла була вражена і від виду струму. Фізичні наслідки для людини можуть коливатися від опіків частин тіла до серйозних вражень внутрішніх органів [18].

Часто люди стикаються з підвищеним ризиком ураження високою напругою. Низька напруги не наносить серйозних травм для люди, а з іншої сторони висока напруги, яка має більше 500 В, може призвести до серйозного пошкодження тканин. У дітей або підлітків при враженні електричним струмом в діапазоні від 110 до 200 В можуть вникнути значні травми. Зазвичай це трапляється підчас порушення техніки безпеки при роботі з електричними приладами, які є в побуті. Це можуть бити електричні шнури, подовжувачі, несправні розетки та багато чого іншого.

Виділяють чотири основні фактори від яких залежить вплив електричного струму на організм: величина струму, що протікає через тіло; органи, через які проходить струм; час, протягом якого струм вражає тіло; частота струму.

Фізичні наслідки на пряму залежать від величини струму, яка вражає тіло людини. Струм менший за 1 мА не завдає жодної шкоди людина фізичного ефекту. При 1 мА можуть відчуватися слабкі поколювання, а при 5 мА – легкий поштовх. Однак при струмі ведичною від 6 до 25 мА людина може відчувати больовий шок і деяку втрату контролю над м'язами.

При ураженні електричним струмом від 50 до 150 мА може спричинити у людини сильний біль, м’язове скорочення і навіть призвести до зупинку дихання. У певних випадках можливий летальний результат для людини. Струм від 1000 до 4300 мА призводить до ймовірної смерть, тому що дана напруга спричиняє пошкодження нервових зав’язків, м’язових скорочення та порушення ритму серця. До 10 000 мА електричний струм спричиняє важкі опіки шкіри людини та зупинку серця. Висока ймовірність смерті.

Найпоширеніші ознаки та симптоми враження електричним струмом включають:

- втрата свідомості;
- ускладнення або зупинка дихання;
- опіки, які виникають там, де струм входить і виходить з тіла;
- зупинка серця;
- слабкий і непостійний пульс або його припинення.

Перш за все, що потрібно зробити при першій допомозі, це відключити джерело живлення. Вимкнути електропостачання, від’єднати електроприлад від джерела електричного струму або вимкнути блок запобіжників, якщо він знаходиться неподалік. Не потрібно намагатися підходити близько до жертви, якщо не переконані, що це безпечно і живлення вимкнено.

Потрібно бути обережним у вологих місцях, тому що вода є електричним провідником і рятівник може стати теж жертвою. Якщо людина не впевнена щодо вологості поверхні, потрібно відключити основне електропостачання будинку. У випадку коли це неможливо, використати підручний предмет, який не є провідником і відокремити людину від джерела струму. Це може бути дерев’яна або пластмасова річ.

Після того як постраждалого відокремили від джерела електрики, потрібно викликати швидку і надати першу медичну допомогу. Далі потрібно визначити стан жертви. Перевірити, чи людина у свідомості і дихає. У складних випадках у жертви може бути слабкий пульс або його відсутність. Можливо, що дихання зупинилося. Якщо людина втратила свідомість і перестала дихати, потрібно почати серцево-легеневу реанімацію. Руки розташовуємо в центрі грудної, одна

на іншу. Сильно і швидко натиснути 30 раз приблизно до третини діаметра грудної клітки. Після кожного натискання на грудну клітку робиться два рятувальні вдихи. Потрібно відкинути голову потерпілого назад і підняти підборіддя. Затиснути ніс і створити повне ущільнення. Далі подути потерпілому в рот і подивитися, чи підніметься грудна клітка. Потрібно продовжувати робити натискань на грудну клітку та вдих, поки не прибуде медична допомога або людина не почне сама дихати. Якщо потерпілий живий, перемістити його у зручне йому положення подальше від небезпеки. Можна запобігти шоку, поклавши людину рівно на землю, з головою трохи нижче тіла [18].

Якщо людина при свідомості, нормально дихає і на тілі є опіки, потрібно накрити їх звичайною харчовою плівкою або іншою неклейкою пов’язкою, але без мазі чи лосьйону. Якщо кровотеча у потерпілого, може знадобитися компресія та джгут.

При роботі з соціальною мережею користувач повинен знати і вміти як правильно поводитися з ПК, тому що людина перебуває у непосередньому контакті з джерелом напруги. Удар електричним струмом є потенційно смертельною травмою. Негайна медична допомога важлива, щоб запобігти серйозним травмам і смерті. Для запобігання уражень електричним струмом при роботі за ПК слід встановити додаткові захисні пристрої, що забезпечують недоступність токопровідних частин для дотику. Для зменшення небезпеки використовувати розділовий трансформатор для розв'язки з основною мережею.

Удар електричним струмом є потенційно смертельною травмою. Негайна медична допомога важлива, щоб запобігти серйозним травмам і смерті.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Робоче місце – це ділянка простору, яка облаштована необхідним обладнанням, відповідно до трудової діяльності, для виконання поставлених завдань.

