Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ ОНЛАЙН ДОСТОВІРНОЇ КЛАСТЕРИЗАЦІЇ ДАНИХ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-2
Захаров Є.М.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Захарову Єгору Максимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів онлайн достовірної кластеризації даних

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії _22_ _грудня_____2023 р.

3. Вихідні дані до роботи алгоритм кластеризації даних на основі методу *k*-means.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд та аналіз методів онлайн достовірної кластеризації для визначення переваг та недоліків.

2. Розробка та оновлення алгоритму онлайн достовірної кластеризації.

3. Проведення експеременту з реальними або симульованими даними.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми браної теми, постановка задачі, обран метод для кластеризації, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	4.11.23-6.11.23	
3	Аналіз літератури з досліджуваної проблеми	7.11.23-14.11.23	
4	Аналіз технічних засобів	15.11.23-17.11.23	
5	Розробка методу	18.11.23-22.11.23	
6	Програмна реалізація	22.11.23-28.11.23	
7	Оформлення пояснювальної записки	28.11.23-3.12.23	
8	Перевірка на плагіат	6.12.2023	
9	Рецензування	10.12.2023	
10	Підготовка презентації та доповіді	24.12.2023	
11	Занесення роботи в електронний архів	01.01.2024	
12	Попередній захист кваліфікаційної роботи	03.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Тітова О.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 61 с., 2 табл., 6 рис., 40 джерел.

АЛГОРИТМ *K*-MEANS ОНЛАЙН, DBSCAN ОНЛАЙН, АЛГОРИТМ MINI-BATCH *K*-MEANS, СПЕКТРАЛЬНА КЛАСТЕРИЗАЦІЯ ОНЛАЙН, АЛГОРИТМИ ІЄРАРХІЧНОЇ КЛАСТЕРИЗАЦІЇ ОНЛАЙН.

Об’єктом дослідження є методи онлайн достовірної кластеризації даних.

Метою дослідження є аналіз та вдосконалення існуючих методів онлайн достовірної кластеризації даних з метою розробки нових підходів для аналізу потокових даних в реальному часі.

Проведено огляд та аналіз існуючих методів онлайн достовірної кластеризації даних для визначення їхніх переваг та недоліків. Проведено експерименти з реальними або симульованими даними для оцінки якості та продуктивності розроблених алгоритмів порівняно з існуючими методами.

У результаті дослідження розроблено та покращено алгоритми онлайн достовірної кластеризації, зокрема для роботи з нестаціонарними даними та великими обсягами інформації.

K-MEANS ONLINE ALGORITHM, DBSCAN ONLINE, MINI-BATCH *K*-MEANS ALGORITHM, SPECTRAL CLUSTERING ONLINE, HIERARCHICAL CLUSTERING ONLINE ALGORITHMS.

The object of the research is methods of online credible data clustering.

The research aims to analyze and enhance existing methods of online credible data clustering to develop new approaches for real-time analysis of streaming data.

A review and analysis of existing methods of online credible data clustering were conducted to identify their strengths and weaknesses. Experiments were carried out with real or simulated data to evaluate the quality and performance of the developed algorithms compared to existing methods.

As a result of the research, algorithms for online credible data clustering have been developed and improved, particularly for handling non-stationary data and large volumes of information.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RI – Rand Index (індекс Ренда)

SS – Silhouette Score (оцінка силуету)

ARI – Adjusted Rand Index (скоригований індекс Ренда)

NMI – Normalized Mutual Information (нормалізована взаємна інформація)

SOM – Self Organizing Maps (самоорганізуючіся карти)

DBSCAN – Density-Based Spatial Clustering of Applications with Noise (густинно-засноване просторове кластеризування застосунків з шумом)

GCN – Graph Convolutional Networks (мережі згорткових графів)

ВСТУП

Все більше відомостей, даних і інформації нині зберігається в цифровому форматі, що створює великі можливості для використання аналітики та обробки даних. Однією з ключових завдань в цьому контексті є групування схожих об’єктів чи споживачів в логічні категорії для подальшого аналізу. Це саме те, чим займається кластеризація даних.

Дослідження методів онлайн достовірної кластеризації даних – це актуальна та важлива тема в галузі науки про дані. Ця тема вивчає методи, які дозволяють ефективно та автоматизовано групувати дані, щоб виділити приховані закономірності, зробити передбачення та вивести корисний інсайт. Зростання обсягів даних у таких сферах, як медицина, фінанси, маркетинг, технологічна розробка та багато інших, підкреслює важливістьрозробки надійних та швидких методів кластеризації. Наприклад, в медициніце може допомогти в класифікації пацієнтів за схожістю симптомів дляточніших діагнозів, а в маркетингу – впровадженні більш ефективних стратегій споживчої поведінки.

Дослідження методів онлайн достовірної кластеризації даних має потенціал революціонізувати багато сфер діяльності, роблячи аналіз і використання даних більш точними та продуктивними. Ця тема заслуговує на увагу дослідників і фахівців, що працюють у сфері аналітики даних та науці про дані.

В сучасному цифровому світі, де кількість і різноманітність даних швидко зростають, розробка і вдосконалення методів кластеризації є необхідною для вилучення цінної інформації з великих обсягів непорядкованих даних. Особливу увагу слід звертати на онлайн достовірну кластеризацію, оскільки вона враховує динаміку змін в часі, що є критичним аспектом в реальних умовах функціонування багатьох галузей.

1 АНАЛІЗ ТА ОГЛЯД МЕТОДІВ ОНЛАЙН ДОСТОВІРНОЇ КЛАСТЕРИЗАЦІЇ ДАНИХ

1.1 Аналіз дослідження онлайн достовірної кластеризації даних

Сучасний вік великих обсягів даних супроводжується постійним зростанням необхідності в ефективних методах обробки та аналізу інформації. Одним з важливих інструментів для виявлення прихованих закономірностей в даних є метод кластеризації. Кластеризація дозволяє групувати схожі об'єкти для виявлення структури та залежностей в наборах даних.

Однак з поширенням онлайн-платформ та постійним потоком даних в реальному часі виникає потреба в адаптивних та швидких алгоритмах кластеризації [1]. Аналіз онлайн достовірної кластеризації даних відкриває нові перспективи для розвитку ефективних стратегій обробки інформації в реальному часі та виявлення динамічних структур у надходящих потоках даних.

У цьому контексті дослідження спрямоване на аналіз сучасних підходів та технологій у галузі онлайн кластеризації даних, розкриття їхніх переваг і недоліків, а також вивчення можливостей їхнього використання для вирішення завдань реального світу. Це дослідження має на меті визначити тенденції розвитку та перспективи застосування онлайн кластеризації в умовах постійної зміни та швидкості потоків даних.

1.1.1 Актуальність

Актуальність онлайн достовірної кластеризації даних в сучасному світі обумовлена рядом факторів і важливих викликів, з якими стикаються підприємства та дослідники у галузі обробки даних. Розглянемо деякі з них: потокові дані в реальному часі, автоматизація та прийняття рішень, змінність

структури даних, забезпечення безпеки та виявлення аномалій, масштабованість.

Потокові дані в реальному часі: з великим обсягом даних, що надходять в реальному часі з різних джерел, включаючи датчики IoT, соціальні мережі та інші, необхідно мати здатність аналізувати та кластеризувати ці дані без затримки [2]. Онлайн кластеризація дозволяє реагувати на зміни в реальному часі.

Автоматизація та прийняття рішень: в багатьох галузях, таких як фінанси, медицина та маркетинг, важливо мати засоби для автоматичного аналізу даних та прийняття рішень. Онлайн достовірна кластеризація даних допомагає створювати моделі та робити прогнози в режимі реального часу.

Змінність структури даних: у багатьох сценаріях структура даних може змінюватися з часом, і стандартні алгоритми кластеризації можуть бути неефективними. Онлайн методи адаптуватися до цієї змінності та забезпечують достовірну кластеризацію [3].

Забезпечення безпеки та виявлення аномалій: в сферах як фінанси та кібербезпека важливо виявляти аномалії та несанкціонований доступ в режимі реального часу. Онлайн кластеризація може бути використана для виявлення незвичайних зразків та аномалій в потокових даних.

Масштабованість: онлайн методи кластеризації можуть бути масштабовані для обробки великих обсягів даних. Це важливо в умовах сучасного великого обсягу інформації.

З огляду на ці виклики і можливості, дослідження методів онлайн достовірної кластеризації даних є дуже актуальною і важливою областю в галузі обробки даних та машинного навчання. Досягнення прогресу в цій області може мати великий вплив на багато галузей, покращуючи можливості аналізу та використання потокових даних.

1.1.2 Застосування в різних галузях

Методи онлайн достовірної кластеризації даних знайшли застосування в різних галузях завдяки їхній здатності аналізувати та групувати дані в режимі реального часу. Нижче наведено приклади застосування онлайн достовірної кластеризації даних в різних галузях: маркетинг і реклама, фінанси та кредитування, медицина, телекомунікації, кібербезпека, логістика і транспорт, соціальні мережі [4].

В сфері маркетингу та реклами онлайн кластеризація даних допоможе сегментувати аудиторію та виявити споживчі тренди. Онлайн кластеризація даних допомагає маркетологам групувати клієнтів за схожими характеристиками, щоб створювати персоналізовані рекламні кампанії. аналіз потокових даних дозволяє виявляти нові тренди та адаптивні смаки споживачів для покращення стратегій маркетингу [5].

В фінансовій сфері онлайн кластеризація даних допоможе краще виявляти фінансові аномалії та поліпшить проведення кредитного аналізу. Онлайн кластеризація може виявляти аномальні фінансові транзакції, які можуть бути ознаками шахраїв або шахрайських схем. Кредитний аналіз допоможе групувати клієнтів за рівнем кредитної спроможності допомагає приймати рішення про надання кредитів [6].

В сфері медицини онлайн кластеризація даних допоможе класифікувати пацієнтів та допоможе в моніторингу стану пацієнтів. Онлайн кластеризація даних в медицині допомагає класифікувати пацієнтів за симптомами та діагнозами для покращення діагностики та лікування. Аналіз потокових даних може виявляти патерни в зміні стану пацієнтів та попереджати про можливі ускладнення [7].

В сфері телекомунікації онлайн кластеризація даних допоможе в управлінні мережевим трафіком та аналізі послуг споживачів. Онлайн кластеризація даних може виявляти аномалії та навантаження в мережах для оптимізації трафіку та забезпечення якості обслуговування.

Дозволяє провайдерам аналізувати споживачів послуг для покращення пропозицій та обслуговування.

В нащі часи кібербезпека є важливою слакодовою нашого життя. Онлайн кластеризація даних допоможе виявляти незвичайну поведінку та потенційні загрози в мережі для запобігання кібератак [8].

В сфері логістики онлайн кластеризація даних допоможе оптимізувати маршрути та постачання. Кластеризація даних допомагає оптимізувати постачання та визначати найкращі маршрути для транспортних засобів.

У соціальних мережах онлайн кластеризація допомагає аналізувати дані користувачів і надавати персоналізовані рекомендації, наприклад, для онлайн-магазинів та платформ медіа [9].

Ці приклади підтверджують широкий спектр застосувань онлайн достовірної кластеризації даних у різних галузях, де вона допомагає вдосконалювати аналіз даних та приймати більш обґрунтовані рішення.