Правильно побудоване робоче місце повинне забезпечувати:

– найкраще розміщення обладнання і предметів праці;

– не допускати дискомфорту;

– підвищувати продуктивність праці;

– зменшувати втому працівника.

Розмір робочого місця повинен бути таким, щоб людина не виконувала лишніх рухів і не відчувала дискомфорту під час виконання роботи. Також для працівника важливо мати змогу змінити робочу позу, наприклад, положення тулуба, рук або ніг. Потрібно мінімізувати або звести до нуля всі незручності положення тіла [19].

Різні дослідження заявляють, що при правильному проектуванні робочого місця продуктивність людини може зрости від 15-25%. Такі фактори як рівень освітлення, вологість повітря, температура, шум, вібрація, токсичність, мають значний вплив на умови життєдіяльності і працездатності людини.

Антропометричні вимоги визначають відповідність робочого місця до фізіологічних параметрів тіла людини як зріст і розміри тіла. Індикатором цього є правильна робоча поза, відсутність дискомфорту, оптимальні зони досягнення, раціональні рухи. Психофізіологічні та фізіологічні вимоги формулюють відповідність обладнання і робочого місця можливостям співробітника щодо розуміння, обробки даних, пошук і реалізації рішень.

Організація робочого місця передбачає наступні пункти:

– раціональне положення робочого місця у приміщенні;

– вибір робочих меблів відповідно до фізіологічних характеристик працівника;

– правильне компонування і розміщення обладнання на робочих місцях;

– урахування особливостей та характеру професійної діяльності.

До загальних принципів організації робочого місця відносять:

– робоче місце повинне містити тільки ті предмети, які беруть учать у робочому процесі, але не заважати йому;

– предмети, які часто використовуються у роботі, розміщуються ближче, ніж ті речі, якими користуються рідше;

– предмети, які беруться лівою рукою, повинні розміщуватися зліва, а предмети, які використовуються правою рукою — справа;

– якщо при роботі з предметом працівник використовує дві руки, то він розміщується з урахуванням зручності захоплювання його двома руками;

– робоче місце не повинно бути засмічене;

– необхідна оглядовість повинна бути забезпечена при правильній організації робочого місця.

Робоча поза – це найбільш тривале положення тіла працівника протягом робочого дня. При зручній робочій позі забезпечується стійкість положення тулуба, ніг, рук, голови і витрачається мінімальний запас енергії та максимальну продуктивність [19].

Сидячи і стоячи – дві найпопулярніших пози у робочому процесі. При проектуванні робочого місця потрібно враховувати, що з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи. При роботі стоячи, людина стомлюється більше ніж сидячи. У відсотковому еквіваленті це на 10% більше енергії. При додатковому навантаженні підвищується артеріальний і венозний тиск крові, розширення вен, пошкоджуються ступені та викривляється хребет. У свою чергу при сидячій роботі нижня частина тіла розслаблена, а основне навантаження спрямоване на м'язи шиї, спини, таза, стегон. При неправильній сидячій позі розвивається застій крові у ногах, а якщо пальці виконують багато роботи можливе запалення суглобів.

Організація робочого місця при використанні ПК повинна відповідати усім ергономічним вимогам. Ключові ергономічні вимоги до проектування робочого місця оператора ПК зображені на рисунку 4.1.

При роботі з персональним комп’ютером потрібно:

– зменшувати кількість статичних напружень;

– розподіляти кількість і час статичних напружень;

– змінювати робочі пози під професійної діяльності.

Саме вибір правильної робочої пози визначається від впливу багатьох факторів. Одні з найважливіших це – кількість зусиль яка прикладається, велична робочої зони, відношення висоти робочої поверхні і ростом працівника.

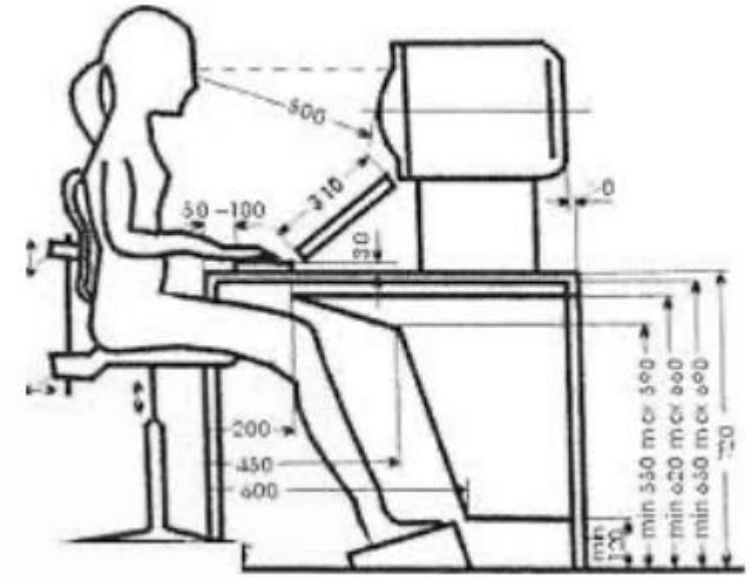

Рисунок 4.1 – Робочий стіл і розміщення користувача ПК

При використанні соціальної мережі користувач повинен дотримуватися вище зазначених правил, які будуть сприяти комфортній і продуктивній роботі. Важливо регулярно робити короткі перерви. Часта зміна заняття – кращий спосіб уникнути можливих неприємностей.