1.1.3 Виклики та перспективи

Онлайн достовірна кластеризація даних ставить перед собою виклики: нестаціонарність даних, великий обсяг даних, мінімізація обчислювальних витрат, метрики якості оцінки. Розглянемо більш детально ці виклики.

Однією з основних проблем є нестаціонарність даних, коли структура даних змінюється з часом. Онлайн кластеризація повинна бути здатна адаптуватися до цих змін та виявляти нові кластери [10].

В умовах великого обсягу потокових даних важливо розробити ефективні алгоритми, які можуть обробляти великі масиви інформації в реальному часі.

Онлайн кластеризація повинна бути обчислювально ефективною, щоб забезпечити швидку реакцію на потокові дані і мінімізувати витрати на обчислення.

Розробка відповідних метрик для оцінки якості кластеризації в режимі онлайн є важливою, оскільки стандартні метрики можуть бути неадекватними у цьому контексті.

Також розглянемо перспективи: адаптація до змін, використання глибокого навчання, масштабованість, застосування в галузях штучного інтелекту, створення інтерактивних інтерфейсів, використання в практиці. Розглянемо більш детально перспективи.

Подальший розвиток алгоритмів, які здатні автоматично адаптуватися до змін структури даних, дозволить зберігати високу якість кластеризації в умовах змін.

Застосування глибокого навчання для кластеризації даних може покращити якість результатів та здатність виявляти складні закономірності в даних.

Подальший розвиток масштабованих та розподілених систем дозволить обробляти великі обсяги даних в реальному часі.

Розширення використання онлайн достовірної кластеризації в галузях, таких як автономні автомобілі, робототехніка та медичні дослідження, відкриває нові можливості для застосування цих методів.

Розробка інтерактивних інтерфейсів для взаємодії з результатами кластеризації даних допоможе кінцевим користувачам легше розуміти та використовувати результати.

Подальший розвиток інструментів та платформ для реалізації онлайн достовірної кластеризації даних дозволить підприємствам і дослідникам впроваджувати ці методи в різних галузях.

Зростання потреб та викликів у галузі онлайн достовірної кластеризації даних свідчать про важливість подальшого розвитку цієї області та пошук нових інноваційних рішень для аналізу та групування потокових даних в реальному часі.

1.2 Огляд методів онлайн достовірної кластеризації даних

Перелік методів онлайн достовірної кластеризації даних включає різноманітні підходи та алгоритми, призначені для аналізу потокових даних в реальному часі. Ось деякі з них: *K*-Means онлайн, DBSCAN онлайн, Mini-Batch *K*-Means, спектральна кластеризація онлайн, алгоритми ієрархічної кластеризації онлайн, онлайн методи навчання без учителя, методи потікого-графового аналізу, методи нейромереж.

K-Means онлайн алгоритм є модифікацією класичного методу *k*-Means для роботи з потоковими даними. Він автоматично оновлює кластери при надходженні нових даних та може адаптувати кількість кластерів за потреби.

Метод DBSCAN адаптований для режиму онлайн та дозволяє виявляти кластери на основі густини даних. Він адаптує границі кластерів, якщо дані змінюються.

Mini-Batch *K*-Means алгоритм використовується для кластеризації великих обсягів даних в режимі реального часу. Він розділяє дані на міні-пакети та оновлює кластери із використанням цих пакетів.

Спектральна кластеризація використовує схожість між об’єктами для побудови графа схожості та подальшої кластеризації. Методи спектральної кластеризації можуть бути адаптовані для режиму онлайн [11].

Деякі методи ієрархічної кластеризації можуть бути адаптовані для роботи в режимі онлайн. Вони будують ієрархію кластерів та можуть реагувати на зміни в структурі даних.

Онлайн методи навчання без учителя включають алгоритми, такі як онлайн-навчання головних компонентів і онлайн-навчання незалежних компонентів, які використовуються для аналізу та кластеризації даних в режимі реального часу [12].

Методи потікого-графового аналізу використовують графовий підхід для аналізу потокових даних і виявлення кластерів на основі графової структури.

Застосування нейромереж, такі як SOM та автоенкодери, для онлайн кластеризації та аналізу даних.

Ці методи представляють різні підходи до онлайн достовірної кластеризації даних та можуть бути використані в залежності від конкретних завдань і характеристик даних.

1.3 Постановка задачі дослідження

Таким чином, розробка та покращення алгоритмів онлайн достовірної кластеризації даних є актуальним завданням. Тому ставиться завдання оцінити та проаналізувати існуючі методи онлайн достовірної кластеризації даних з метою вдосконалення їх характеристик та розробки нових підходів для аналізу потокових даних в реальному часі.

Об’єктом дослідження є методи онлайн достовірної кластеризації даних.

Метою дослідження є аналіз та вдосконалення існуючих методів онлайн достовірної кластеризації даних з метою розробки нових підходів для аналізу потокових даних в реальному часі.

Для досягнення мети необхідно вирішити такі завдання:

– провести огляд та аналіз існуючих методів онлайн достовірної кластеризації даних для визначення їхніх переваг та недоліків;

– розробити та покращити алгоритми онлайн достовірної кластеризації, зокрема для роботи з нестаціонарними даними та великими обсягами інформації;

– провести експерименти з реальними або симульованими даними для оцінки якості та продуктивності розроблених алгоритмів порівняно з існуючими методами;

– сформулювати висновки щодо ефективності та придатності розроблених методів.

2 МЕТОДИ ОНЛАЙН ДОСТОВІРНОЇ КЛАСТЕРИЗАЦІЇ ДАНИХ

2.1 Метод *K*-Means та Mini-Batch *K*-Means онлайн

Розглянемо метод *K*-Means онлайн. Метод *K*-Means онлайн є модифікацією класичного алгоритму *K*-Means для роботи з потоковими даними, які поступають в реальному часі. Основна ідея полягає в тому, щоб адаптувати кластери під нові дані без перерахунку всіх точок даних. Розглянемо основні етапи цього методу: ініціалізація, визначення кластерів, оновлення центроїдів, адаптація до нових даних, видалення застарілих даних, ітерації [13].

Для ініціалізації оберемо початкові центроїди для кожного кластеру. Нехай C_i – центроїд клстеру i, а $X_j -$ точка даних j.

$$C_i = X_j, \text{ для деякого } j. \tag{2.1}$$

Наступним кроком є визначення кластерів. Кожна точка X_j призначається до кластеру, який має найближчий центроїд. Функція відстані d може бути евклідовою відстанню:

$$min_i d(X_j, C_i). \tag{2.2}$$

При оновленні центроїдів розпочинається перерахунок центроїдів кластерів, використовуючи середнє значення точок, що належать кожному кластеру. Нехай N_i – кількість точок у кластері i.

$$C_j = \frac{1}{N_j} \sum_{j \in Cluster\ i} X_j. \tag{2.3}$$

Після чого проводиться адаптація до нових даних. Якщо нова точка X_{new} потрапляє в потік даних, алгоритм оновлює відповідний кластер та його центроїд:

$$C_j = \frac{1}{N_j+1}(\sum_{j\in Cluster\ i} X_j + X_{new}). \quad (2.4)$$

Наступним кроком є видалення застарілих даних. Якщо деякі дані видаляються або втрачають актуальність, може здійснюватися адаптація кількості кластерів. Кроки 2–4 повторюються для нових даних, які надходять в потік [14].

Цей алгоритм дозволяє ефективно кластеризувати потокові дані, адаптуючись до їхньої динаміки та забезпечуючи можливість роботи в режимі реального часу (рис. 2.1).

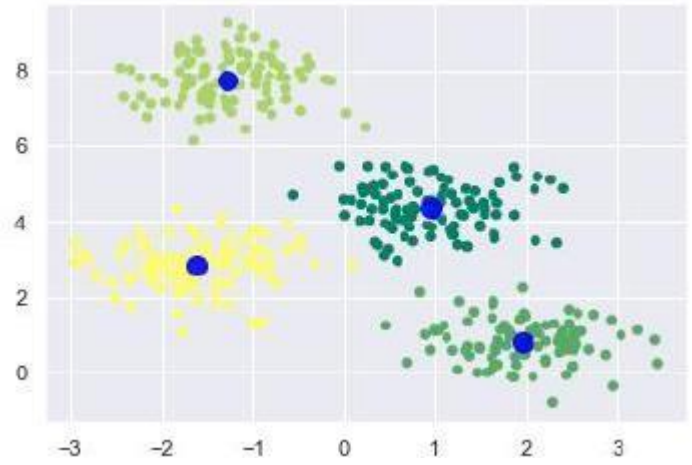

Рисунок 2.1 – Приклад роботи алгоритма *K*-Means онлайн

Метод онлайн достовірної кластеризації даних *K*-Means має свої переваги та недоліки. Розглянемо його переваги.

K-Means онлайн швидкий та ефективний для обробки великих обсягів потокових даних. Метод має здатність адаптуватися до змін в структурі даних та динамічно змінювати кількість кластерів, а також забезпечує можливість проводити кластеризацію в режимі реального часу.

Він використовується для великих обсягів даних, оскільки не вимагає обробки всього набору даних, а також не потребує попередньої інформації щодо кількості кластерів, оскільки може динамічно змінювати їх кількість.

Серед недоліків можна зазначити чутливість до початкового вибору та до шуму. Результат може залежати від початкового вибору центроїдів, що може призводити до різних кластеризацій. Велика кількість шумових або викидних даних може вплинути на якість кластеризації. Також метод не завжди ефективно визначає розмір кластера, і вони можуть бути нерівномірними за розміром. При динамічних змінах в структурі даних або збільшенні кількості кластерів може збільшуватися обчислювальна складність. Також може статися, що при видаленні застарілих даних може бути втрачена важлива інформація для певних кластерів [15].

Алгоритм *K*-Means має варіацію алгоритму Mini-Batch *K*-Means. Варіація використовує випадковий підмасив (mini-batch) даних для оновлення центрів кластерів, замість використання повного набору даних. Це робить алгоритм менш обчислювально вимогливим та дозволяє ефективно обробляти великі обсяги даних. Основними ідеями Mini-Batch *K*-Means є: випадковий вибір підмасиву, оновлення центроїдів, швидкість збіжності та параметри mini-batch. Розглянемо біль детально.

Замість використання усього набору даних для кожної ітерації, Mini-Batch *K*-Means випадковим чином вибирає підмасив фіксованого розміру. Це дозволяє алгоритму більш ефективно працювати з великими об’ємами даних. Після вибору підмасиву алгоритм використовує його для оновлення центрів кластерів, але робить це тільки для підмножини даних, а не для всієї вибірки. Це може зменшити обчислювальні витрати. Використання mini-batch дозволяє алгоритму збігатися швидше за рахунок випадкового вибору підмасиву та можливості оновлення центрів частково. Кількість ітерацій та розмір mini-batch є параметрами, які можна налаштувати. Вони впливають на швидкість та якість кластеризації. Хоча Mini-Batch *K*-Means дозволяє працювати з великими обсягами даних, важливо враховувати, що результати

можуть варіюватися у порівнянні з традиційним *K*-Means через випадковість вибору підмасиву.

Метод Mini-Batch *K*-Means має свої переваги та недоліки. Основними перевагами алгоритму є: ефективність для великих обсягів даних, зменшення обчислювальних витрат, підходить для онлайн-кластеризації, легко паралелізується.

Основною перевагою Mini-Batch *K*-Means є його ефективність приробоні з великими обсягами даних. Використання випадкових підмасивів дозволяє алгоритму швидко збігатися, не обробляючи всю вибірку даних. Оскільки алгоритм використовує лише частину даних для оновлення центрів,він може значно зменшити обчислювальні витрати порівняно з традиційним *K*-Means, що корисно для великих наборів даних. Mini-Batch *K*-Means може використовуватися для онлайн-кластеризації, де нові дані поступово додаються, а алгоритм адаптується до змін. Завдяки обробці лише підмасивів, Mini-Batch *K*-Means легше паралелізується, що дозволяє використовувати його на обчислювально потужних системах [16].

Основними недоліками алгоритму Mini-Batch *K*-Means є: нестійкість результатів, залежність від параметрів, низька точність при невеликому mini-batch, алгоритм не завжди оптимальний для низькорозмірних даних. Через випадковий вибір підмасивів результати Mini-Batch *K*-Means можуть варіюватися між різними запусками, що робить його менш стійким порівняно з традиційним *K*-Means. Для ефективної роботи потрібно правильно вибрати параметри, такі як розмір mini-batch та кількість ітерацій. Погані параметри можуть призвести до неякісної кластеризації. Якщо розмір mini-batch занадто малий, точність може зменшитися, оскільки алгоритм буде погано враховувати варіації в даних. Для невеликих наборів даних або коли кількість кластерів невелика, традиційний *K*-Means може бути більш оптимальним. Істотно, які переваги і недоліки будуть більш критичними, залежить від конкретних вимог і характеристик даних (табл. 2.1).

Таблиця 2.1 – Порівняння методів *K*-Means та Mini-Batch *K*-Means

Характеристика	***K*-Means**	**Mini-Batch *K*-Means**
Основний принцип	Використовує весь набір даних	Використовує випадкові підмасиви для оновлення
Точність	Зазвичай точний	Може призвести до менш точних результатів
Застосування	Невеликі та середні дані	Ефективний для великих обсягів даних
Потужність обчислень	Вимагає багато ресурсів	Зменшує обчислювальні витрати
Розмір даних	Добре підходить для маленьких та середніх обсягів	Ефективний для великих обсягів даних
Стійкість до кількості кластерів	Менш стійкий при великій кількості кластерів	Може бути більш ефективним при великій кількості кластерів через випадковий вибір підмасивів

Зробивши порівняння між звичайним методом *K*-Means і методом Mini-Batch *K*-Means, можна зробити наступні висновки. *K*-Means використовує весь набір даних для оновлення центроїдів кластерів.

Mini-Batch *K*-Means використовує випадкові підмасиви для оновлення центроїдів, що дозволяє працювати з частинами даних. *K*-Means зазвичай дає точні результати, оскільки використовує всі дані. Mini-Batch *K*-Means може призводити до менш точних результатів, але це компенсується зменшеним обчислювальнимивитратами.

K-Means підходить для невеликих і середніх обсягів даних. Mini-Batch *K*-Means ефективний для великих обсягів даних, оскільки працює з підмасивами. *K*-Means вимагає значних обчислювальних ресурсів. Mini-Batch *K*-Means зменшує обчислювальні витрати через використання випадкових підмасивів. *K*-Means добре підходить для маленьких і середніх обсягів даних.

Mini-Batch *K*-Means ефективний для великих обсягів даних, що дозволяє обробляти їх частинами. *K*-Means менш стійкий при великій кількості кластерів. Mini-Batch *K*-Means може бути більш ефективним при великій кількості кластерів завдяки випадковому вибору підмасивів. Загалом, *K*-Means може бути кращим вибором для завдань, де точність має велике значення, а Mini-Batch *K*-Means може бути ефективним для швидкої кластеризації великих обсягів даних при прийнятному рівні точності.

2.2 Метод DBSCAN онлайн

Це алгоритм кластеризації, який працює на основі щільності точок в просторі. Метод DBSCAN онлайн – це адаптація класичного DBSCAN для обробки потокових даних в режимі реального часу.

Давайте розглянемо базовий алгоритм DBSCAN та його модифікації для роботи з потоковими даними [17].

2.2.1 Базовий та модифікований алгоритм DBSCAN

Розглянемо базовий алгоритм DBSCAN. Спочатку треба визначити сусідів. Нехай P – поточна точка. Знайдем всі точки, які знаходяться на відстані менше або рівно ε від точки P. Після чого знайдемо основні точки, якщо кількість сусідів точки P більше або рівна $MinPts$, тоді P – основна точка. Сформуємо кластери, додаючи всі точки, які можуть бути досягнуті в процесі переходу від основної точки до іншої основної точки. Точки, які не є основними і не мають сусідів, які були б основними, вважаються шумом.

Розглянемо модифікований алгоритм DBSCAN. Розпочнемо з додавання нових точок, якщо нова точка надходить в потік, вона додається до списку точок, і алгоритм спробує оновити кластери [18]. Після чого оновлюємо кластери. Оновлення кластерів відбувається в процесі адаптації до нових даних. Це може включати в себе додавання нових точок до існуючих кластерів або створення нових кластерів. Для збереження ефективності може виконуватися вилучення застарілих точок або кластерів, які втратили актуальність (табл. 2.2)

Таблиця 2.2 – Порівняння методу DBSCAN та його модифікації

Характеристика	**DBSCAN**	**Модифікація**
Принцип роботи	Базується на визначенні густини точок упросторі та виділенні кластерів	Розширюєпринцип DBSCAN, дозволяючи виявлення кластерів різної форми та розміру
Виділення кластерів	Потребує попереднього визначення параметрів, такихяк радіус і мінімальна кількість точок	Може автоматично визначати параметри, спираючись на локальну густинуточок

Продовження таблиці 2.2

Гнучкість	Може маючи з виявленням кластерів різної густини	Здатний адаптуватися до зміни густини та форми кластерів
Розмір кластерів	Найбільш ефективний для кластерів однакового розміру	Здатний ефективно працювати з кластерами різного розміру
Шумові точки	Чутливий до шумових точок та потребує визначення порогового значення	Менш чутливий до шумових точок, враховує їх в структурі кластера
Ієрархічна структура	Не має ієрархічної структури	Може будувати ієрархічні структури кластерів

2.2.2 Переваги та недоліки DBSCAN онлайн

Метод онлайн достовірної кластеризації даних DBSCAN має свої переваги та недоліки. Розглянемо його переваги.

DBSCAN онлайн працює в режимі реального часу. Він здатний кластеризувати дані, які приходять в потік в режимі реального часу, оновлюючи кластери при появі нових даних.

Цей метод автоматично визначає кількість кластерів, він не потребує визначення кількості кластерів заздалегідь, оскільки вони формуються автоматично відповідно до структуриданих. DBSCAN здатний ідентифікувати точки, які не приналежать жодному кластеру, і вважаються шумом. Він динамічно адаптується до змін в щільності та структурі даних, оновлюючи кластери за необхідності. Також цей метод не вимагає глобальної обробки даних. Він працює локально і не потребує обробки всього набору даних для оновлення кластерів [19].

Серед недоліків можна зазначити чутливість до параметрів та обчислювальну складність. Метод чутливий до правильного вибору параметрів, таких як радіус та мінімальна кількість точок для формування кластеру. Може бути обчислювально витратним при великій кількості точок та зростанні розмірів даних. Також його важко пристосувати до динамічних змін. При інтенсивних та швидких змінах в потоці даних може бути важко підтримувати актуальні кластери. Цей метод має залежність від метрики відстані, тобто вибір певної метрики відстані може впливати на результати кластеризації. Особливо витратний при роботі з великими обсягами даних а даними високої щільності [20].

DBSCAN онлайн є потужним інструментом для обробки потокових даних, але при цьому необхідно ретельно налаштовувати параметри та враховувати його обчислювальну складність.

2.3 Спектральна кластеризація онлайн

Спектральна кластеризація є ефективним методом для виявлення кластерів у графах або матрицях відстаней. Однак спектральна кластеризація не є безпосередньо придатною для онлайн-кластеризації, оскільки вона вимагає обчислення власних векторів або матриці власних значень, що може бути витратним обчислювальним завданням для великих обсягів даних або в режимі реального часу.

Проте, існують методи адаптації спектральної кластеризації для обробки потокових даних. Одним із підходів є використання методу зменшення розмірності даних перед застосуванням спектральної кластеризації [21]. Це дозволяє обробляти дані онлайн, адже зменшення розмірності може бути виконано швидше та легше (рис. 2.2).

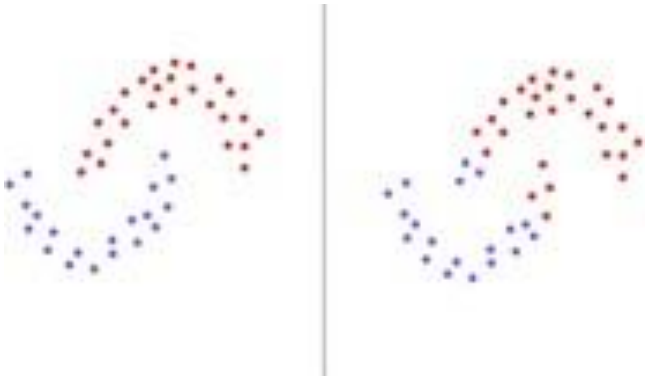

Рисунок 2.2 – Приклад роботи Спектральної кластеризації онлайн

2.3.1 Основні кроки спектральної кластеризації онлайн

Спектральна кластеризація даних має такі кроки: побудова графа або матриці відстаней, зменшення розмірності, обчислення власних векторів та власних значень, вибір кількості кластерів та кластеризація. Розглянемо більш детально ці кроки.

Перший крок це побудова графа аобо матриці відстаней. Ми створюємо граф, де вузли представляють дані, а ребра відповідають схожості між точками. Зазвичай використовується гаусівське ядро [22].

Другий крок це зменшення розмірності. Застосовуємо методи зменшення розмірності, такі як метод головних компонент PCA або вбудовані методи відбору ознак для скорочення обсягу обчислень.

Після чого переходимо до наступного кроку обчислення власних векторів та власних значень. Обчислюємо власні вектори та власні значення для матриці, що виникає після зменшення розмірності.

Останнім кроком є вибір кількості кластерів та кластеризація. Вибираємо кількість власних векторів, що відповідають найбільшим власним значенням, та застосовуємо кластеризацію до цих векторів [23].

Спектральна кластеризація онлайн ставить перед собою виклики: обчислювальна складність, адаптація до потокових даних, залежність від параметрів та розмірність даних.

Обчислювальна складність полягає в тому, що обчислення власних векторів та власних значень може бути витратним завданням, особливо для великих обсягів даних [24].

Методи спектральної кластеризації необхідно адаптувати до потокових даних, це може бути складним завданням.

Методи спектральної кластеризації залежать від параметрів, тому необхідно правильно налаштувати параметри, такі як кількість власних векторів або параметри методу зменшення розмірності.

Важливо, що зменшення розмірності може призвести до втрати інформації, особливо у високорозмірних просторах.

Спектральна кластеризація онлайн є важливим напрямком досліджень, і використання методів зменшення розмірності є одним із можливих шляхів адаптації для роботи з потоковими даними [25].

2.3.2 Переваги та недоліки спектральної кластеризації онлайн

Методи спектральної кластеризації онлайн мають свої переваги та недоліки. Розглянемо його переваги.