ВИСНОВКИ

У роботі розглянуто основні поняття, пов'язані з кластеризацією даних та паралельними обчисленнями. Досліджено та програмно реалізовано три алгоритми кластеризації: K-means, Fuzzy C-means та DBSCAN, проведено оцінку точності. Проведено чисельні експерименти роботи програм із різними реалізаціями.

Зазначено, що вибір мови програмування впливає продуктивність. Усі реалізації на C++ значно перевищують за швидкістю роботу програм мовою Python, що має важливу роль прикладних завданнях. Можливість контролю над пам'яттю та застосування паралельних обчислень сприяє оптимізації процесу та скорочення витрати ресурсів.

Результати показали, що застосування технологій організації паралельних обчислень сприяє значному прискоренню роботи програм. Таким чином, завдяки OpenMP вдалося майже вдвічі прискорити роботу алгоритмів. А використання графічного процесора за допомогою технології CUDA дозволило суттєво перевершити усі розглянуті варіанти реалізацій.

Варто відмітити і те, що написання власних програм мовою C++, які здійснюють роботу алгоритмів машинного навчання, можуть мати перевагу над використанням готових бібліотек.

ПЕРЕЛІК ДЖЕРЕЛ

1. Everitt, B. S., Landau, S., Leese, M., & Stahl, D. Cluster Analysis. Wiley, 2011. 346 p.

2. Gan, G., Ma, C., & Wu, J. Data Clustering: Theory, Algorithms, and Applications. SIAM, 2007. 466 p..

3. Болюбаш Н. М. Інтелектуальний аналіз даних : навч. посіб. / Н. М. Болюбаш. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. 320 с.

4. Han, J., Kamber, M., & Pei, J. Data Mining: Concepts and Techniques. - Morgan Kaufmann, 2011. 744 p.

5. Jain, A., Murty, M., Flynn, P. "Data Clustering: A Review." // ACM Computing Surveys, 1999, Vol. 31, No. 3, pp. 264-323.

6. Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of Internal Clustering Validation Measures. In Proceedings of the 2010 IEEE International Conference on Data Mining (pp. 911-916).

7. Rohit Chandra et al. Parallel Programming in OpenMP. Morgan Kaufmann Publisher. 2001. 230 p.

8. Технологія паралельного програмування. Навчальний посібник: для студентів галузі знань «12 Інформаційні технології» спеціальності 123 «Комп’ютерна інженерія» спеціалізації 123.02 «Системне програмування» Муляревич О.В. Львів:/ Видавництво "Магнолія 2006", 2024. 213 с..

9. CUDA Toolkit Documentation. [Електронний ресурс] – Режим доступу: https://docs.nvidia.com/cuda/ (Дата звернення 16.05.2026).

10. scikit-learn: Machine Learning in Python, "Clustering." [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/modules/clustering.html (Дата звернення 17.05.2026).

11. Bezdek, James C. "Pattern Recognition with Fuzzy Objective Function Algorithms." Boston, MA: Springer US, 1981. 272 p.

12. Ester, M., Kriegel, H.-P., Sander, J., Xu, X. A Density-Based Algorithm для Discovering Clusters в Великі простірні Databases with Noise // 2D International Conference on Knowledge Discovery and Data Mining (KDD-96). Portland, OR,

USA: AAAI Press, 1996. P. 226-231.

13. Методичні вказівки до виконання кваліфікаційної роботи ор Бакалавр для студентів спеціальності 122 – Комп’ютерні науки, всіх форм навчання / укладачі: Готович В.А., Дуда О.М. Никитюк В.В. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 43 с.

14. Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021). Mathematical model of gas consumption process in the form of cyclic random process. 2021 IEEE 16th International Conference on Computer Sciences and Information.

15. Lytvynenko I. V. Method of segmentation of determined cyclic signals for the problems related to their processing and modeling. Scientific journal of the Ternopil National Technical University. No. 4 (88). 2017. ISSN: 2522-4433. P. 153–169. https://doi.org/10.33108/visnyk_tntu2017.04.153

16. Lupenko, S. A., Lytvynenko, I. V., Sverstiuk, A., Shelestovskyi, B., & Horkunenko, A. (2021). Software for Statistical Processing and Modeling of a Set of Synchronously Registered Cardio Signals of Different Physical Nature. CMIS, 194-205.

17. Bodnarchuk, I., Skorenkyy, Y., Kramar, T., Duda, O., & Nykytyuk, V. (2022). Use of Analytical Hierarchy Process in Scenarios Design for a Digital Museum with XR components. ITTAP, 414–425

18. Заікіна Д., Глива В. Основи охорони праці та безпека життєдіяльності. 2019. URL: https://doi.org/10.31435/rsglobal/001 (дата звернення: 19.05.2025)

19. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.