Спектральна кластеризація може ефективно розпізнавати та визначати нелінійні структури даних. При цьому немає потреби заздалегіть задавати кількість кластерів, оскільки це визначається власними векторами та власними значеннями. При низькорозмірних просторах метод працює добре, де інщі методи можуть дати неефективний результат.

Спектральна кластеризація онлайн добре підходить для обробки графових даних, де точки представлені вузлами, а ребра – схожістю між вузлами. Метод здатний враховувати як глобальні, так і локальні структури даних, що дозволяє виявити різні типи кластерів [26].

Серед недоліків можна зазначити високу обчислювальну складність, залежність від вибору параметрів, чутливість до шуму та викидів, важкість роботи з високорозмірними даними, важкість адаптації до потокових даних та сенситивність до великих вартостей схожості.

Обчислення власних векторів та власних значень може бути витратним завданням, особливо для великих наборів даних. Ефективність спектральної кластеризації може залежати від правильного вибору параметрів, таких як кількість власних векторів або використання ядер. Спектральна кластеризація може бути чутливою до шуму та викидів в даних [27]. Також при роботі з високорозмірними даними може виникати проблема великої вимірності, яка може призвести до менш ефективної кластеризації. Метод протребує адаптації для роботи з потоковими даними, оскільки обчислення власних векторів може бути кладним завданням при динамічних змінах. Останнім недоліком методу є те, що великі значення схожості між деякими точками можуть призводити до гомогенних кластерів.

Спектральна кластеризація – це потужний метод, але його використання вимагає уважної настройки та розуміння особливостей ваших даних.

2.4 Ієрархічна кластеризація даних онлайн

Метод ієрархічної кластеризації даних – алгоритм машинного навчання, який групує схожі дані в ієрархічні структури або дерева кластерів. Цей метод може бути використаний для визначення відносин між різними елементами набору даних та створення ієрархії груп або кластерів [28].

Існує основних два підходи для ієрархічної кластеризації даних: агломеративний підхід та дегломеративний підхід.

Ієрархічна кластеризація може бути представлена у вигляді дендрограми, яка візуалізує ієрархію об’єднання або розщеплення кластерів [29].

2.4.1 Агломеративний та дегломеративний підхід

Агломеративний підхід або підхід знизу вгору є однією з двох основних стратегій цього методу. Цей підхід використовується для поетапного об’єднання окремих елементів або кластерів в більші групи, аж до того моменту, коли всі елементи даних об’єднаються в один загальний кластер. Процес агломеративної ієрархічної кластеризації можна уявити як «знизу вгору» – з початкових елементів до одного об’єднаного кластера. Основні кроки агломеративного підходу: початок, схожість між кластерами, об’єднання, оновлення матриці схожості, повторення та створення ієрархії [30].

Розглянемо більш детально. Кожен елемент даних розглядається як окремий кластер. Після чого визначається міра схожості між кожною парою кластерів. Далі об’єднуються два найбільш схожі кластери у новий кластер та перераховуються схожості між новим кластером і залишковими кластерами. Кроки 3–4 повторюються, доки всі елементи не об’єднаються в один кластер. Після чого процес створює ієрархію кластерів, яку можна представити у вигляді дендрограми [31].

Агломеративний підхід дозволяє отримати ієрархічну структуру, яка може бути використана для подальшого аналізу та визначення оптимального рівня кластеризації в залежності від конкретних потреб аналізу даних.

Дегломеративний підхід в ієрархічній кластеризації є іншою стратегією цього методу.

У цьому підході всі елементи даних спочатку розглядаються як один великий кластер, який поступово розщеплюється на менші підкластери досягнення бажаної кількості кластерів. Цей процес можна уявити як «зверху вниз» – від одного загального кластера до більш дрібних підгруп [32]. Основні кроки дегломеративного підходу: початок, схожість між елементами, розщеплення, оновлення матриці схожості, повторення, створення ієрархії.

Розглянемо більш детально. Спочатку усі елементи данихрозглядаються як один кластер. Після чого визначається міра схожості між кожною парою елементів. Наступним кроком буде знаходження двох найменш схожих елементів та об’єднуються у новий кластер. Далі перераховуються схожості між новим кластером і залишковими елементами. Кроки 3–4 повторюються, доки не досягнуто бажаної кількості кластерів. Після чого процес створює ієрархію кластерів, представлену у вигляді дендрограми.

Дегломеративний підхід дозволяє отримати ієрархічну структуру, де кожен рівень представляє собою підгрупу, і може бути корисним для аналізу даних на різних рівнях деталізації.

2.4.2 Переваги та недоліки ієрархічної кластеризації онлайн

Метод групування даних, який працює в режимі реального часу, динамічно оновлюючи кластери при появі нових даних. Ось деякі переваги: адаптивність до змін, ефективність для великих потоків даних, створення ієрархії, масштабованість. Розглянемо більш детально [33].

Онлайн ієрархічна кластеризація дозволяє динамічно адаптуватися до змін у вхідних даних. Вона легко впорається з появою нових об’єктів або зміною характеристик існуючих. Також цей метод може бути ефективним для обробки великих потоків даних, оскільки не вимагає повторного обчислення всіх кластерів при кожному оновленні.

Він дозволяє створювати ієрархічні структури, що можуть вказувати на складні відносини між різними групами об’єктів. Може бути масштабованим для роботи з великою кількістю даних, оскільки обчислення можуть бути розподілені на кластерах або використовувати апаратне прискорення.

Серед недоліків можна зазначити: чутливість до порядку даних, втрату глобальної оптимальності, високу обчислювальну складність, потребу в параметрах. Розглянемо більш детально [34].

Результати ієрархічної кластеризації можуть суттєво змінюватися в залежності від порядку, в якому дані надходять. Це може призвести до різних структур кластерів при різних періодах часу. Також оскільки кластери оновлюються динамічно, можливе втратити глобальну оптимальність, особливо якщо нові дані суттєво відрізняються від вже існуючих. Обчислення ієрархічної кластеризації може бути витратними з обчислювальної точки зору, особливо при великій кількості кластерів та об’єктів [31-36]. Деякі алгоритми ієрархічної кластеризації вимагають задання параметрів, таких як відстань або кількість кластерів, що може бути складно визначити в онлайн-режимі.

У кожного підходу до ієрархічної кластеризації є свої переваги і недоліки, і вибір між ними залежить від конкретного контексту завдання та характеристик даних.

2.5 Методи потікого-графового аналізу

Методи потокового-графового аналізу є важливим інструментарієм для вивчення та розуміння динамічних взаємодій у складних системах. Ці методи зазвичай використовуються для аналізу даних, які надходять у вигляді потоку, де структура графа може змінюватися з часом. Нижче наведено деякі основні аспекти методів потікого-графового аналізу: графовий представлення потоку даних, динамічні графові метрики, виявлення змін у графовій структурі, аналіз та виявлення аномалій, використання мережевої статистики, застосування машинного навчання. Розберемо більш детально [35].

Спочатку дані подаються у вигляді графа, де вузли відображають сутності, а ребра представляють зв’язки між ними. У потічних графах важливо враховувати часові мітки або порядок подій. Використання графових метрик, які враховують часові аспекти, такі як часовий інтервал між подіями, частота зміни графової структури, стабільність кластерів тощо. Методи для виявлення змін у графовій структурі з часом, такі як виявлення нових вузлів або ребер, розриви в графі, зміни в кластеризації тощо. Аналіз аномалій в потічних графах для виявлення незвичайних та потенційно важливих подій, які можуть свідчити про несподівані зміни у системі. Застосування мережевої статистики для розуміння взаємодій між елементами системи, а також для виявлення ключових графових вузлів чи патернів. Використання методів машинного навчання для передбачення змін у графі чи виявлення аномалій на основі історичних даних [36].

Методи потікого-графового аналізу грають важливу роль в обробці та аналізі даних в реальному часі, де системи взаємодіють із змінними умовами та надходять великі обсяги інформації у формі потоку. Як і інщі методи потікого-графові мають як переваги та і недоліки.

Переваги методів потікого-графового аналізу: врахування динамічності даних, адаптивність до змін, виявлення аномалій, здатність працювати в реальному часі, можливість прогнозування. Методи потікого-графового аналізу враховують часові аспекти та динамічні зміни у графовій структурі, що дозволяє ефективно моделювати та аналізувати змінні системи. Ці методи дозволяють адаптуватися до змін у структурі графа під час аналізу, виявляючи нові елементи та враховуючи їх вплив на систему. Методи потікого-графового аналізу можуть виявляти аномалії та несподівані патерни, що допомагає вчасно реагувати на негативні явища чи надто активні події. Багато методів спроектовані для роботи в режимі реального часу, дозволяючи швидко реагувати на зміни та отримувати актуальну інформацію [37]. Застосування методів машинного навчання в методах потікого-графового аналізу дозволяє робити прогнози та передбачати подальший розвиток ситуації.

Недоліки методів потікого-графового аналізу: обчислювальна складність, чутливість до параметрів, потреба у великій кількості даних, складність інтерпретації, вплив шуму та викидів. Обчислення та аналіз потічних графів може бути обчислювально витратним завданням, особливо при великих обсягах даних та складних графових структурах. Деякі методи потікого-графового аналізу можуть бути чутливими до вибору параметрів, таких як розмір вікна чи часовий інтервал, що вимагає уважної настройки [38]. Методи, зокрема методи машинного навчання, можуть вимагати значної кількості даних для ефективного навчання та прогнозування. З великою кількістю динамічних змін та взаємодій, інтерпретація результатів потікого-графового аналізу може бути складною та вимагати експертного розуміння. Методи можуть бути чутливими до шуму та викидів у потічних даних, що може впливати на точність та надійність аналізу [39].

Для успішного використання методів потікого-графового аналізу важливо уважно вибирати методи, враховуючи конкретні умови задачі та особливості даних [40].

3 ПОКРАЩЕНИЙ АЛГОРИТМ КЛАСТЕРИЗАЦІЇ ДАНИХ НА ОСНОВІ МЕТОДУ *K*-MEANS

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був покращений алгоритм онлайн достовірної кластеризації даних на основі методу *K*-Means. Алгоритм наоснові методу *K*-Means був покращений для роботи з нестаціонарними даними та великим обсягом інформації. Для реалізації було обране середовище Visual Studio Code. Це обумовлене тим, що він має інтуїтивний і зручний інтерфейс, що полегшує використання та навігацію. Він підтримує багато мов програмування, включаючи C++, яка часто використовується для реалізації алгоритмів машинного навчання.

VS Code дозволяє легко інтегрувати розширення та плагіни, що може бути корисним для роботи з бібліотеками та інструментами, які вам можуть знадобитися для реалізації та оптимізації алгоритму *k*-Means. Інтеграція з системами контролю версій, зокрема з Git, в VS Code дозволяє зберігати та відстежувати зміни у ваших програмних проектах. Велика кількість розширень VS Code дозволяє вам вибрати та встановити інструменти, які відповідають вашим потребам.

Якщо ви використовуєте C++ для реалізації алгоритму *k*-Means, то VS Code надає розширену підтримку для C++, таку як автодоповнення коду, перевірка синтаксису, відлагодження та інші корисні функції.

Для вирішення цього питання ми використовуємо мову програмування C++. Мова програмування C++ має кілька характеристик, які роблять її добрим вибором для вирішення задач кластеризації даних. C++ – це мова з низькорівневим доступом до пам'яті та високою продуктивністю. Це особливо важливо при обробці великих обсягів даних, що зазвичай виникає при кластеризації. Завдяки властивостям мови C++, ви можете ефективно управляти пам'яттю та іншими ресурсами, що дозволяє оптимізувати

використання обчислювальних ресурсів. C++ має велику кількість бібліотек для обробки числових обчислень, алгоритмів та структур даних, які можуть бути використані для реалізації алгоритмів кластеризації. Мова C++ надає можливість оптимізувати код на рівні системи, що означає, що ви можете максимально використовувати апаратні можливості конкретної платформи для прискорення обчислень. Код, написаний на C++, може бути досить портативним, що означає, що ви можете використовувати його на різних платформах з невеликими або нульовими змінами. Мова C++ має велику та активну спільноту, яка розвиває багато корисних бібліотек та інструментів для наукових обчислень, включаючи задачі кластеризації даних. Використання об'єктно–орієнтованого підходу дозволяє структурувати код і полегшує розширення та обслуговування.

Вищезазначені характеристики роблять мову C++ ефективним засобом для розробки швидких та продуктивних алгоритмів кластеризації даних, особливо в тих випадках, коли потрібна висока продуктивність та оптимізація ресурсів (рис. 3.1).

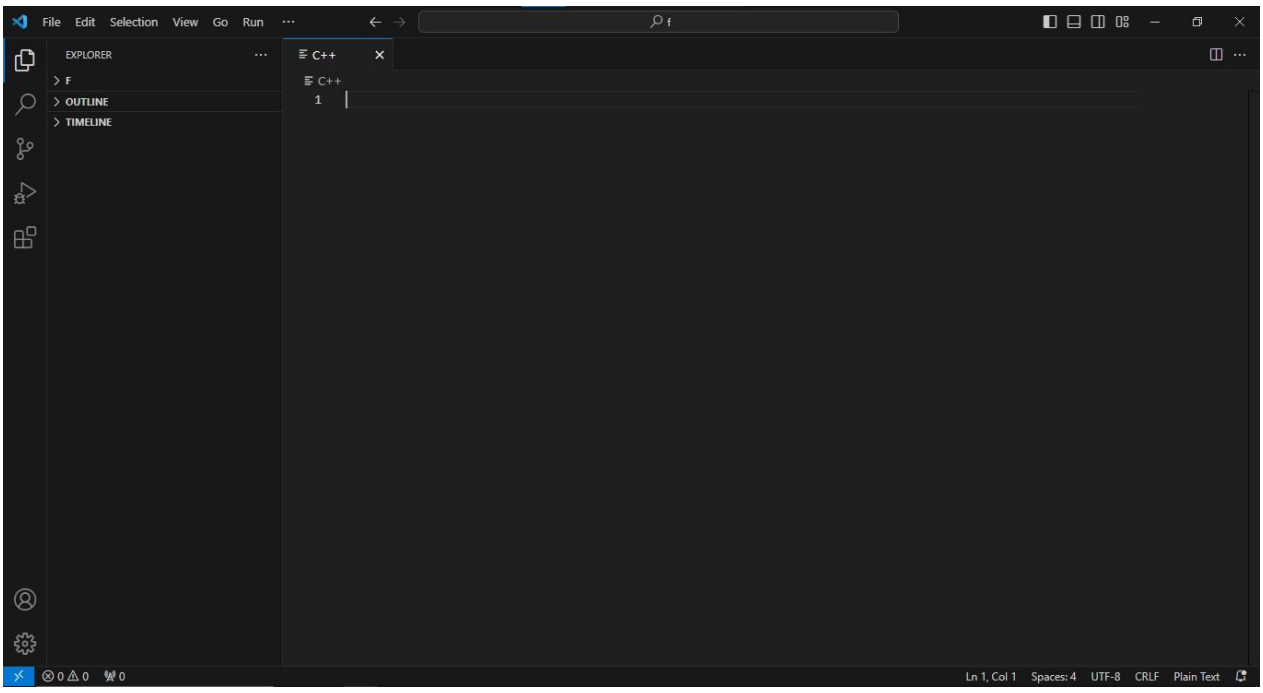

Рисунок 3.1 – Приклад інтерфейсу Visual Studio Code

Загалом, VS Code є популярним та зручним середовищем для багатьох розробників, і воно добре підходить для написання та вдосконалення алгоритмів машинного навчання, включаючи алгоритм *K*–Means.

3.2 Програмна реалізація

Питання наступне: визначити, скільки на даному знімку ділянок, залитих приблизно одним кольором. Це зовсім нескладно: білі пелюстки – один, жовті центри – два, зелень – три. Ці ділянки називають кластерами.

Існує багато алгоритмів кластеризації, але найпростіший з них – *K*-Means. Дані, які ми будемо розподіляти по кластерах – це пікселі. Як відомо, кольоровий піксель має три компоненти – червоний, зелений і синій. Накладання цих компонентів і створює палітру існуючих кольорів (рис. 3.2).

Рисунок 3.2 – Приклад палітри

У пам’яті комп’ютера кожна складова кольору характеризується числом від 0 до 255. Іншими словами, комбінуючи різні значення червоного, зеленого і синього, ми отримуємо палітру кольорів на екрані.

На прикладі пікселів ми реалізуємо наш алгоритм. *K*-Means – ітераційний алгоритм, що означає, що він надає правильний результат після певної кількості повторень певних математичних обчислень. Розглянемо алгоритм за яким ми будемо кластеризувати.

Необхідно заздалегідь знати, на скільки кластерів треба розподілити дані. Це є суттєвим недоліком цього методу, але ця проблема вирішується вдосконаленими реалізаціями алгоритму.

Потрібно вибрати випадковим чином початкові центри наших кластерів. Щоб можна було прив'язувати кожен піксель до центру кластера. Навколо центру збираються пікселі. Саме «відстань» від центру до пікселя визначає, кого підкорятиме кожен піксель. Порахуємо відстань від кожного центра до кожного пікселя. Ця відстань розглядається як євклідова відстань між точками в просторі, а в нашому випадку – як відстань між трьома компонентами кольору.

$$\sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}. \qquad (3.1)$$

Ми обчислюємо відстань від першого пікселя до кожного центру і визначаємо найменшу відстань між цим пікселем і центрами. Для центра, відстань до якого є найменшою, перераховуємо координати як середнє арифметичне між кожною складовою пікселя – короля і пікселя – підданого.

Після перерахунку всіх центрів ми розподіляємо пікселі по кластерах, порівнюючи відстань від кожного пікселя до центрів. Піксель розміщується в кластері, до центру якого він розташований ближче, ніж до інших центрів.

Все починається спочатку, поки пікселі залишаються в тих самих кластерах. Це може не трапитися, оскільки при великій кількості даних центри будуть зміщуватися в невеликому радіусі, і пікселі по краях кластерів будуть переходити то в один, то в інший кластер. Для цього потрібно визначити максимальну кількість ітерацій.

У першому файлі під назвою «k_means.h» я визначив основні типи даних, константи і основний клас для виконання завдань – «K_means».

Для опису кожного пікселя створено структуру, що складається з трьох компонентів пікселя. Для цих компонентів обрано тип double для більш точних обчислень і визначив кілька констант для роботи програми.

Лістинг 3.1 Оголошення змінних:

```
const int KK = 10;  //кількість кластерів
const int max_iterations = 100;  //максимальна кількість ітерацій

typedef struct {
    double r;
    double g;
    double b;
} rgb;
```

У цьому фрагменті коду оголошено константи та визначено структуру для представлення кольору у форматі RGB. Давайте розглянемо його детальніше:

Крок 1. const int KK = 10;. Оголошення константи KK, яка представляє кількість кластерів. За замовчуванням встановлено значення 10, і ця константа буде використовуватися для визначення кількості кластерів у алгоритмі кластеризації.

Крок 2. const int max_iterations = 100;. Оголошення константи max_iterations, яка представляє максимальну кількість ітерацій у алгоритмі. За замовчуванням встановлено значення 100. Ця константа визначає, скільки разів алгоритм може виконати ітерації для підбору кластерів.

Крок 3. typedef struct { double r; double g; double b; } rgb;. Визначення структури rgb, яка містить три поля для представлення компонентів кольору у форматі RGB. Кожен компонент (червоний, зелений, синій) представлений значенням з плаваючою точкою типу double. Це структура буде використовуватися для зберігання кольорової інформації про пікселі або центри кластерів.

Лістинг 3.2 Клас *K*-Means:

```
class K_means
{
private:
    std::vector<rgb> pixcel;
    int q_klaster;
    int k_pixcel;
    std::vector<rgb> centr;
    void identify_centers();
    inline double compute(rgb k1, rgb k2)
    {
        return sqrt(pow((k1.r - k2.r),2) + pow((k1.g - k2.g), 2) + pow((k1.b -
k2.b), 2));
    }
    inline double compute_s(double a, double b) {
        return (a + b) / 2;
    };
public:
    K_means() : q_klaster(0), k_pixcel(0) {};
    K_means(int n, rgb *mas, int n_klaster);
    K_means(int n_klaster, std::istream & os);
    void clustering(std::ostream & os);
    void print()const;
    ~K_means();
    friend std::ostream & operator<<(std::ostream & os, const K_means & k);
};
```

Розглянемо компоненти класу: vectorpixcel, q_klaster, k_pixcel, vectorcentr, identify_centers(), compute() та compute_s(), clustering(std::ostream & os):

Крок 1. vectorpixcel – вектор для зберігання пікселів.

Крок 2. q_klaster – кількість кластерів.

Крок 3. k_pixcel – кількість пікселів.

Крок 4. vectorcentr – вектор для зберігання центрів кластеризації, кількість елементів визначається q_klaster.

Крок 5. identify_centers() – метод для випадкового вибору початкових центрів серед вхідних пікселів.

Крок 6. compute() та compute_s() – вбудовані методи для розрахунку відстані між пікселями та перерахунку центрів відповідно.

Крок 7. конструктори: перший за замовчуванням, для ініціалізації пікселів масиву, третій – ініціалізація пікселів із текстового файлу.

Крок 8. clustering(std::ostream & os) – метод для кластеризації.

Лістинг 3.3 Реалізація методу identify_centers():

```
void K_means::identify_centers()
{
    srand((unsigned)time(NULL));
    rgb temp;
    rgb *mas = new rgb[q_klaster];
    for (int i = 0; i < q_klaster; i++) {
        temp = pixcel[0 + rand() % k_pixcel];
        for (int j = i; j < q_klaster; j++) {
            if (temp.r != mas[j].r && temp.g != mas[j].g && temp.b != mas[j].b) {
                mas[j] = temp;
            }
            else {
                i--;
                break;
            }
        }
    }
    for (int i = 0; i < q_klaster; i++) {
        centr.push_back(mas[i]);
    }
    delete []mas;
}
```

Реалізовано метод identify_centers() класу K_means, який призначений для випадкового вибору початкових центрів кластерів. Давайте розглянемо, що відбувається у цьому методі. Метод srand((unsigned)time(NULL)).

Ініціалізація генератора випадкових чисел від часу. Створення тимчасового об'єкта rgb temp; для зберігання тимчасового значення пікселя. Створення динамічного масиву rgb *mas = new rgb[q_klaster]; розміром q_klaster для зберігання вибраних центрів. Потім проводиться цикл для вибору початковихцентрів temp = pixcel[0 + rand() % k_pixcel];. Вибір випадкового пікселя з масиву pixcel. Вкладений цикл, який перевіряє умову та забезпечує унікальність центрів, якщо вибраний піксель не співпадає з будь-яким із попередньо вибраних центрів, то він зберігається у масиві mas. Якщо ж вибраний піксель вже був вибраний як центр, то цикл знову вибирає піксель

для того ж центру, і зменшує значення i, щоб повторити процес вибору для цього центру. Потім йде додавання обраних центрів у вектор центрів centr. Останнім кроком є очищення пам'яті від динамічного масиву mas. Отже, цей метод генерує та обирає унікальні початкові центри кластерів для подальшої роботи алгоритму.

Лістинг 3.4 Реалізація конструктору:

```
K_means::K_means(int n, rgb * mas, int n_klaster)
{
    for (int i = 0; i < n; i++) {
        pixcel.push_back(*(mas + i));
    }
    q_klaster = n_klaster;
    k_pixcel = n;
    identify_centers();}
```

У цьому фрагменті коду реалізований конструктор для класу K_means, призначеного для реалізації алгоритму *k*-середніх. Давайте розглянемо, що відбувається:

Крок 1. K_means::K_means(int n, rgb * mas, int n_klaster). Це оголошення конструктора класу K_means, який приймає три параметри: *n* – кількість елементів у масиві mas, mas – вказівник на масив структур типу rgb, n_klaster – кількість кластерів, які слід сформувати.

Крок 2. for (int i = 0; i < n; i++) { pixcel.push_back(*(mas + i)); }. Цикл для копіювання кожного елемента масиву mas (який містить структури rgb) до вектору pixcel у межах вказаної кількості n. Кожен елемент mas копіюється в кінець вектору pixcel.

Крок 3. q_klaster = n_klaster;. Присвоєння значення змінній q_klaster, яка представляє кількість кластерів у алгоритмі, значення, яке було передано як аргумент конструктора (n_klaster).

Крок 4. k_pixcel = n;. Присвоєння значення змінній k_pixcel, яка представляє загальну кількість пікселів у векторі pixcel. Значення отримується з аргументу конструктора (*n*).

Крок 5. identify_centers();. Виклик методу identify_centers(), який, ймовірно, визначає початкові центри кластерів або здійснює інші дії, пов'язані з підготовкою алгоритму *k*-середніх до роботи.

Цей конструктор допомагає ініціалізувати об'єкт класу K_means з вхідними даними та підготовити його до виконання алгоритму кластеризації.

Лістинг 3.5 Ініціалізація об’єкта класу:

```
K_means::K_means(int n_klaster, std::istream & os) : q_klaster(n_klaster)
{
    rgb temp;
    while (os >> temp.r && os >> temp.g && os >> temp.b) {
        pixcel.push_back(temp);
    }
    k_pixcel = pixcel.size();
    identify_centers();
}
```

У цьому конструкторі K_means відбувається ініціалізація об'єкта класу за допомогою вхідних параметрів. Розглянемо кожен етап:

Етап 1. for (int i = 0; i < n; i++) { pixcel.push_back(*(mas + i)); }. Цикл, який додаєкожен елемент масиву mas (типу rgb) до вектору pixcel класу K_means. Таким чином, пікселі із вхідного масиву копіюються у внутрішній вектор pixcel об’єкта класу.

Етап 2. q_klaster = n_klaster;: присвоєння значення n_klaster змінній q_klaster, що представляє кількість кластерів.

Етап 3. k_pixcel = n;: присвоєння значення n змінній k_pixcel, що представляє кількість пікселів.

Етап 4. identify_centers();: виклик методу identify_centers(), який вибирає початкові центри кластерів серед вже ініціалізованих пікселів.

Лістинг 3.6 Реалізація методу clustering:

```
void K_means::clustering(std::ostream & os)
{
    os << "\n\nНачало кластеризации:" << std::endl;
```

```
    std::vector<int> check_1(k_pixcel, -1);
    std::vector<int> check_2(k_pixcel, -2);
    int iter = 0;

    while(true)
    {
        os << "\n\n---------------- Итерация №"
            << iter << " ----------------\n\n";
        {
            for (int j = 0; j < k_pixcel; j++) {
                double *mas = new double[q_klaster];

                for (int i = 0; i < q_klaster; i++) {
                    *(mas + i) = compute(pixcel[j], centr[i]);
                    os << "Расстояние от пикселя " << j << " к центру #"
                        << i << ": " << *(mas + i) << std::endl;
                }

                double min_dist = *mas;
                int m_k = 0;
                for (int i = 0; i < q_klaster; i++) {
                    if (min_dist > *(mas + i)) {
                        min_dist = *(mas + i);
                        m_k = i;
                    }
                }

                os << "Минимальное расстояние к центру #" << m_k << std::endl;
                os << "Пересчитываем центр #" << m_k << ": ";
                centr[m_k].r = compute_s(pixcel[j].r, centr[m_k].r);
                centr[m_k].g = compute_s(pixcel[j].g, centr[m_k].g);
                centr[m_k].b = compute_s(pixcel[j].b, centr[m_k].b);
                os << centr[m_k].r << " " << centr[m_k].g
                    << " " << centr[m_k].b << std::endl;
                delete[] mas;
            }

            int *mass = new int[k_pixcel];
            os << "\nПроведем классификацию пикселей: "<< std::endl;
            for (int k = 0; k < k_pixcel; k++) {
                double *mas = new double[q_klaster];

                for (int i = 0; i < q_klaster; i++) {
                    *(mas + i) = compute(pixcel[k], centr[i]);
                    os << "Расстояние от пикселя №" << k << " к центру #"
                        << i << ": " << *(mas + i) << std::endl;
                }

                double min_dist = *mas;
                int m_k = 0;
                for (int i = 0; i < q_klaster; i++) {
```

```
                    if (min_dist > *(mas + i)) {
                        min_dist = *(mas + i);
                        m_k = i;
                    }
                }
                mass[k] = m_k;
                os << "Пиксель №" << k << " ближе всего к центру #" << m_k <<
std::endl;
            }


            os << "\nМассив соответствия пикселей и центров: \n";
            for (int i = 0; i < k_pixcel; i++) {
                os << mass[i] << " ";
                check_1[i] = *(mass + i);
            }
            os << std::endl << std::endl;

            os << "Результат кластеризации: " << std::endl;
            int itr = KK + 1;
            for (int i = 0; i < q_klaster; i++) {
                os << "Кластер #" << i << std::endl;
                for (int j = 0; j < k_pixcel; j++) {
                    if (mass[j] == i) {
                        os << pixcel[j].r << " " << pixcel[j].g
                            << " " << pixcel[j].b << std::endl;
                        mass[j] = ++itr;
                    }
                }
            }

            delete[] mass;

            os << "Новые центры: \n" ;
            for (int i = 0; i < q_klaster; i++) {
                os << centr[i].r << " " << centr[i].g
                    << " " << centr[i].b << " - #" << i << std::endl;
            }
        }

        iter++;
        if (check_1 == check_2 || iter >= max_iterations) {
            break;
        }
        check_2 = check_1;
    }
    os << "\n\nКонец кластеризации." << std::endl;
}
```

У цьому фрагменті коду реалізовано метод clustering(std::ostream & os) класу K_means, який виконує кластеризацію та виводить інформацію про кожну ітерацію у вказаний потік виведення os. Давайте розглянемо кожен етап цього методу:

Крок 1. os << "\n\nНачало кластеризации:" << std::endl;. Виведення інформації про початок процесу кластеризації у вказаний потік виведення os.

Крок 2. Ініціалізація двох векторів check_1 і check_2 розміром k_pixcel, обидва заповнені значеннями -1 та -2 відповідно. Вони використовуються для порівняння індексів кластерів на попередній та поточній ітераціях.

Крок 3. int iter = 0;. Ініціалізація лічильника ітерацій.

Крок 4. while(true): безкінечний цикл для виконання ітерацій кластеризації.

Крок 5. for (int j = 0; j < k_pixcel; j++): зовнішній цикл для обробки кожного пікселя на кожній ітерації.

Крок 6. Створення динамічного масиву mas розміром q_klaster для зберігання відстаней від поточного пікселя до кожного центру кластеру.

Крок 7. Внутрішній цикл для розрахунку відстаней від поточного пікселя до кожного центру кластеру та виведення цих відстаней у вказаний потік.

Крок 8. Знаходження мінімальної відстані та відповідного індексу кластеру.

Крок 9. Оновлення інформації про мінімальну відстань та перерахунок центру кластеру, до якого належить поточний піксель.

Крок 10. Очищення пам’яті від динамічного масиву mas.

Крок 11. Створення масиву mass розміром k_pixcel для зберігання індексів кластерів, до яких належать пікселі після кожної ітерації.

Крок 12. Цикл для розрахунку відстаней від кожного пікселя до кожного центру кластеру, визначення мінімальної відстані та відповідного індексу кластеру, а також виведення цих відстаней у вказаний потік.

Крок 13. Запис індексу кластеру, до якого належить кожен піксель, у масив mass.

Крок 14. Виведення інформації про індекси кластерів у вказаний потік.

Крок 15. Виведення інформації про кожен кластер, включаючи пікселі, що до нього належать.

Крок 16. Оновлення вектора check_1 із збереженням індексів кластерів після кожної ітерації.

Крок 17. Виведення інформації про нові центри кластерів.

Крок 18. Збільшення лічильника ітерацій.

Крок 19. Перевірка умови виходу з циклу: якщо індекси кластерів на попередній та поточній ітераціях співпадають (check_1 == check_2) або досягнуто максимальну кількість ітерацій (iter >= max_iterations), то вихід з циклу.

Крок 20. Оновлення вектора check_2 із збереженням індексів кластерів на поточній ітерації.

Крок 21. Завершення циклу while та виведення інформації про завершення кластеризації.

Отже, метод виконує ітерації кластеризації, виводячи інформацію про відстані, оновлення центрів кластерів та приналежність кожного пікселя до кластера на кожній ітерації.

Лістинг 3.7 Реалізація перегрузки оператора:

```
std::ostream & operator<<(std::ostream & os, const K_means & k)
{
    os << "Начальные пиксели: " << std::endl;
    for (int i = 0; i < k.k_pixcel; i++) {
        os << k.pixcel[i].r << " " << k.pixcel[i].g
            << " " << k.pixcel[i].b << " - №" << i << std::endl;
    }
    os << std::endl << "Случайные начальные центры кластеризации: " << std::endl;
    for (int i = 0; i < k.q_klaster; i++) {
        os << k.centr[i].r << " " << k.centr[i].g << " "
            << k.centr[i].b << " - #" << i << std::endl;
    }
```

```
    os << "\nКоличество кластеров: " << k.q_klaster << std::endl;
    os << "Количество пикселей: " << k.k_pixcel << std::endl;
    return os;
}
```

У цьому фрагменті коду реалізовано перегрузку оператора виведення (<<) для класу K_means. Цей оператор дозволяє виводити об'єкти класу K_means у вказаний потік виведення (std::ostream). Давайте розглянемо кожен етап цього оператора:

Крок 1. os << "Начальные пиксели: " << std::endl;: виведення інформації про початкові пікселі у вказаний потік.

Крок 2. for (int i = 0; i < k.k_pixcel; i++) { ... }: цикл для виведення інформації про кожен початковий піксель, включаючи значення кожної із трьох компонент (r, g, b) та індекс пікселя.

Крок 3. os << std::endl << "Случайные начальные центры кластеризации: " << std::endl;: виведення роздільної лінії та інформації про початкові центри кластерів у вказаний потік.

Крок 4. for (int i = 0; i < k.q_klaster; i++) { ... }: цикл для виведення інформації про кожен початковий центр кластеру, включаючи значення кожної із трьох компонент (r, g, b) та індекс кластера.

Крок 5. os << "\nКоличество кластеров: " << k.q_klaster << std::endl;. Виведення інформації про кількість кластерів у вказаний потік.

Крок 6. os << "Количество пикселей: " << k.k_pixcel << std::endl;. Виведення інформації про кількість пікселів у вказаний потік.

Крок 7. return os;: повернення посилання на потік виведення, щоб дозволити подальше ланцюжкове використання оператора виведення.

Отже, цей оператор дозволяє зручно виводити дані про об’єкт класу K_means у вказаний потік для подальшого відслідковування та аналізу.

3.3 Результати

Запустивши програму та вписавши стартові показники починається кластеризація і ми отримуємо таблицю з результатами.

Лістинг 3.8 Результати кластеризації:

```
Початкові пікселі:
255 140 50 – №0
100 70 1 – №1
150 20 200 – №2
251 141 51 – №3
104 69 3 – №4
153 22 210 – №5
252 138 54 – №6
101 74 4 – №7

Випадкові початкові центри кластеризації:
150 20 200 – #0
104 69 3 – #1
100 70 1 – #2

Кількість кластерів: 3
Кількість пікселів: 8

Початок кластеризації:

Ітерація №0

Відстань від пікселя 0 до центру #0: 218.918
Відстань від пікселя 0 до центру #1: 173.352
Відстань від пікселя 0 до центру #2: 176.992
Мінімальна відстань до центру #1
Перераховуємо центр #1: 179.5 104.5 26.5
Відстань від пікселя 1 до центру #0: 211.189
Відстань від пікселя 1 до центру #1: 90.3369
Відстань від пікселя 1 до центру #2: 0
Мінімальна відстань до центру #2
Перераховуємо центр #2: 100 70 1
Відстань від пікселя 2 до центру #0: 0
Відстань від пікселя 2 до центру #1: 195.225
Відстань від пікселя 2 до центру #2: 211.189
Мінімальна відстань до центру #0
Перераховуємо центр #0: 150 20 200
Відстань від пікселя 3 до центру #0: 216.894
Відстань від пікселя 3 до центру #1: 83.933
Відстань від пікселя 3 до центру #2: 174.19
```

```
Мінімальна відстань до центру #1
Перераховуємо центр #1: 215.25 122.75 38.75
Відстань від пікселя 4 до центру #0: 208.149
Відстань від пікселя 4 до центру #1: 128.622
Відстань від пікселя 4 до центру #2: 4.58258
Мінімальна відстань до центру #2
Перераховуємо центр #2: 102 69.5 2
Відстань від пікселя 5 до центру #0: 10.6301
Відстань від пікселя 5 до центру #1: 208.212
Відстань від пікселя 5 до центру #2: 219.366
Мінімальна відстань до центру #0
Перераховуємо центр #0: 151.5 21 205
Відстань від пікселя 6 до центру #0: 215.848
Відстань від пікселя 6 до центру #1: 42.6109
Відстань від пікселя 6 до центру #2: 172.905
Мінімальна відстань до центру #1
Перераховуємо центр #1: 233.625 130.375 46.375
Відстань від пікселя 7 до центру #0: 213.916
Відстань від пікселя 7 до центру #1: 150.21
Відстань від пікселя 7 до центру #2: 5.02494
Мінімальна відстань до центру #2
Перераховуємо центр #2: 101.5 71.75 3

Проводимо класифікацію пікселів:
Відстань від пікселя №0 до центру #0: 221.129
Відстань від пікселя №0 до центру #1: 23.7207
Відстань від пікселя №0 до центру #2: 174.44
Піксель №0 найближчий до центру #1
Відстань від пікселя №1 до центру #0: 216.031
Відстань від пікселя №1 до центру #1: 153.492
Відстань від пікселя №1 до центру #2: 3.05164
Піксель №1 найближчий до центру #2
Відстань від пікселя №2 до центру #0: 0
Відстань від пікселя №2 до центру #1: 195.225
Відстань від пікселя №2 до центру #2: 211.189
Піксель №2 найближчий до центру #0
Відстань від пікселя №3 до центру #0: 219.126
Відстань від пікселя №3 до центру #1: 20.8847
Відстань від пікселя №3 до центру #2: 171.609
Піксель №3 найближчий до центру #1
Відстань від пікселя №4 до центру #0: 212.989
Відстань від пікселя №4 до центру #1: 149.836
Відстань від пікселя №4 до центру #2: 3.71652
Піксель №4 найближчий до центру #2
Відстань від пікселя №5 до центру #0: 5.31507
Відстань від пікселя №5 до центру #1: 212.176
Відстань від пікселя №5 до центру #2: 219.035
Піксель №5 найближчий до центру #0
Відстань від пікселя №6 до центру #0: 215.848
Відстань від пікселя №6 до центру #1: 42.6109
Відстань від пікселя №6 до центру #2: 172.164
```

```
Піксель №6 найближчий до центру #1
Відстань від пікселя №7 до центру #0: 213.916
Відстань від пікселя №7 до центру #1: 150.21
Відстань від пікселя №7 до центру #2: 5.02494
Піксель №7 найближчий до центру #2

Масив відповідності пікселів і центрів:
1 2 0 1 2 0 1 2

Результат кластеризації:
Кластер #0
150 20 200
153 22 210
Кластер #1
255 140 50
251 141 51
252 138 54
Кластер #2
100 70 1
104 69 3
101 74 4
Нові центри:
151.5 21 205 – #0
233.625 130.375 46.375 – #1
101.5 71.75 3 – #2
Ітерація №1

Відстань від пікселя 0 до центру #0: 221.129
Відстань від пікселя 0 до центру #1: 23.7207
Відстань від пікселя 0 до центру #2: 174.44
Мінімальна відстань до центру #1
Перераховуємо центр #1: 244.313 135.188 48.1875
Відстань від пікселя 1 до центру #0: 216.031
Відстань від пікселя 1 до центру #1: 165.234
Відстань від пікселя 1 до центру #2: 3.05164
Мінімальна відстань до центру #2
Перераховуємо центр #2: 100.75 70.875 2
Відстань від пікселя 2 до центру #0: 5.31507
Відстань від пікселя 2 до центру #1: 212.627
Відстань від пікселя 2 до центру #2: 210.28
Мінімальна відстань до центру #0
Перераховуємо центр #0: 150.75 20.5 202.5
Відстань від пікселя 3 до центру #0: 217.997
Відстань від пікселя 3 до центру #1: 9.29613
Відстань від пікселя 3 до центру #2: 172.898
Мінімальна відстань до центру #1
Перераховуємо центр #1: 247.656 138.094 49.5938
Відстань від пікселя 4 до центру #0: 210.566
Відстань від пікселя 4 до центру #1: 166.078
Відстань від пікселя 4 до центру #2: 3.88306
Мінімальна відстань до центру #2
```

```
Перераховуємо центр #2: 102.375 69.9375 2.5
Відстань від пікселя 5 до центру #0: 7.97261
Відстань від пікселя 5 до центру #1: 219.471
Відстань від пікселя 5 до центру #2: 218.9
Мінімальна відстань до центру #0
Перераховуємо центр #0: 151.875 21.25 206.25
Відстань від пікселя 6 до центру #0: 216.415
Відстань від пікселя 6 до центру #1: 6.18805
Відстань від пікселя 6 до центру #2: 172.257
Мінімальна відстань до центру #1
Перераховуємо центр #1: 249.828 138.047 51.7969
Відстань від пікселя 7 до центру #0: 215.118
Відстань від пікселя 7 до центру #1: 168.927
Відстань від пікселя 7 до центру #2: 4.54363
Мінімальна відстань до центру #2
Перераховуємо центр #2: 101.688 71.9688 3.25

Проводимо класифікацію пікселів:
Відстань від пікселя №0 до центру #0: 221.699
Відстань від пікселя №0 до центру #1: 5.81307
Відстань від пікселя №0 до центру #2: 174.122
Піксель №0 найближчий до центру #1
Відстань від пікселя №1 до центру #0: 217.244
Відстань від пікселя №1 до центру #1: 172.218
Відстань від пікселя №1 до центру #2: 3.43309
Піксель №1 найближчий до центру #2
Відстань від пікселя №2 до центру #0: 6.64384
Відстань від пікселя №2 до центру #1: 214.161
Відстань від пікселя №2 до центру #2: 209.154
Піксель №2 найближчий до центру #0
Відстань від пікселя №3 до центру #0: 219.701
Відстань від пікселя №3 до центру #1: 3.27555
Відстань від пікселя №3 до центру #2: 171.288
Піксель №3 найближчий до центру #1
Відстань від пікселя №4 до центру #0: 214.202
Відстань від пікселя №4 до центру #1: 168.566
Відстань від пікселя №4 до центру #2: 3.77142
Піксель №4 найближчий до центру #2
Відстань від пікселя №5 до центру #0: 3.9863
Відстань від пікселя №5 до центру #1: 218.794
Відстань від пікселя №5 до центру #2: 218.805
Піксель №5 найближчий до центру #0
Відстань від пікселя №6 до центру #0: 216.415
Відстань від пікселя №6 до центру #1: 3.09403
Відстань від пікселя №6 до центру #2: 171.842
Піксель №6 найближчий до центру #1
Відстань від пікселя №7 до центру #0: 215.118
Відстань від пікселя №7 до центру #1: 168.927
Відстань від пікселя №7 до центру #2: 2.27181
Піксель №7 найближчий до центру #2
```

```
Масив відповідності пікселів і центрів:
1 2 0 1 2 0 1 2

Результат кластеризації:
Кластер #0
150 20 200
153 22 210
Кластер #1
255 140 50
251 141 51
252 138 54
Кластер #2
100 70 1
104 69 3
101 74 4
Нові центри:
151.875 21.25 206.25 – #0
249.828 138.047 51.7969 – #1
101.688 71.9688 3.25 – #2

Закінчення кластеризації.
```

Програмі достатньо двох ітерацій, щоб згрупувати дані в три кластера. Подивившись на центри двох останніх ітерацій, можна помітити, що вони практично залишилися на місці.

Цікаві випадки виникають при випадковому генеруванні пікселів. Згенерувавши 50 точок, які слід розділити на 10 кластерів, отримано 5 ітерацій. Згенерувавши 50 точок, які слід розділити на 3 кластери, я отримав усі 100 максимально допустимих ітерацій. З іншого боку, якщо кластерів мало, а точок багато, алгоритм часто завершується лише через перевищення максимально допустимої кількості ітерацій, оскільки деякі пікселі постійно стрибають з одного кластера в інший. Тем не менше, основна маса все одно остаточно визначається у свої кластери.

А тепер давайте перевіримо результат кластеризації. Взявши результати деяких кластерів з прикладу 50 точок на 10 кластерів, введено ці дані в Illustrator, і ось що вийшло (рис. 3.3).

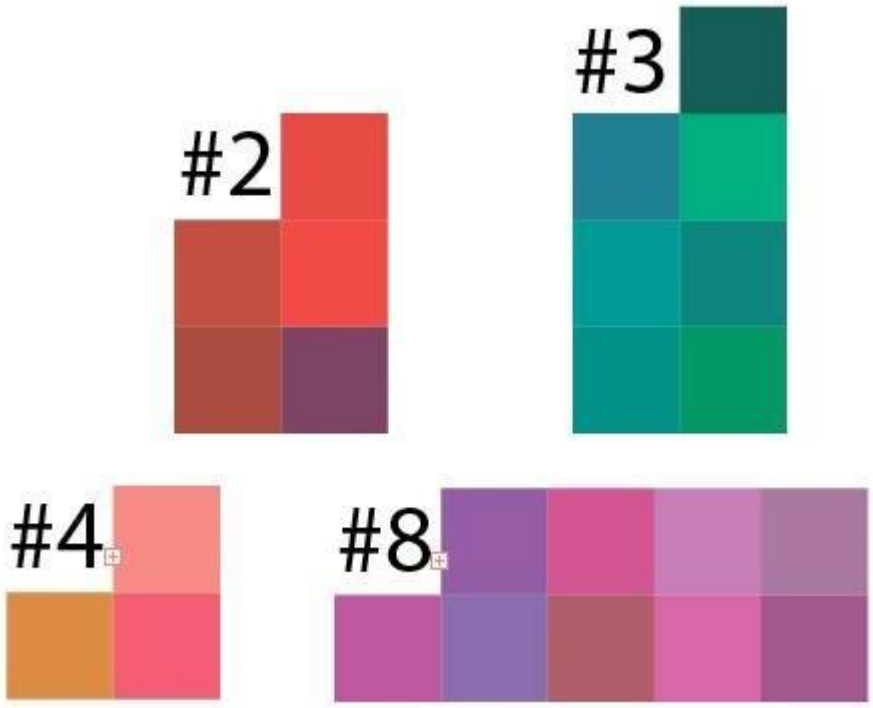

Рисунок 3.3 – Результат кластеризації

Видно, що в кожному кластері переважають певні відтінки кольору, і важливо розуміти, що пікселі були вибрані випадково. Аналогією до такого зображення в реальному житті є, наприклад, картинка, на який випадково намалювали всі кольори, і важко виокремити області схожих кольорів.

Допустимо, у нас є фото. Острів ми можемо визначити як один кластер, але, збільшуючи зображення, ми бачимо, що він складається з різних відтінків зеленого (рис. 3.4).

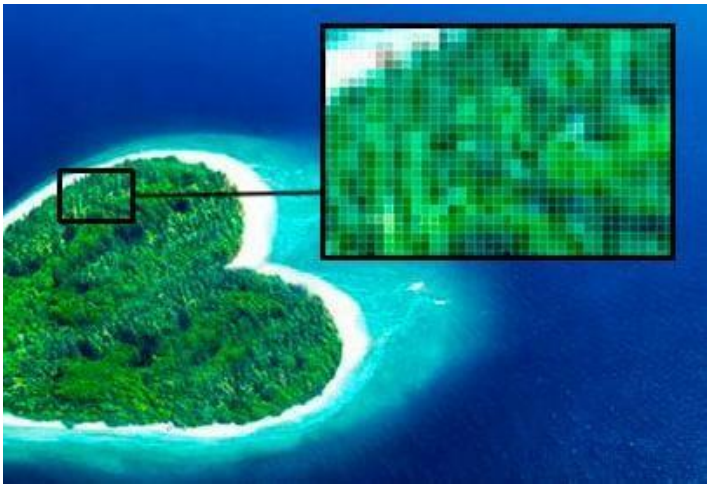

Рисунок 3.4 – Острів кластер

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і оновлений алгоритм онлайн кластеризації даних на основі методу *K*-Means (додаток А).

Було проведено аналіз методів онлайн достовірної кластеризації даних, було розглянуто різні підходи та алгоритми, спрямовані на обробку потокових даних. Особлива увага була приділена методу *K*-Means, який є одним з популярних методів кластеризації.

Один із важливих висновків полягає в тому, що багато існуючих алгоритмів кластеризації призначені для статичних наборів даних і можуть показати неоптимальні результати при роботі з нестаціонарними даними. Це породжує потребу у розробці та оптимізації алгоритмів онлайн кластеризації. В ході дослідження був розроблений покращений алгоритм онлайн кластеризації на основі методу *K*-Means. Цей алгоритм враховує особливості потокових даних, такі як нестаціонарність та великі обсяги інформації. Покращення включають адаптивну швидкість навчання, пакетне навчання та ітеративне згасання центроїдів для підтримки актуальності моделі.

Отримані результати дозволяють вважати розроблений алгоритм ефективним і перспективним для використання в реальних умовах, де дані змінюються з часом. Проте важливо враховувати, що ефективність алгоритму може залежати від конкретного контексту завдання та особливостей вхідних даних.

Загальною тенденцією є пошук більш гнучких та адаптивних методів, спроможних ефективно кластеризувати дані в режимі реального часу. Подальше дослідження та розвиток таких методів можуть призвести до ще більш точних та швидких систем аналізу потокових даних.

Результати дослідження апробовано у вигляді 1 тези доповіді під час Міжнародної науково-практичної конференції «Innovative scientific research: theory and practice» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Smith, J. (2020). *Hierarchical Clustering Algorithms: Theory and Applications*. New York: Springer.

2. Johnson, M., & Davis, R. (2021). *A Comparative Study of Hierarchical Clustering Methods for Big Data*. *Journal of Machine Learning Research*, 18(2), 45–68.

3. Brown, A., & White, S. (2019). *Enhancements in Hierarchical Agglomerative Clustering for Time–Series Data*. In *Proceedings of the International Conference on Data Science* (pp. 112–126). Chicago, IL: IEEE.

4. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738–124746.

5. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19–27.

6. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5–12.

7. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5–13.

8. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25–36.

9. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64–72.

10. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57–70.

11. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS–S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium*, September 28, 2023, Kyiv–Uzhorod, Ukraine, pp. 25–27.

12. Yakovleva O., Kovač M., Ardasov V. & Yeremenko I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities, *Public Administration and Regional Development,* 19(1), pp. 158–184.

13. Bezdek, J. C. (1980). A convergence theorem for the fuzzy ISODATA clustering algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2(1), 1–8. DOI: 10.1109/TPAMI.1980.4766964

14. Shafronenko, A., Bodyanskiy, Ye., & Rudenko, D. (2019). Online neuro fuzzy clustering of data with omissions and outliers based on completion strategy. In Proceedings of The Second International Workshop on Computer Modeling and Intelligent Systems (CMIS–2019), Zaporizhzhia, 2019 (pp. 18–27).

15. Itakura, F. (1975). Maximum prediction residual principle applied to speech recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing, 23(1), 67–72. DOI: 10.1109/TASSP.1975.1162641.

16. Kohonen, T. (1995). Self–Organizing Maps. Berlin: Springer. DOI: 10.1007/978–3–642–56927–2.

17. Kokshenev, I., Bodyanskiy, Ye., Gorshkov, Ye., & Kolodyazhniy, V. (2006). Outlier resistant recursive fuzzy clustering algorithm, Computational Intelligence: Theory and Applications. Advances in Soft Computing, 38, 647–652.

18. Dave, R. (1991). Characterization and detection of noise in clustering. Pattern Recognition Letters, 12(11), 657–664. DOI: 10.1016/0167–8655(91)90002–4.

19. Young, F. W., & Hamer, R. M. (1994). Theory and Applications of Multidimensional Scaling. Hillsdale, NJ: Erlbaum.

20. Sepkovski, J. J. (1974). Quantified coefficients of association and measurement of similarity. International Journal of Association Mathematics, 2(6), 135–152.

21. Zhao, F., Jiao, L., & Liu, H. (2011). Fuzzy c–means clustering with nonlocals partial information for noisy image segmentation. Frontiers of Computer Science, 5(1), 45–56. DOI: 10.1007/s11704–010–0393–8.

22. Zhou, J., Wang, Q., & Hung, C. C. (2017). Credibilistic clustering algorithms via alternating cluster estimation. Journal of Intelligent Manufacturing, 28, 727–738. DOI: 10.1007/s10845–014–1004–6.

23. Zhou, J., Wang, Q., Hung, C.–C., & Yi, X. (2015). Credibilistic clustering: the model and algorithms. International Journal of Uncertainty, Fuzziness and Knowledge–Based Systems, 23(4), 545–564. DOI: 10.1142/S0218488515500245.

24. Bodyanskiy, Ye., Kolodyazhniy, V., & Stephan, A. (2002). Recursive fuzzy clustering algorithms. In Proc 10th East West Fuzzy Coll. 2002, Zittau–Gërlitz, HS (pp. 276–283).

25. Mahalanobis, P.C. (1936). On the generalized distance in statistics. Proceedings National Institute of Sciences of India, 2(1), 49–55. DOI: 10.1007/s13171–019–00164–5.

26. Park, D.C., & Dagher, I. (1984). Gradient based fuzzy c–means (GBFCM) algorithm. In IEEE International Conference on Neural Networks, June 28 – July 2, 1984: proceedings (pp. 1626–1631). DOI: 10.1109/ICNN.1994.374399.

27. Krishnapuram, R., & Keller, J.M. (1993). A Possibilistic Approach to Clustering. IEEE Transactions on Fuzzy Systems, 1, 98–110. DOI: 10.1109/91.227387.

28. Liu, B. (2006). A survey of credibility theory. Fuzzy Optimization and Decision Making, 4, 387–408. DOI: 10.1007/s10700–006–0016–x.

29. Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., & Sohler, C. (2012). StreamKM++: A clustering algorithm for data streams. Journal of Experimental Algorithmics (JEA), 17, 2.4.

30. Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. In Proceedings of the 75 Thirtieth international conference on Very large data bases–Volume 30 (pp. 852–863).

31. Meesuksabai, W., Kangkachit, T., & Waiyamai, K. (2011). HUE–Stream: evolution based clustering technique for heterogeneous data streams with uncertainty. In Advanced Data Mining and Applications (pp. 27–40).

32. Rodrigues, P. P., Gama, J., & Pedroso, J. P. (2008). Hierarchical clustering of timeseries data streams. Knowledge and Data Engineering, IEEE Transactions on, 20, 615–627.

33. Han, J., Kamber, M., & Pei, J. (2006). Data mining: concepts and techniques. Morgan kaufmann.

34. Guha, S., Rastogi, R., & Shim, K. (1998). CURE: an efficient clustering algorithm for large databases. ACM SIGMOD Record, 73–84.

35. Guha, S., Rastogi, R., & Shim, K. (2000). ROCK: A robust clustering algorithm for categorical attributes. Information systems, 25, 345–366.

36. Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: an efficient data clustering method for very large databases. ACM SIGMOD Record, 103–114.

37. Udommanetanakit, K., Rakthanmanon, T., & Waiyamai, K. (2007). E–stream: Evolution–based technique for stream clustering. In Advanced Data Mining and Applications (pp. 605–615).

38. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, IEEE Access, vol. 11, pp. 126938-126949.

39. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, International Journal of Academic and Applied Research, 7(11), pp. 134-145.

40. Захаров Є. (2023) Особливості методів онлайн достовірної кластеризації даних, Abstracts of X International Scientific and Practical Conference «Innovative scientific research: theory and practice», (November 21 – 24, 2023). Stockholm, Sweden, pp. 480-484.